Path Graphs: Iterative Path Space Filtering

XI DENG, Cornell University, USA
MILOŠ HAŠAN, Adobe Research, USA
NATHAN CARR, Adobe Research, USA
ZEXIANG XU, Adobe Research, USA
STEVE MARSCHNER, Cornell University, USA

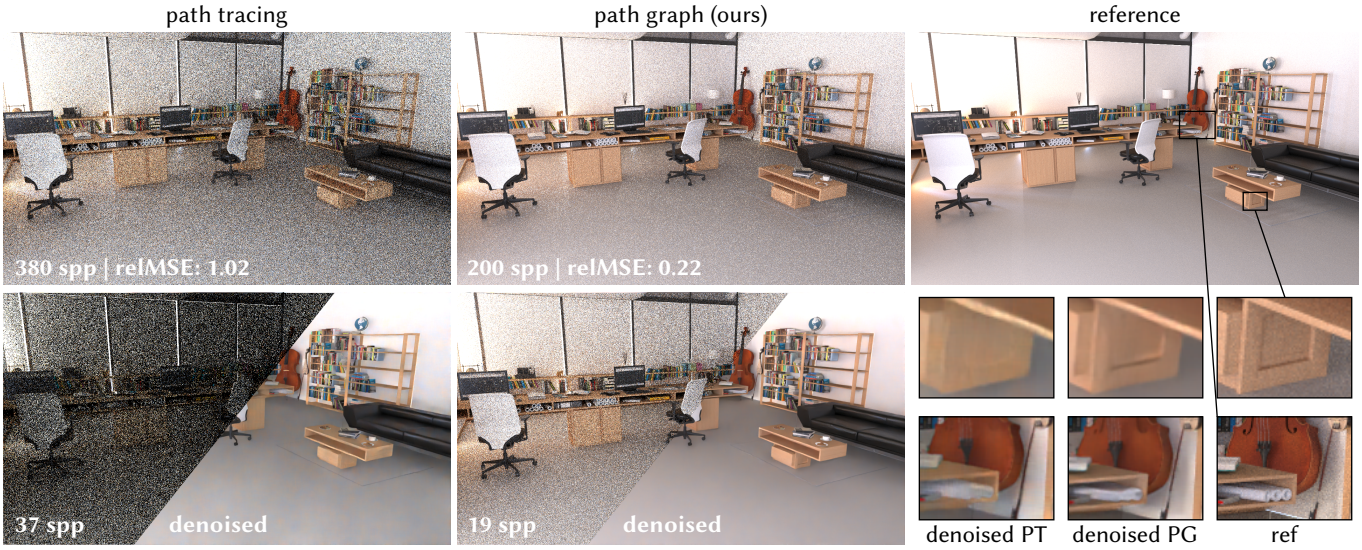

Fig. 1. Equal-time comparisons showing the improvement provided by the path graph in an indirect illumination-dominated scene. The top row shows a higher sample count; 200 samples processed with the path graph takes less time than path tracing 380 samples without the path graph; noise is reduced substantially. The bottom row shows the results of denoising a lower sample count; 19 path graph passes take less time than 37 samples of path tracing, and the denoised result is markedly improved in resolving details of illumination. Scene: OFFICE.

To render higher quality images from the samples generated by path tracing with a low sample count, we propose a novel path reuse approach that processes a fixed collection of paths to iteratively refine and improve radiance estimates throughout the scene. Our method operates on a *path graph* consisting of the union of the traced paths with additional neighbor edges inserted among clustered nearby vertices. Our approach refines the initial noisy radiance estimates via an aggregation operator, treating vertices within clusters as independent sampling techniques that can be combined using MIS. In a novel step, we also introduce a propagation operator to forward the refined estimates along the paths to successive bounces. We apply the aggregation and propagation operations to the graph iteratively, progressively refining the radiance values, converging to fixed-point radiance estimates with lower variance than the original ones. We also introduce a decorrelation (final gather) step, which uses information already in the graph and is cheap to compute, allowing us to combine the method with standard denoisers. Our approach is lightweight, in the sense that it can be easily plugged into any standard path tracer and neural final image denoiser. Furthermore, it is independent of scene complexity, as the graph size only depends on image resolution and average path depth. We demonstrate that our technique leads to realistic rendering results starting from as low as 1 path per pixel, even in complex indoor scenes dominated by multi-bounce indirect illumination.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Ray tracing**.

Additional Key Words and Phrases: ray tracing, global illumination, Monte Carlo, path graph, path space filtering

ACM Reference Format:
Xi Deng, Miloš Hašan, Nathan Carr, Zexiang Xu, and Steve Marschner. 2021. Path Graphs: Iterative Path Space Filtering. *ACM Trans. Graph.* 40, 6, Article 1 (December 2021), 15 pages. https://doi.org/10.1145/3478513.3480547

Authors' addresses: Xi Deng, Cornell University, 345 Gates Hall, Ithaca, NY, 14850, USA; Miloš Hašan, Adobe Research, 345 Park Avenue, San Jose, CA, 95110, USA; Nathan Carr, Adobe Research, 345 Park Avenue, San Jose, CA, 95110, USA; Zexiang Xu, Adobe Research, 345 Park Avenue, San Jose, CA, 95110, USA; Steve Marschner, Cornell University, 313 Gates Hall, Ithaca, NY, USA.

0730-0301/2021/12-ART1 $15.00
https://doi.org/10.1145/3478513.3480547

1 INTRODUCTION

In rendering, variants of Monte Carlo global illumination algorithms are ubiquitous. Among these, standard forward path tracing with direct illumination (next-event estimation) remains most common. Bidirectional approaches are also used, though their benefits do not always outweigh the extra costs and complexity.

Certain practically important scenarios remain challenging for all of these algorithms, such as indoor scenes lit predominantly by external illumination entering through windows. Forward path tracing struggles to connect shading points to the external illumination sources. Bidirectional approaches do not fully resolve this issue, since targeting the light through the windows becomes a non-trivial research topic by itself. Path guiding and denoising approaches have been shown to help. However, sparse sampling and reconstruction often work better with more information about the underlying scene (e.g. in the form of additional feature buffers), rather than just operating on the final radiances. We can go even further to extract more information, using knowledge about paths and their relationships, ultimately leading to more accurate input for a final denoiser.

Our goal is to extract more information from the paths constructed by the light transport method during the tracing of a single sample per pixel. This is the problem of *path reuse*: designing estimators that combine information available on a path vertex with information from nearby vertices, without the need to trace more rays. Many path reuse methods have been proposed [Davidovič et al. 2010; Keller et al. 2014; West et al. 2020], but in all of them, the reuse is largely local. Instead, we study how to make this process global: we take a local radiance estimate at a vertex, improve it through path reuse from nearby vertices, and propagate the improvement to other radiance estimates on other surfaces that may receive light from this vertex.

In the context of forward path tracing, each pixel sample gives rise to a "path," which is really a tree: a vertex (shading point) is typically connected with one edge to a light source and another edge to a continuation vertex, gathering illumination by recursively continuing the tree. Our approach is to take the union of such per-pixel trees, and to further extend it by adding information-sharing edges between spatially nearby vertices, by grouping vertices into clusters of size K (on average), and inserting edges among vertices in each cluster.

The resulting graph has three kinds of edges: *light edges* sampled for next-event estimation, *continuation edges* created through BSDF sampling to extend paths, and *neighbor edges*, connecting to spatial neighbors within each cluster. Neighbor edges do not represent light path segments, but instead denote information flow during path reuse. Given such a graph, we compute an improved per-pixel radiance estimate by iteratively applying two operations: aggregation (which locally improves the illumination estimate at each shading point through path reuse within each cluster) and propagation (which refines the estimate by propagating over an additional light bounce). As in radiosity methods, our solution converges to a steady state. We discuss how to ensure the convergence of the process. The additional computation needed for the construction and analysis of such a graph is relatively cheap, but it can significantly reduce the noise of the resulting estimate without tracing any extra paths. Like other path filtering methods, this method introduces bias by assuming the same incoming irradiance across a neighborhood. However, the resulting smoothing only affects the image via additional reflections, which reduces the visibility of this bias. We show later in the paper an error analysis to illustrate how this bias appears in images, showing its effects are subtle.

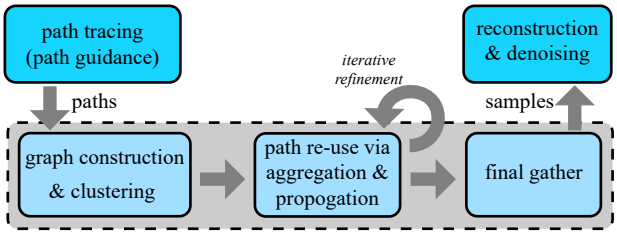

Fig. 2. Our method fits in between the traditional steps of path tracing and denoising. This means that it can provide additional benefits on top of (rather than replacing) techniques like modern CNN denoisers and path guiding techniques.

We believe ours is the first Monte Carlo light transport method that globally refines the estimate extracted from a given set of paths. While multiple path-reuse mechanisms have been proposed in previous work, and the idea of iterative refinement of a light transport estimate has been used extensively in classical radiosity approaches (using e.g. Gauss-Seidel or Jacobi iteration schemes), the combination of these concepts is largely missing from existing research. The contributions of our paper include:

- Formalizing the problem of reconstructing an image from traced light paths in a graph framework, where the initial estimate can be iteratively refined through the application of two operators (aggregation and propagation) over edges of the graph, with no need to trace any further rays.
- An efficient aggregation operator derived using multiple importance sampling (MIS). A straightforward application of MIS requires $O(K^2)$ pdf evaluations to combine K sampling techniques, but we show that an $O(K)$ approach is possible for path reuse through MIS within clusters.
- An iterative solver to reach the steady state of the above aggregation and propagation operators, and an analysis of its convergence.
- A decorrelation (final gather) pass that produces images suitable for standard denoisers, at minimal additional cost and using information already in the path graph.

The design of our algorithm leads to the following benefits:

- Independence of scene complexity; the graph size only depends on image resolution and average path depth, and using the graph does not require any further queries to scene geometry. This is beneficial in scenes where ray-tracing is particularly expensive.
- An ability to integrate into any rendering system, using an arbitrary path tracer to create the graph and an arbitrary final denoiser (Fig. 2). Path guiding methods could also be combined with our technique.

We demonstrate our approach on a number of indoor architectural scenes with indirect-dominated lighting conditions. Our method could benefit practical renderers at a minor performance cost and software engineering effort. We expect the benefits to be greatest in domains that require very complex scenes, where ray-tracing is most expensive, and where fast GPUs cannot fit the entire scene due to their limited memory (since they can still fit our path graph). We believe the scene-independent nature of the method, and its orthogonality to denoising and path guiding, could make it a good addition to practical rendering systems for such situations.

2 RELATED WORK

Radiosity methods. Early *radiosity* approaches to global illumination solve for diffuse inter-reflection among surfaces as a steady state energy flow problem [Goral et al. 1984; Hanrahan and Salzman 1991]. The problem takes the form of a (dense) linear matrix system (though further improvements address the dense nature of the matrix, such as stochastic and hierarchical radiosity). The radiosity matrix consists of form factors between discretized surface patches that form the scene, essentially encoding a one-bounce transport operator. While the solution to the linear system can be reached using direct solvers, many implementations rely on iterative approaches such as Jacobi or Gauss-Siedel. These iterative solvers propagate light between surface locations, improving estimates until a steady solution is encountered. Our approach takes inspiration from these early works by iteratively refining estimates over a graph of path vertices (shading and light points) to improve the solution. We do so, however, in the context of modern Monte Carlo path-tracing capturing arbitrary BRDFs and lighting, including image-based lighting. Moreover, our operators are always sparse.

Monte Carlo methods. The rendering equation [Kajiya 1986] can be effectively evaluated via Monte Carlo path tracing, which has been widely used in the graphics industry for realistic global illumination computation. However, standard path tracing can require a large number of samples per pixel (spp) to reduce the variance and compute a noise-free image, and in complex scenes the samples can be expensive. Various more advanced Monte Carlo methods, like bidirectional path tracing [Chaitanya et al. 2018; Lafortune and Willems 1993; Veach 1997; Veach and Guibas 1995a], Metropolis light transport [Cline et al. 2005; Pauly et al. 2000; Veach and Guibas 1995b], path guiding [Hey and Purgathofer 2002; Jensen 1995; Müller et al. 2017; Vorba et al. 2014] and portal sampling [Bitterli et al. 2015], have presented additional sampling techniques that can produce better results in challenging scenes; however, path tracing is still the most common choice for industrial applications. We base our method on standard Monte Carlo path tracing; we can utilize the noisy data from 1-spp path tracing to extract enough information for high-quality rendering.

Previous work has extensively studied sparse sampling and reconstruction for path tracing, in order to achieve high-quality low-spp rendering (see a survey presented by [Zwicker et al. 2015]). Recently, deep learning based de-noising techniques [Bako et al. 2017; Chaitanya et al. 2017; Gharbi et al. 2019] have been introduced in this space and have outperformed traditional methods. These neural methods reconstruct a high-quality image from the noisy screen-space samples in path tracing. Our method can be combined with screen-space denoising techniques. We demonstrate that our path-graph filtering results can be effectively denoised by a standard deep denoiser, not specialized to our outputs.

Reusing light transport computation. Much effort has gone into finding ways to reuse paths to improve estimates of the rendering equation integral. Irradiance caching [Ward et al. 1988] computes accurate irradiance estimates at sampled spatial cache points, allowing for smooth and efficient indirect lighting approximation via interpolation; this technique has also been extended to radiance caching [Křivánek et al. 2008], enabling efficient indirect lighting computation for glossy surfaces. Several algorithms have also been developed for reusing paths to improve convergence in both the primal and gradient domains [Bauszat et al. 2017; Bekaert et al. 2002]. To reduce the code complexity of coupling those path extension techniques, Fascione et al. [2019] present the vertex graph, which stores path information as vertices and segments for efficient query and evaluation. Techniques exist that take advantage of temporal coherence to improve rendering efficiency (e.g. [Bitterli et al. 2020]); this could also be seen as a variant of path reuse but in a very different form. In contrast our work focuses on further improving the efficiency of reuse within a single frame, which may benefit in the future from additional temporal filtering. Keller et al. [2014; 2016] directly consider the noisy radiance estimates on the path in Monte Carlo path tracing; their method leverages a simple path-space filtering analogous to our aggregation operation to reduce variance and can use multiple aggregation passes for stronger smoothing, but it does not include propagation.

These path reuse methods have recently been extended by West et al. [2020], which we treat as state of the art for path reuse. Our work can be seen as a further, iterative extension of their path-space filtering application. Note that the continuous MIS concept was used as inspiration by West et al., but is not actually necessary to derive theirs or our aggregation approaches, which can be seen as a discrete MIS combining a finite number of vertices in a neighborhood. We extend their neighborhood aggregation by alleviating the $O(K^2)$ aggregation complexity to $O(K)$. Furthermore, our method can refine the radiance estimates through a new propagation and iteration approach. We also introduce a new decorrelation (final gather) step at minimal additional cost, making path reuse methods compatible with Monte Carlo denoisers, which was not trivial before.

Multiple importance sampling. The multiple importance sampling (MIS) [Veach 1997; Veach and Guibas 1995b] has been widely used in Monte Carlo rendering to combine different Monte Carlo estimators. Recently, several advanced MIS techniques [Grittmann et al. 2019; Karlík et al. 2019; Kondapaneni et al. 2019] have been proposed to improve upon traditional MIS heuristics. West et al. [2020], already mentioned above, introduce a continuous MIS that enables properly combining an arbitrary (even uncountably infinite) set of estimators, and showed how this can be applied this to path reuse, closely related to our aggregation. Inspired by their path filtering application, we present an aggregation operation in our path graph computation that can be understood in an MIS framework; we combine it with a

propagation operation, which iteratively improves and propagates radiance estimates in the graph.

Virtual point lights. Our approach is also related to prior work based on virtual point lights (VPL), which are also a form of path reuse. [Hašan et al. 2009, 2007; Keller 1997; Walter et al. 2006, 2012]. The VPL-based methods usually distribute virtual lights in the scene by tracing paths from the light sources and then reuse these virtual lights to compute indirect lighting by connecting them with shading points. Some methods [Ou and Pellacini 2011; Wang et al. 2013] cluster VPLs and refine lights locally, which resembles our clustering of shading points, but our overall approach is quite different: it uses forward path tracing and does iterative refinement and image reconstruction in a very different way. Other works [Davidovič et al. 2010; Segovia et al. 2006] also propose creating virtual lights from the camera subpaths, developing techniques which turn out to be quite relevant to our work. We do not treat the vertices on the paths as virtual lights (at least not in an obvious sense), but like the VPL techniques, we do combine the vertex estimates to achieve path reuse.

Photon mapping. Photon mapping techniques [Hachisuka et al. 2008; Jensen 1996; Knaus and Zwicker 2011] are a classical approach for global illumination computation. Similar to VPL-based methods, photon mapping methods also trace and reuse light subpaths; they introduce some bias by merging nearby vertices while assuming their incoming light subpaths are still valid. Gathering photons is often based on a kernel density estimation framework; previous work has presented many advanced kernel functions to improve the reconstruction accuracy and efficiency [Jakob et al. 2011; Kaplanyan and Dachsbacher 2013; Schjøth et al. 2008], recently also including neural kernels [Zhu et al. 2020]. This technique has also been combined with (bi-directional) path tracing [Georgiev et al. 2012; Hachisuka et al. 2012; Křivánek et al. 2014] and also extended to unbiased solutions [Deng et al. 2019; Qin et al. 2015]. Our path aggregation is very similar to the photon gathering process; however, we leverage camera (instead of light) path vertices with noisy Monte Carlo radiance estimates (instead of light energies), and our aggregation is based on multiple importance sampling (MIS), rather than kernel density estimation.

3 PATH GRAPH FRAMEWORK

In this section, we will define the path graph framework in an abstract form, while in the next section, we will introduce our specific aggregation operators and further details.

3.1 Notation

We will use the following notation conventions, summarized in Table 1. We construct a graph $G = (V, E)$, given by vertices V and directed edges E. The vertices represent points on the scene surfaces and lights. We will use indices j, k, l for vertices, and indices e, e' for edges. (We avoid index i to avoid confusion with "incoming.") The edges are equivalent to pairs of vertex indices and are directed in the direction of light flow. Edges can have associated weights w_e.

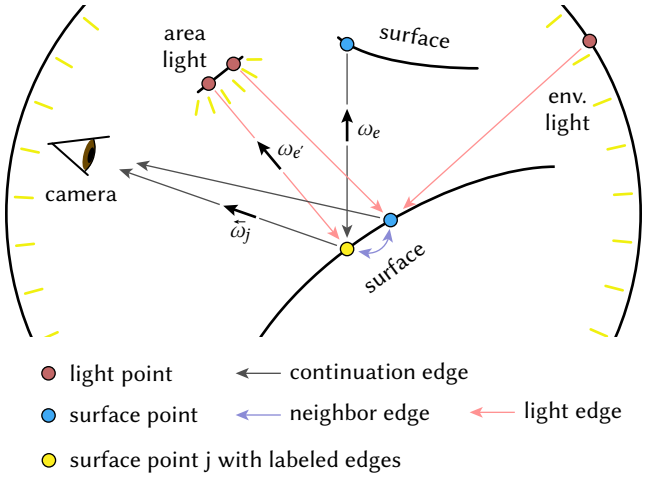

Fig. 3. Illustration of the path graph construction process from paths created during a standard path tracing pass with direct illumination connections.

Surface shading points (path vertices) will be denoted by $\mathbf{x}_j$. If a light connection succeeds (whether through next event estimation or through hitting a light source directly), we will denote the corresponding light points $\mathbf{y}_k$. All positions and directions will be assumed to be in world space. The light points can be at infinity for environment lights, so they can be considered as 4-element vectors in homogeneous coordinates (with the fourth element being zero for points at infinity).

Outgoing quantities at a shading point (i.e. the ones pointing towards the camera side of the path), such as outgoing directions and radiances, will be denoted with a left-facing arrow. Namely, $\overleftarrow{\omega}_j$ is the outgoing direction at the j-th shading point, and $\overleftarrow{L}_j$ is the outgoing radiance in direction $\overleftarrow{\omega}_j$.

We index incoming quantities by the corresponding edge e whose endpoint is j, rather than by the vertex index j, since there could be zero, one, or two valid direct edges for any shading point, and since not every shading point has an incoming indirect edge (paths have to eventually stop).

The BSDF at the shading point $\mathbf{x}_j$ will be denoted by $f_j(\omega)$, where the vector ω is assumed to vary over incoming directions. We are incorporating the dependence on $\mathbf{x}_j$ and $\overleftarrow{\omega}_j$ into f_j; the full definition would thus be $f_j(\omega) = f_s(\mathbf{x}_j, \overleftarrow{\omega}_j, \omega)$, where f_s is the spatially-varying BSDF.

The pdf from which the direction ω_e was sampled will be denoted as $p_e(\omega)$ in the solid angle measure at $\mathbf{x}_j$; any other measures can be converted to solid angle. The pdf p_e again incorporates the shading point $\mathbf{x}_j$, its outgoing direction $\overleftarrow{\omega}_j$, as well as the chosen light, in case e is a direct connection to a light.

3.2 Path tracing summary

In terms of the above notation, a standard path tracing implementation with next event estimation will operate as follows: First, through each pixel we will trace a single path tree, connecting each surface hit to a light and then continuing recursively. When a surface point is hit, the outgoing radiance in the negative ray direction

Table 1. Notation used in the paper.

Symbol	Definition
$\mathbf{x}_j$	surface shading point
$\mathbf{y}_k$	light point (could be point at infinity)
j, k, l	vertices (or vertex indices)
e, e'	edges (or edge indices)
w_e	weight of edge e
X	set of shading points
Y	set of light points
$V = X \cup Y$	set of path graph vertices
E	set of path graph edges
$G = (V, E)$	path graph
ω	direction on incoming sphere
$f_j(\omega)$	BSDF at $\mathbf{x}_j$
$\overleftarrow{\omega}_j$	outgoing direction at $\mathbf{x}_j$
ω_e	incoming direction along edge e
$\overleftarrow{L}_j$	outgoing radiance at $\mathbf{x}_j$
$\overleftarrow{L}_j^+$	updated (improved) value of $\overleftarrow{L}_j$
L_e	incoming radiance along edge e
L_e^+	updated (improved) value of L_e
$p_e(\omega)$	BSDF sampling pdf along edge e
K	cluster size (on average; may not be exact)
$C(j)$	indices in cluster containing j
$CE(j)$	continuation edges with endpoints in $C(j)$
$LE(j)$	light edges with endpoints in $C(j)$
$\hat{\sqcup}$	direct aggregation operator
$\sqcup$	indirect aggregation operator

is estimated by a combination of two Monte Carlo sampling techniques. The first one is direct light sampling (next event estimation), resulting in choosing a light point. The second technique is BSDF sampling, resulting in a path continuation direction. The tracing continues recursively to estimate the incoming radiance from this direction. Sometimes the recursive tracing hits a light source instead of a surface, which means the illumination from this direction is also treated as direct lighting.

This process computes a Monte Carlo estimate for the radiance at each pixel, and can be repeated to average more samples per pixel. The following exposition will assume a single sample per pixel (we will later show that we can also progressively average single-sample estimates). The data created during a single run of the above process can be turned into a *path graph* $G = (V, E)$ as follows.

3.3 Constructing path graphs

Let $X = \{\mathbf{x}_j\}$ be the world positions of N shading points (the union of all path vertices $\mathbf{x}_j$ on scene surfaces created during the path tracing process). Let $Y = \{\mathbf{y}_k\}$ be the set of N_L light points resulting from successful light connections (again note that some light points may be at infinity). The set of vertices of the path graph will be $V = X \cup Y$. We assume these sets are disjoint; if a BSDF-sampled ray hits an area light that also has a valid BSDF, we treat this as two separate edges and vertices.

The set of edges E will include three types of edges (see also Fig. 3). First, if the path continues from a shading point $\mathbf{x}_j$ and hits another valid surface point $\mathbf{x}_k$, we include these connections as *continuation edges* $e = kj$. Note that some shading points may not have continuation points, as paths have to end eventually, either through hitting a light that does not have a valid nonzero BSDF, exiting the scene, or being culled through Russian roulette.

Next, for every shading point $\mathbf{x}_j$ that has a valid light connection to a light point $\mathbf{y}_k$, we add this connection to the graph. We denote these connections as *light edges*. Note again that light edges can come from either light sampling (next event estimation) or hitting a light directly (this includes exiting the scene and hitting the environment light). Therefore, up to two light edges can exist at a given shading point. These are commonly weighted by the MIS power heuristic; we treat the corresponding MIS weights as edge weights on the light edges. Continuation edges do not have MIS weights in forward path tracing, so their edge weights are equal to 1.

Finally, we locally connect each shading point to approximately $K - 1$ nearby shading points. We split the set of shading points into clusters of size approximately K, and connect points within each cluster. Other metrics beyond Euclidean distance could be used if desired (e.g. considering the normal as well). We denote these connections as *neighbor edges*, in addition to the above light edges and continuation edges. Neighbor edges are not segments of physical light transport paths; they are instead denoting routes of information flow between neighbors during path reuse.

Furthermore, let us denote the set of indices in the cluster containing j by $C(j)$; by convention, we include $j \in C(j)$. We denote the set of continuation edges pointing towards any point in $C(j)$ by $CE(j)$, and the set of light edges pointing towards $C(j)$ by $LE(j)$.

3.4 Invalid samples and Russian roulette

Glossy/specular BSDF sampling can produce invalid samples in many commonly used models [Walter et al. 2007]. This can be modeled by modifying the domain over which the sampling pdf is defined. Instead of a hemisphere (or sphere, for transmissive BSDFs), we take the domain of the pdf to be the union with a special symbol $\varnothing$, which is returned in case of failure. In practice, this just means that pdfs will integrate to less than 1 over the (hemi)sphere, and this can be handled by our framework without any problems (standard MIS heuristics also handle this case correctly).

Conveniently, we can also use this property to implement Russian roulette, which can be theoretically modeled as intentionally increasing the failure rate of BSDF sampling. Say we would like to apply Russian roulette at point $\mathbf{x}_j$ with continuation probability q_j. We terminate the path with probability $1 - q_j$, and if the path has not been terminated, we simply modify the pdf $p_j(\omega)$ by multiplying with q_j. In summary, we just increased the probability of returning $\varnothing$, and the rest of our framework is unchanged.

3.5 Aggregation and propagation

Once we construct a path graph, we can use the information available in the cluster containing each shading point $\mathbf{x}_j$ to improve its estimate of its outgoing radiance. Furthermore, we can do this iteratively: given estimates $\overleftarrow{L}_j$ in a given iteration, we can compute

improved estimates $\overleftarrow{L}_j^+$, and then repeat this process until convergence. Two operators are key to this refinement: aggregation and propagation.

Aggregation operators. An aggregation operator combines the incoming radiance estimates (direct and indirect) at neighbors of $\mathbf{x}_j$ to compute an improved estimate of its outgoing radiance $\overleftarrow{L}_j^+$.

There is a subtle distinction between direct and indirect light aggregation. We therefore split aggregation into direct and indirect operators, and write them separately in an abstract form as

$$\overleftarrow{L}_j^+ = \hat{\bigsqcup}_{e \in LE(j)} L_e + \bigsqcup_{e \in CE(j)} L_e. \tag{1}$$

Here $\hat{\sqcup}$ is the direct aggregation operator that gathers and aggregates light edges within the neighborhood and computes an updated direct light estimate, while $\sqcup$ is the indirect aggregation operator that aggregates continuation edges (bringing indirect light from other surfaces). The main reason behind this separation is that only the indirect aggregation benefits from iterative refinement.

The aggregation is technically achieved by treating all points in the cluster containing $\mathbf{x}_j$ as independent sampling techniques that can be combined using MIS; the details of this will be presented in the next section.

Propagation operator. The propagation operator is much simpler: it updates the incoming indirect radiance estimate at shading point $\mathbf{x}_j$ by copying the outgoing radiance from its continuation point $\mathbf{x}_k$. That is, for a continuation edge $e = kj$,

$$L_e^+ = \overleftarrow{L}_k. \tag{2}$$

Combining the operators. We can combine the above operators, to get an iterative update of outgoing radiance at each shading point. In other words, we are looking for outgoing radiance values that form the *fixed point* of the combined aggregation/propagation operator. One can observe some similarity to traditional radiosity formulations. One corollary of the formulation is that direct aggregation only needs to be computed once; it does not change with iterative refinement, as propagation does not affect it. However, this obviously skips over a number of details. In the next section, we show how to actually define the aggregation operators, under what conditions the iteration converges, and how to finally obtain per-pixel estimates, and how to combine the framework with an external denoiser.

4 AGGREGATION AND PROPAGATION

In this section, we will first look at the general problem of combining N Monte Carlo estimators for a single integral, each with a different probability distribution. Next, we apply the idea to the design of our aggregation operator. Finally, we will complete the pipeline by introducing the propagation operator and the entire iterative pipeline.

4.1 Multi-sample Monte Carlo estimators

Consider the problem of computing an integral $I = \int_D f(x)\, dx$, by taking N random samples on the integration domain D. Rather than the simple case where the samples $x_1, \cdots, x_n$ are all independently taken from the same probability distribution, let us consider the case of different, separate pdfs per sample, $p_1(x), \cdots, p_n(x)$. We will assume that at least one $p_j(x)$ is non-zero for every x where $f(x)$ is non-zero. Below we discuss two approaches to derive an estimator for this problem, and show that they lead to equivalent results. Note: in this subsection, and only this subsection, we use x (with or without subscripts) to denote samples of any abstract domain, not necessarily related to path graph vertices.

Using the marginal density. Let us define the *marginal density* to be $\rho(x) = \sum_k p_k(x)$. The function $\rho(x)$ is a generalization of a pdf to n samples, and its integral over domain D is n. Our integral can now be computed by the unbiased estimator:

$$I \approx \sum_{j=1}^{n} \frac{f(x_j)}{\rho(x_j)}. \tag{3}$$

One can easily check that this estimator is unbiased:

$$\sum_{j=1}^{n} E\left[\frac{f(x_j)}{\rho(x_j)}\right] = \sum_{j=1}^{n} \int_D p_j(x) \frac{f(x)}{\rho(x)} dx = I. \tag{4}$$

In fact, the estimator clearly remains unbiased even if the function queries $f(x_j)$ are themselves independent random variables that give unbiased approximations to the true function values.

Using MIS. Combining the n different sampling techniques through multiple importance sampling [Veach 1997; Veach and Guibas 1995b] introduces weighting functions $w_j(x)$ that sum to unity, leading to an estimator

$$I \approx \sum_{j=1}^{n} w_j(x_j) \frac{f(x_j)}{p_j(x_j)}, \tag{5}$$

where the weights can be chosen using different heuristics. The commonly used *balance heuristic* makes the weights equal to the corresponding pdfs, normalizing to sum to 1 for every x:

$$w_j(x) = \frac{p_j(x)}{\sum_{j=1}^{n} p_k(x)}. \tag{6}$$

By plugging in this weighting approach into Eq. (5), we find that it simplifies to the same form as Eq. (3). Note also that the evaluation of the estimator requires n evaluations of the integrand f, but n^2 evaluations of the pdfs p_j, since we need the values of $p_k(x_j)$ for all pairs (k, j). As we will see, amortization can reduce this quadratic complexity , because within each cluster we need to compute a number of integrals that share the underlying pdf computations.

4.2 Designing aggregation operators

Recall that the aggregation operator turns the incoming direct and indirect radiance estimates of all points in the neighborhood of a given shading point x_j into an updated outgoing radiance estimate. In both the direct and the indirect case, we have a number of samples of incoming radiance in $C(j)$, with known pdfs. We can therefore treat them as independent sampling techniques, and combine them in a straightforward manner using the above estimator (Eq. (3)) to compute an updated outgoing radiance estimate. For the indirect aggregation operator, this becomes:

$$\bigsqcup_{e \in CE(j)} L_e = \sum_{e \in CE(j)} \frac{f_j(\omega_e)\, |n_j \cdot \omega_e|\, L_e}{\rho_e}, \tag{7}$$

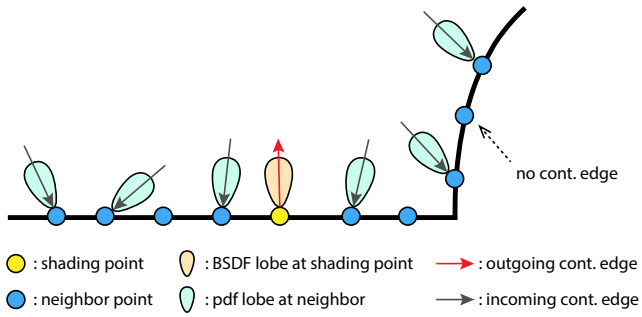

Fig. 4. The local neighborhood for the indirect aggregation operator around a shading point (yellow) . Neighbor points (blue) may have incoming radiance estimates (continuation edges) from indirect illumination (though not all neighborhood points have continuation edges, due to paths exiting the scene or terminating by Russian roulette). The aggregation operator combines the incoming radiances to produce an updated outgoing radiance estimate shown in red. The combination weights can be derived in several ways including MIS; see Sec. 4.2.

where

$$\rho_e = \sum_{e' \in CE(j)} p_{e'}(\omega_e). \tag{8}$$

The same idea can be applied to the direct aggregation operator:

$$\hat{\sqcup}_{e \in LE(j)} L_e = \sum_{e \in LE(j)} \frac{f_j(\omega_e)\, |n_j \cdot \omega_e|\, w_e\, L_e}{\hat{\rho}_e}, \tag{9}$$

where

$$\hat{\rho}_e = \sum_{e' \in LE(j)} p_{e'}(\omega_e). \tag{10}$$

4.3 Bias in aggregation

The aggregation operators are making the approximation that incoming radiance estimated at neighbors are valid at the original shading point. This approximation introduces bias. It is the exact same approximation made by West et al. [2020] and is closely related to the approximation made in photon mapping (vertex merging) approaches. While West et al. theoretically discuss making the aggregation unbiased, they do not implement it, as it would not be practical.

Instead, we can only obtain the following *conditional unbiased property*: If we assume the incoming radiance estimates are unbiased random approximations of their true values at x_j, then the results of the aggregation operators will be unbiased approximations to the true outgoing (direct and indirect) radiances. This follows directly from the estimator in *Eq.* (4). In other words, the bias does not come from the design of the aggregation operator, but from the assumption that incoming radiance estimates at neighbors are still valid at the original shading point.

4.4 Efficient aggregation

While these aggregation operators work well, there is a problem: they require $O(K^2)$ computation per shading point for neighborhoods with K points, as the computation of marginal densities ρ_e requires the evaluation of pdfs $p_{e'}(\omega_e)$ for all pairs (e, e') of edges incident to a cluster. This is a general problem when applying MIS to combine K estimators, and was also noted by West et al. [2020]. However, we note that these pdf evaluations $p_{e'}(\omega_e)$ are shared between aggregations happening at different shading points in a cluster. This lets us achieve $O(K)$ instead of $O(K^2)$ complexity per shading point.

Since the set of neighbors of a point is the same as the set of neighbors of any of its neighbors (namely, the cluster itself), we can apply the following *gather-scatter* approach to compute the indirect aggregation in $O(K)$ time per point.

(1) Gather: For each continuation edge $e = kj$ compute the marginal density of sampling ω_e:

$$\rho_e \leftarrow \sum_{k \in C(j)} p_k(\omega_e). \tag{11}$$

(2) Scatter: For each continuation edge $e = kj$ contribute outgoing radiance to each point $k \in C(j)$:

$$\tilde{L}_k \leftarrow \tilde{L}_k + \frac{f_k(\omega_e)\, |n_k \cdot \omega_e|\, L_e}{\rho_e}. \tag{12}$$

The same approach can be used for the direct aggregation operator, using the light edges rather than the continuation edges. Clearly, the gather and scatter operations both take $O(K)$ time per shading point; furthermore, the algorithm exactly computes the operator from Eq. (7), because $C(j) = C(k)$ is always the same set of indices in the cluster.

At a high level, this gather-scatter algorithm is analogous to one used by [Davidovič et al. 2010] (sec. 5 of their paper), though in their case, image-space tiles take the role of clusters, and the details of the scattered contribution differ from ours.

5 LINEAR SYSTEM SOLUTION

Recall that we defined two update operations, aggregation and propagation. These can be combined into a single update that improves the outgoing radiance values:

$$\tilde{L}_j^+ = \hat{\sqcup}_{e \in LE(j)} L_e + \sqcup_{e \in CE(j)} \tilde{L}_k, \text{ where } e = kj. \tag{13}$$

In this section we show how to ensure convergence of this iterative process.

Our goal can be stated as finding a vector of outgoing radiances $\tilde{L}_j$ that is a fixed point of the update rule in Eq. (13), and proving that the iteration converges to that fixed point. The operation is clearly linear in the outgoing radiance values, so it is convenient to rewrite it in matrix-vector form.

Since the direct aggregation has an input (L_e where e is a light edge) that does not depend on the variable being updated ($\tilde{L}_j$), its output is a constant in the iteration. Let $\mathbf{B}$ be the vector of direct radiances, improved by a single application of the direct aggregation operator. That is, $\mathbf{B}$ will have elements $\mathbf{B}_j = \hat{\sqcup}_{e \in LE(j)} L_e$. Next, we denote the unknown vector of fixed-point outgoing radiances by $\mathbf{X}$, with elements $\mathbf{X}_j = \tilde{L}_j$.

We next express the indirect aggregation and propagation as a matrix $\mathbf{M} = \mathbf{AP}$. The aggregation matrix $\mathbf{A}$ has rows corresponding to path vertices and columns corresponding to continuation edges. It is sparse and consists of one nonzero block corresponding to

each cluster. The elements within the blocks are defined by Eq. (7) and Eq. (8). The propagation matrix $\mathbf{P}$ has rows corresponding to continuation edges and columns corresponding to path vertices; it has a single nonzero value per row and at most one per column, in element (e, k) for every continuation edge $e = kj$. Their product $\mathbf{M}$ is square, mapping an outgoing total radiance per path vertex to an outgoing indirect radiance per path vertex. The linear system we are solving is

$$\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{B}. \tag{14}$$

We use a simple iteration scheme to find a fixed point of the combined aggregation-propagation operator, by repeated application of the simple update rule $\mathbf{X}^+ = \mathbf{M}\mathbf{X} + \mathbf{B}$. We can initialize $\mathbf{X}$ either to the noisy radiance estimate from path graph construction, or to zero; both options typically lead to a good solution within a few iterations.

5.1 Convergence of Jacobi iterative method

Since $\mathbf{M}$ has a zero diagonal (a point's outgoing radiance never contributes to updating itself), the simple update rule above is an instance of the well-known Jacobi iterative method.

Jacobi iteration is guaranteed to converge if the spectral radius of $\mathbf{M}$ (the magnitude of its maximum eigenvalue) is less than 1. We will show that this property holds under the condition that BSDF importance sampling always produces weights smaller than 1; afterwards we will present a technique that guarantees convergence even without this property.

Theorem. Assume the importance sampling weights of all materials in the scene (i.e. the values of the BSDF times cosine term divided by the corresponding sampling pdf) are always less than 1. Then the spectral radius of $\mathbf{M}$ is less than 1 and Jacobi iteration converges.

Proof. To prove that the spectral radius of $\mathbf{M}$ is less than 1, we will prove the stronger property that $\|\mathbf{M}\mathbf{X}\|_1 < \|\mathbf{X}\|_1$. This implies that $\|\mathbf{M}\mathbf{X}\|_1/\|\mathbf{X}\|_1$ is always less than 1, and the supremum of this ratio is known to bound the spectral radius.

First, it is clear that $\|\mathbf{P}\mathbf{X}\|_1 \le \|\mathbf{X}\|_1$, since propagation cannot increase energy; it merely reorders the values of $\mathbf{X}$ and omits some of them, since the first vertex on each path does not have an outgoing continuation edge. Therefore, it is enough to show that $\|\mathbf{A}\mathbf{X}\|_1 < \|\mathbf{X}\|_1$, which holds if for each cluster, the sum of outgoing indirect radiance values is less than the sum of incoming indirect radiance values. Intuitively, the indirect aggregation numerically "loses energy" in every cluster.

Take a cluster C. Let E denote the set of incoming radiance edges into this cluster, that is, $E = CE(j)$ for any $j \in C$ (these sets are all equivalent for any j). Using our definition of indirect aggregation in eq. Eq. (7), the sum S_C of outgoing indirect radiance values of this cluster can be written as:

$$S_C = \sum_{j \in C} \sum_{e \in E} \frac{f_j(\omega_e)\ |n_j \cdot \omega_e|\ L_e}{\rho_e}, \tag{15}$$

where

$$\rho_e = \sum_{j \in C} p_j(\omega_e). \tag{16}$$

We can change the order of summation and rewrite the sum as follows:

$$S_C = \sum_{e \in E} \left(\frac{\sum_{j \in C} f_j(\omega_e)\ |n_j \cdot \omega_e|}{\sum_{j \in C} p_j(\omega_e)} \right) L_e. \tag{17}$$

Recall that we want to prove $S_C < \sum_{e \in E} L_e$, which is true if the term in parentheses is bounded from above by 1. Note, the numerator contains the sum of BSDF-times-cosine values, while the denominator is the sum of the corresponding importance sampling pdf values. We assumed that all importance sampling weights are less than 1, or equivalently, each BSDF-times-cosine value is less than its corresponding pdf. Therefore, the sum of the BSDF-times-cosine values has to be less than the sum of their corresponding pdfs, finishing our proof. □

Clamping strategy. While we can design BSDF models and importance sampling strategies that always satisfy the sampling weight property (and in fact it is trivial for Lambertian BRDFs used in traditional radiosity approaches), it can be inconvenient in practice. Many rendering systems already implement complex materials that would be difficult to modify to never break this property.

We introduce a simple clamping approach that guarantees convergence even with the simple Jacobi solver. The idea is to enforce $\|\mathbf{M}\mathbf{X}\|_1 < \|\mathbf{X}\|_1$ by explicitly making sure each cluster's sum of outputs is less than its sum of inputs. Note that this is the same property that holds automatically when the sampling weight property is true. We compute the input and output sums, and clamp the latter to be at most $1 - \epsilon$ times the former, for every cluster and every iteration. We find that this simple technique guarantees convergence without visibly affecting the result in cases where convergence would occur anyway. However, we did indeed observe cases that would otherwise diverge.

5.2 Final gather

The solution of the above iterative process will generally have much lower variance than the original one-sample path tracing estimate; however, it has significant correlation between nearby points, and it shows cluster boundaries, making it unsuitable for applying image-space denoisers. Instead, we apply a "final gather" idea: after sufficient convergence with an appropriate cluster size K, we use a single iteration with $K = 1$ to compute final pixel values. This leads to a solution with lower variance than the original path tracing (especially under difficult indirect-dominated lighting), but with no pixel correlation. A major benefit of this final gather step is that its result can be fed to a standard Monte Carlo denoiser.

6 RENDERING SYSTEM

We implemented our method in the open-source renderer Mitsuba [Jakob 2010]; however, most path graph operations are accelerated on the GPU using CUDA. The implementation is summarized in pseudocode in Figure 5. The phases of our system consist of path graph construction (on the CPU), clustering (on the GPU), iterative aggregation and propagation (GPU), final gather (GPU), and optional denoising (CPU). We currently use the Intel Open Image Denoise [Intel 2020] on the CPU, to which we feed additional feature buffers (diffuse color, normal); any similar denoiser could be used.

Image **render**(scene):
  VertexSet X —*path vertices on surfaces*
  EdgeSet E_C, E_L, E_N —*edges of path graph*
  PerVertexData hitData —*sampling directions, BSDFs, PDFs*
  GBuffer auxBuffer —*normal, albedo per pixel*
  PerVertexRadiance $\overleftarrow{L}_d$, $\overleftarrow{L}$ —*direct and total outgoing radiance*
  $\overleftarrow{L}$, auxBuffer, X, E_C, E_L, hitData = **pathTrace**(scene)
  E_N = **cluster**(X)
  $\overleftarrow{L}_d$ = **aggregate**(E_L, E_N, L, hitData)
  for N_{iter} iterations:
    PerEdgeRadiance L —*indirect radiance on continuation edges*
    L = **propagate**(E_C, $\overleftarrow{L}$)
    $\overleftarrow{L}$ = $\overleftarrow{L}_d$ + **aggregate**(E_C, E_N, L, hitData)
  PerSampleRadiance L —*radiance of image space samples*
  L = **finalGather**(E_C, $\overleftarrow{L}$, hitData)
  return **reconstructAndDenoise**(L, auxBuffer)

PerEdgeRadiance **propagate**(E_C, $\overleftarrow{L}$):
  PerEdgeRadiance L
  for $e = kj$ in E_C:
    $L_e = \overleftarrow{L}_k$
  return L

PerVertexRadiance **aggregate**(E_a, E_N, L, hitData):
  PerVertexRadiance $\overleftarrow{L}$
  PerEdgeScalar ρ
  for $e = lj$ in E_a: *%gather*
    for k in $C(j)$: —$C(j) = \{k \mid jk \in E_N\} \cup \{j\}$
      ρ_e += hitData.pdf(k, e) —*pdf at vertex k evaluated for ω_e*
  for $e = lj$ in E_a: —*scatter*
    for k in $C(j)$:
      f = hitData.bsdf(k, e) —*bsdf at vertex k evaluated for ω_e*
      μ = hitData.cosine(k, e) —$|n_k \cdot \omega_e|$
      $\overleftarrow{L}_k$ += $f\, \mu\, L_e\, w_e / \rho_e$
  return $\overleftarrow{L}$

PerSampleRadiance **finalGather**(E_C, $\overleftarrow{L}_d$, $\overleftarrow{L}$, hitData):
  PerVertexRadiance $\overleftarrow{L}$
  $\overleftarrow{L}$ = $\overleftarrow{L}_d$ + **aggregate**(E_C, {}, **propagate**(E_C, $\overleftarrow{L}$), auxData)
  return **firstBounce**($\overleftarrow{L}$) —*the first vertex of each path*

Fig. 5. Algorithm summary, omitting clamping of outgoing radiance during aggregation for simplicity.

The aggregation/propagation phase is implemented in CUDA to leverage the GPU for these arithmetic-dense computations. This requires porting BSDF and pdf evaluation to CUDA. This is a fairly small part of the material system; importance sampling, texture mapping, procedurals, displacements and other details can stay within the path graph construction stage. Of course, one could port the entire system to the GPU, though we envision a method like ours to be most useful for complex scenes that do not fit into GPU memory.

Path graph construction. The points of the path graph are recorded during a standard path tracing process. For each pixel, a path is traced, and every time the path hits a surface a feature vector for the corresponding point is saved. This feature vector contains the incoming/outgoing vectors, as well as enough information to locally evaluate the point's BSDF and pdf. The path terminates when it exits the scene or by Russian roulette after a depth of 5, with the Russian roulette probability derived from the maximum color channel of the albedo. This provides all the vertices of the path graph, as well as the continuation and light edges.

To construct the clustering, N/K points are selected uniformly at random as cluster centers, then we place each point in the cluster corresponding to the nearest cluster center, using a spatial hashgrid on the GPU to accelerate the search for the nearest center. This is essentially one iteration of the k-means algorithm; we could run more iterations but did not find it necessary.

7 EXPERIMENTS AND RESULTS

We evaluate our path graph framework on several scenes, especially focusing on indoor environments with a strong indirect illumination component. This choice of examples follows the intuition that the primary benefit from our approach would occur in environments that contain strong lighting contribution from indirect paths. For example, a closed interior scene with light entering through windows and doors requires light to propagate several bounces before reaching the camera.

The hardware we use consists of an AMD Threadripper 2950 (48GB RAM), with 16 cores and 32 threads, and an Nvidia RTX2080Ti (11GB). The CPU we use could be considered higher-end than the GPU. A further opportunity to overlap CPU and GPU computation also exists, but is not currently utilized.

The additional memory used for recording graph data is around 800MB. The resolution of most of the images is 1280x720 pixels, with the exception of the CAUSTICS scene, which has an image resolution of 1024x1024 pixels.

Comparison to West et al. In Fig. 6, we show a comparison of our aggregation method to the one from the method of West et al. [2020]. We compare on two image tiles taken from indirectly lit areas of the images shown on the left. The two methods compared are the West et al. method (which is essentially equivalent to one application of the $O(K^2)$ aggregation operator, using K nearest neighbors insted of clustering), and our faster scatter-gather approach ($O(K)$ with propagation/iteration); we also use improved direct lighting through direct aggregation. Our method is both faster and has lower error, both when visualized directly and through the additional 1-sample final gather.

Note that our approach can consistently improve the results and reduce the variance. In general, we observe that our results with $K = 16$ are already of sufficient quality, and increasing it does not benefit the final result much. The results at $K = 16$ only take less than 1s to compute using our fast gather/scatter technique, giving a highly efficient algorithm that can be easily used in practice. After applying the final gather, the $K = 16$ results can be further effectively denoised, leading to further gains in rendering quality.

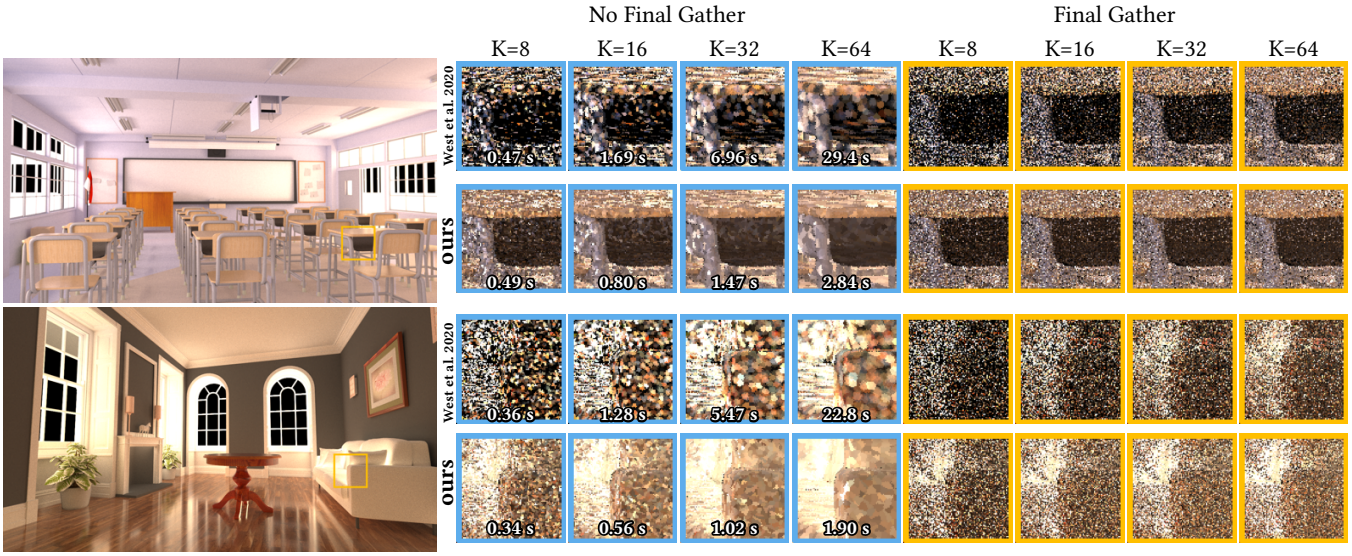

Fig. 6. Comparison of our aggregation method to West et al. [2020]. We compare on two image tiles taken from indirectly lit areas of the images on the left. West et al. is essentially one $O(K^2)$ aggregation using K nearest neighbors. Our method uses clustering and scatter-gather ($O(K)$) with propagation and iteration. Our method is both faster and has the lower error/noise, both when visualized directly and through an extra 1-sample final gather. Scenes, top to bottom: CLASSROOM, LIVING-ROOM-2.

In Fig. 8, we show an equal-time comparison to path tracing on three scenes. In the time it takes to run our method, path tracing can render several additional samples. Nevertheless, our result has lower variance. Furthermore, while denoising the results of both compared methods leads to smooth images in both cases, our RelMSE and visual quality are better. The rightmost column shows the reference computed with many samples per pixel. Note how our denoised solution matches the reference color tint closer than the denoised path tracing.

In Fig. 1, we show an equal-time comparison on a more complex scene, with difficult illumination conditions: the windows are mostly blocked by opaque window shades, and lighting is dominated by multi-bounce indirect illumination. The results show a clear improvement provided by the path graph iteration and propagation in this indirect-dominated scene. The top row shows a higher number of samples (200) with our method, which takes less time than standard path tracing with 380 samples; noise is reduced substantially in our result. The bottom row shows the results of denoising a lower sample count (19spp) with our method, which took less time than 37 samples of standard path tracing. Our denoised result is much better than denoised path tracing in terms of resolving details of illumination, shown in the insets on the right.

In Fig. 9, we show an equal-time (30 seconds) comparison against path tracing, path guiding [Müller et al. 2017] and bidirectional path tracing on six scenes. Although path guiding can sometimes beat our approach (path tracing + path graph) in RelMSE values, our method could theoretically be used in the future together with path guiding to further improve the efficiency. For heavily indirect-dominated scenes such as OFFICE, our method is significantly better in terms of RelMSE. Furthermore, our result quality is stable across different kinds of illumination (direct, indirect, etc).

To illustrate the effects of bias, we compare reference renderings on a variety of scenes including direct-lighting dominated scenes (LIVING-ROOM, DINING-ROOM), indirect-lighting dominated scenes (OFFICE) and the CAUSTICS scene. A subset is shown in Fig. 7, and more results can be found in the supplementary materials. These scenes show various materials including both diffuse and highly specular surfaces (e.g. polished wooden floor, golden ring). The resulting error is a combination of noise, aggregation bias and clamping bias. The error maps are absolute differences between path graph images and references. Note the error curves in the bottom row, showing that despite some bias, our method generally has much lower overall error than unbiased path tracing.

In Table 2, we report detailed timings for different parts of our pipeline. The 2nd to 4th columns are the average render time of 1 sample per pixel for both path tracing, revised path tracing with data recording, and the total execution time of our path graph iteration. The ratio is the number of samples that a path tracer could compute during the same time of 1spp using our method, with path sample recording overhead.

Temporal behavior. We currently do not utilize temporal coherence in our method. Despite this, running our method for each frame separately does not introduce unwanted temporal artifacts. The supplementary material includes a video showing the temporal behavior of our results. There is essentially no temporal coherence across frames (since paths and clusters are chosen differently each frame) and no low-frequency artifacts due to the presence of our final gather step. A possible future direction of utilizing temporal information would be to build our path graph over paths accumulated through frames, selectively dropping paths from older frames; the total graph size could stay constant.

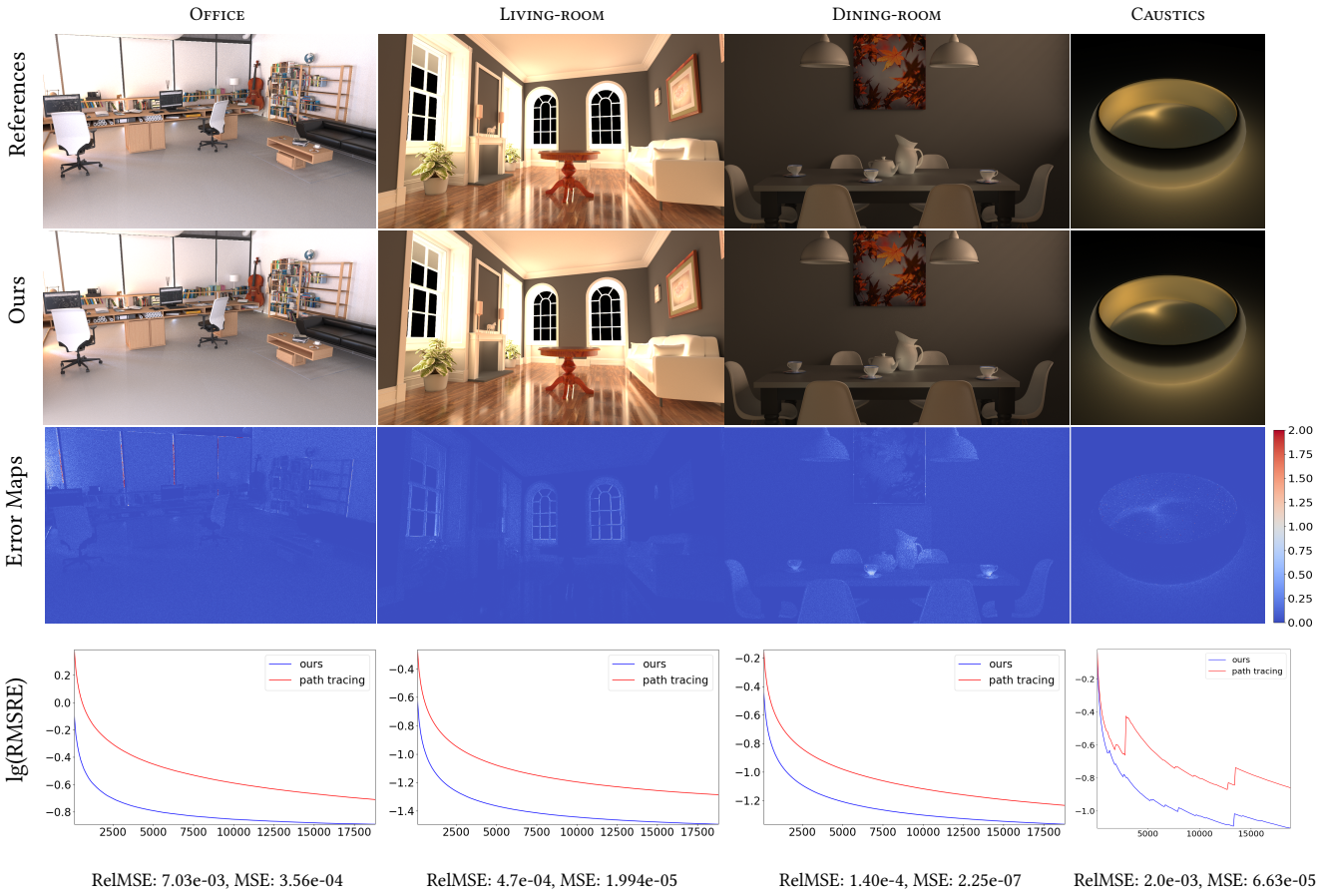

Fig. 7. Error analysis on images with high sampling rates. The first row shows converged path-traced reference images and the second row shows converged renderings with our method, both with indirect illumination only. The third row visualizes the difference, divided by the norm of the reference image. The fourth row shows the convergence of our method vs. path tracing for sampling rates from 100 spp to 20,000 spp; the vertical axis represents the root mean square relative error with respect to the reference image. The RelMSE and MSE between the methods are also listed at the bottom.

Table 2. Detailed timing of our pipeline steps. The 2nd to 4th columns are the average render time of 1 sample per pixel for both path tracing, revised path tracing with data recording, and the total execution time of our path graph iteration. The ratio is the number of samples that a path tracer could compute during the same time of 1spp using our method, with path sample recording overhead.

Scene Name	time (s)			ratio	relMSE for 1SPP		pathtr:ours SPP for equal time(30s)	Equal time RelMSE ratio path tr: ours
	path tr	ours			path tr	ours		
		record path tr	path graph					
VEACH-AJAR	0.28	0.5	0.75	4.5	39.6	7.72	107:24	1.80
STAIRCASE	0.59	0.7	0.61	2.2	20.6	5.79	50:23	1.68
OFFICE	0.8	0.9	0.63	1.9	491	48.8	37:19	4.74
KITCHEN	0.5	0.7	0.62	2.6	17.7	3.41	60:23	2.02
DINING-ROOM	0.45	0.6	0.60	2.7	5.09	1.21	66:24	1.38
CLASSROOM	0.49	0.7	0.60	2.7	14.7	4.79	61:23	1.05
LIVING-ROOM-2	0.51	0.8	0.77	3.1	1.28	0.31	58:18	1.15
DINING-ROOM-2	0.67	0.9	0.52	2.1	31.4	14.2	45:21	0.93
CAUSTICS	0.25	0.3	0.123	1.6	8.27	4.38	110:65	1.75

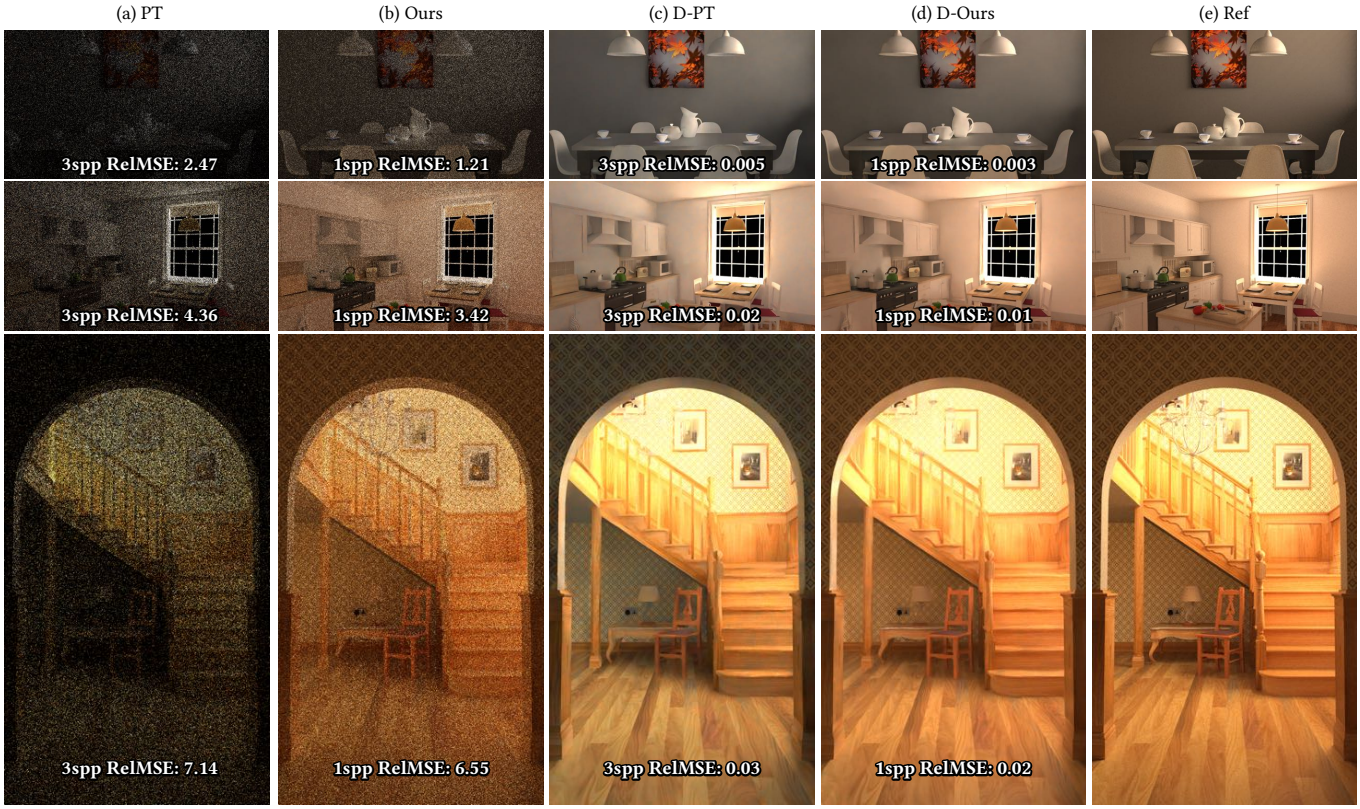

Fig. 8. Equal time comparison between path tracing and our method (path tracing + path graph). We are showing indirect illumination only. In the time it takes to run our method, path tracing can render 3 samples in these scenes. **(a)** Path tracer at 3 spp. **(b)** The result of our method applied to the path graph constructed from 1 spp path tracing has lower variance. **(c-d)** Denoising the results from (a-b) leads to smooth images in both cases, but our RMSE and visual quality are better. **(f)** Shows the reference computed with many samples per pixel. Note that our denoised solution matches the reference color tint closer than the denoised path tracing. Scenes, top to bottom: DINING-ROOM, KITCHEN, STAIRCASE.

Direct illumination. We also show a comparison between our method and path tracing in an indirect dominated scene (OFFICE), a direct-dominated scene (LIVING-ROOM), and an open scene (CAUSTICS) in Fig. 10. In this equal-time comparison, we keep the timing equal by allowing our method to take more direct illumination samples per pixel. Our method is generally more efficient in scenes dominated by indirect lighting, and this comparison shows the transition to where path tracing is more efficient when lighting becomes more direct. Note that even in the LIVING-ROOM scene, which is well lit by direct illumination, our method still shows benefits.

8 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

The main insight of this paper is that the noisy estimates of lighting throughout a scene that are computed during path tracing contain useful information that is lost when we only look at the final radiance estimate for each path. We can squeeze more image quality out of the same data by holding onto a collection of paths and processing them globally to refine our estimates of radiometry in the scene, then computing updated radiance estimates for use in image reconstruction (denoising). The method is orthogonal to different denoising and path guiding methods, and can be combined with them for extra gains. Fundamentally, the size of path graph that can be used is limited by available GPU memory. For preview or interactive applications a single pass of our method can produce a more useful image within a limited path budget, and for offline rendering our method can be used to refine one batch of paths at a time, producing cleaner estimates when they are averaged.

The practical advantage of our method hinges on its scene independence. It competes against simply running the path tracer for the additional time it takes to refine the path graph. Our current implementation does provide practical improvements especially in scenes that are difficult to path-trace, but we believe the benefits will be even greater for even more complex scenes. While our method is light-weight, scene-independent and efficient in complex scenes, it also presents some limitations.

Bias. Like all other path reuse methods, we trade bias for variance. There are two potential sources of bias. First, the aggregation across clusters assumes the incoming radiance along one direction is the same for all the shading points in the cluster. For low sample counts this bias is generally dwarfed by the bias introduced in the denoiser,

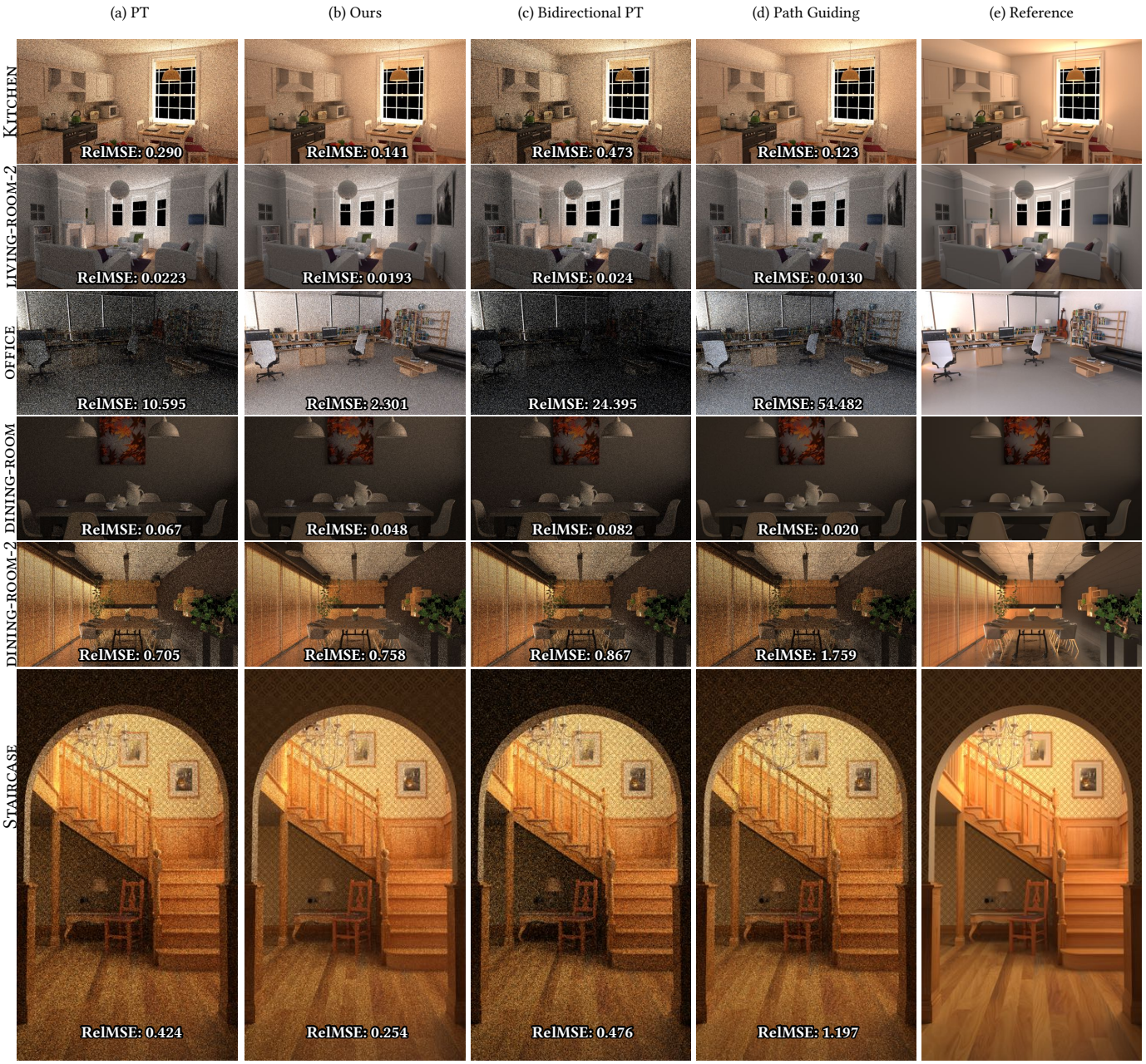

Fig. 9. Comparison between (a) path tracing, (b) our method (path tracing + path graph), (c) bidirectional path tracing and (d) path guiding on an equal time (30sec). We are showing indirect illumination only. The path guiding out performs in most of the scenes but not in extremely difficult scenes. (OFFICE, STAIRCASE), which are interior lit by natural light with one window open. In the direct dominated scenes DINING-ROOM-2, LIVING-ROOM2, KITCHEN the RelMSE value of path guiding and ours are close while ours remain more stable when it comes to indirect dominated scenes.

but it eventually becomes relevant as the result converges. In previous filtering work, the bias due to aggregation could affect e.g. hard shadow edges, but our final gather means this bias only affects the image indirectly. Furthermore, each 1-spp pass of our method has different cluster boundaries, and clusters are small (typically up to 16 points), all of which work against visibility of this bias.

Second, an additional source of bias comes from the clamping that is applied during aggregation when the total outgoing radiance of a cluster is larger than the total incoming radiance. This case is rare, but happens more frequently in scenes with many highly specular surfaces. We use clamping in all results in this paper to ensure that the method is provably convergent. The effect remains

Fig. 10. Equal-time (30s) comparison between the path graph with n indirect samples and m direct samples (top row), and path tracing with m samples (bottom row). The sample count m is determined by the speed of the path tracer, and then n is computed so that the two methods take the same time to run; this provides a comparison with similar quality direct lighting. The path graph is generally more efficient in scenes dominated by indirect lighting, and this comparison shows the transition to where path tracing is more efficient when lighting becomes more direct.

subtle, as shown in Fig. 7: the amount of this bias is comparable or lower than the variance present in a path-traced rendering with thousands of samples (see third column).

Participating media. In this work we assumed vacuum between surfaces. Scenes with participating media are often indirect-dominated and have long light paths, which may be a good opportunity for an extension of our method.

Future work. The path graph framework opens up several opportunities for near-term improvements. In addition to the volume extension, there may be considerable benefit in merging the path graph and denoising phases, so that features deeper in the scene are available when computing reconstruction weights. At the same time it may be useful to replace our deterministic aggregation weights with learned weights computed by a neural network, so path graph refinement and image reconstruction become a unified averaging process that can be tuned through machine learning.

Tuning the implementation can provide further gains, and performance might be further improved by performing selective refinement rather than uniformly refining all points on every iteration. Path graphs could enable fast rendering of scenes that are too large to trace enough paths for acceptable results with denoising alone, and in interactive applications the path graph computation could be pipelined with the path tracing phase, especially in cases where the path graph fits in GPU memory but the scene does not.

REFERENCES

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 97.

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-Domain Path Reusing. *ACM Trans. Graph.* 36, 6, Article 229 (Nov. 2017), 9 pages. https://doi.org/10.1145/3130800.3130886

Philippe Bekaert, Mateu Sbert, and John Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Proceedings of the 13th Eurographics Workshop on Rendering (EGRW '02)*. Eurographics Association, Goslar, DEU, 125–134.

Benedikt Bitterli, Jan Novák, and Wojciech Jarosz. 2015. Portal-Masked Environment Map Sampling. *Computer Graphics Forum (Proceedings of EGSR)* 34, 4 (June 2015). https://doi.org/10.1111/cgf.12674

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10/gg8xc7

Chakravarty R. Alla Chaitanya, Laurent Belcour, Toshiya Hachisuka, Simon Premoze, Jacopo Pantaleoni, and Derek Nowrouzezahrai. 2018. Matrix Bidirectional Path Tracing. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations (SR '18)*. Eurographics Association, Goslar, DEU, 23–32. https://doi.org/10.2312/sre.20181169

Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 98.

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy redistribution path tracing. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1186–1195.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, Philipp Slusallek, and Kavita Bala. 2010. Combining global and local virtual lights for detailed glossy illumination. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–8.

Xi Deng, Shaojie Jiao, Benedikt Bitterli, and Wojciech Jarosz. 2019. Photon Surfaces for Robust, Unbiased Volumetric Density Estimation. *ACM Trans. Graph.* 38, 4, Article 46 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323041

Luca Fascione, Johannes Hanika, Daniel Heckenberg, Christopher Kulla, Marc Droske, and Jorge Schwarzhaupt. 2019. Path Tracing in Production: Part 1: Modern Path Tracing. In *ACM SIGGRAPH 2019 Courses (SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 19, 113 pages. https://doi.org/10.1145/3305366.3328079

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. 1984. Modeling the Interaction of Light between Diffuse Surfaces. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 213–222. https://doi.org/10.1145/964965.808601

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph.* 38, 6, Article 152 (Nov. 2019), 9 pages. https://doi.org/10.1145/3355089.3356515

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 130.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. Graph.* 31, 6, Article 191 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366210

Pat Hanrahan and David Salzman. 1991. A rapid hierarchical radiosity algorithm. In *Computer Graphics*. 197–206.

Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. 2009. Virtual spherical lights for many-light rendering of glossy scenes. In *ACM SIGGRAPH Asia 2009 papers*. 1–6.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix row-column sampling for the many-light problem. In *ACM SIGGRAPH 2007 papers*. 26–es.

Heinrich Hey and Werner Purgathofer. 2002. Importance Sampling with Hemispherical Particle Footprints. In *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG '02)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/584458.584476

Intel. 2020. Intel Open Image Denoise. https://www.openimagedenoise.org.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob, Christian Regg, and Wojciech Jarosz. 2011. Progressive expectation-maximization for hierarchical volumetric photon mapping. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1287–1297.

H. Jensen. 1995. Importance Driven Path Tracing using the Photon Map. In *Rendering Techniques*.

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Rendering Techniques' 96*. Springer, 21–30.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150. https://doi.org/10.1145/15886.15902

Anton S Kaplanyan and Carsten Dachsbacher. 2013. Adaptive progressive photon mapping. *ACM Transactions on Graphics (TOG)* 32, 2 (2013), 16.

Ondřej Karlík, Martin Šik, Petr Vévoda, Tomáš Skřivan, and Jaroslav Křivánek. 2019. MIS Compensation: Optimizing Sampling Techniques in Multiple Importance Sampling. *ACM Trans. Graph.* 38, 6, Article 151 (Nov. 2019), 12 pages. https://doi.org/10.1145/3355089.3356565

Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 49–56.

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2014. Path Space Filtering. In *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. Association for Computing Machinery, New York, NY, USA, Article 68, 1 pages. https://doi.org/10.1145/2614106.2614149

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2016. Path Space Filtering. In *Monte Carlo and Quasi-Monte Carlo Methods*, Ronald Cools and Dirk Nuyens (Eds.). Springer International Publishing, Cham, 423–436.

Claude Knaus and Matthias Zwicker. 2011. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 25.

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Trans. Graph.* 38, 4, Article 37 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323009

Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2008. Radiance Caching for Efficient Global Illumination Computation. In *ACM SIGGRAPH 2008 Classes (SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 75, 19 pages. https://doi.org/10.1145/1401132.1401228

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying Points, Beams, and Paths in Volumetric Light Transport Simulation. *ACM Trans. Graph.* 33, 4, Article 103 (July 2014), 13 pages. https://doi.org/10.1145/2601097.2601219

Eric P Lafortune and Yves Willems. 1993. Bi-directional path tracing. In *Compugraphics' 93*. 145–153.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proceedings of EGSR)* 36, 4 (June 2017), 91–100. https://doi.org/10.1111/cgf.13227

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: Matrix Slice Sampling for the Many-Lights Problem. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–8. https://doi.org/10.1145/2070781.2024213

Mark Pauly, Thomas Kollig, and Alexander Keller. 2000. Metropolis light transport for participating media. In *Rendering Techniques 2000*. Springer, 11–22.

Hao Qin, Xin Sun, Qiming Hou, Baining Guo, and Kun Zhou. 2015. Unbiased Photon Gathering for Light Transport Simulation. *ACM Trans. Graph.* 34, 6, Article 208 (Oct. 2015), 14 pages. https://doi.org/10.1145/2816795.2818119

Lars Schjøth, Jon Sporring, and O Fogh Olsen. 2008. Diffusion based photon mapping. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 2114–2127.

Benjamin Segovia, Jean Claude Iehl, Richard Mitanchey, and Bernard Péroche. 2006. Bidirectional Instant Radiosity.. In *Rendering Techniques*. 389–397.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford University.

Eric Veach and Leonidas Guibas. 1995a. Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques (Proceedings of the Fifth EUROGRAPHICS Workshop on Rendering)*, Georgios Sakas, Stefan Müller, and Peter Shirley (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 145–167.

Eric Veach and Leonidas J Guibas. 1995b. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 419–428.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33, 4 (aug 2014).

Bruce Walter, Adam Arbree, Kavita Bala, and Donald P Greenberg. 2006. Multidimensional lightcuts. In *ACM SIGGRAPH 2006 Papers*. 1081–1088.

Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional Lightcuts. *ACM Trans. Graph.* 31, 4, Article 59 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185555

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *Proceedings of EGSR 2007*.

Beibei Wang, Jing Huang, Bert Buchholz, Xiangxu Meng, and Tamy Boubekeur. 2013. Factorized Point-Based Global Illumination. *Computer Graphics Forum (Special Issue on EGSR 2013)* 32, 4 (2013), 117–123.

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. 1988. A Ray Tracing Solution for Diffuse Interreflection. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88)*. Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/54852.378490

Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. 2020. Continuous Multiple Importance Sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392436

Shilin Zhu, Zexiang Xu, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. 2020. Deep Kernel Density Estimation for Photon Mapping. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 35–45.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 667–681.