



# Refined tri-directional path tracing with generated light portal

Xuchen Wei<sup>1</sup> · GuiYang Pu<sup>2</sup> · Yuchi Huo<sup>1</sup> · Hujun Bao<sup>1</sup> · Rui Wang<sup>1</sup>

Accepted: 4 May 2024 / Published online: 25 June 2024

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2024

## Abstract

The rendering efficiency of Monte Carlo path tracing often depends on the ease of path construction. For scenes with particularly complex visibility, e.g. where the camera and light sources are placed in separate rooms connected by narrow doorways or windows, it is difficult to construct valid paths using traditional path tracing algorithms such as unidirectional path tracing or bidirectional path tracing. Light portal is a class of methods that assist in sampling direct light paths based on prior knowledge of the scene. It usually requires additional manual editing and labelling by the artist or renderer user. Tri-directional path tracing is a sophisticated path tracing algorithm that combines bidirectional path tracing and light portals sampling, but the original work lacks sufficient analysis to demonstrate its effectiveness. In this paper, we propose an automatic light portal generation algorithm based on spatial radiosity analysis that mitigates the cost of manual operations for complex scenes. We also further analyse and improve the light portal-based tri-directional path tracing rendering algorithm, giving a detailed analysis of path construction strategies, algorithm complexity, and the unbiasedness of the Monte Carlo estimation. The experimental results show that our algorithm can accurately locate the light portals with low computational cost and effectively improve the rendering performance of complex scenes.

**Keywords** Rendering · Monte Carlo path tracing · Light portal

## 1 Introduction

The core issue of Monte Carlo path tracing is how to efficiently explore the path space defined by the rendering equation [16]. In some rendering perspectives of complex scenes, the probability of constructing a complete path from the camera to the light source is low due to the occlusion between the geometric surface and the light source, and the convergence speed of the Monte Carlo algorithm is slow. In the past decades, many researchers have invested a lot of effort in developing advanced methods to improve the performance of Monte Carlo algorithms. Methods based on path sampling and connection, including bidirectional path tracing (BDPT) [18, 34], photon mapping [12, 14] and vertex connection and merging [8] construct a complete path through random walks of camera and light sub-paths and connecting vertices between sub-paths. Metropolis light

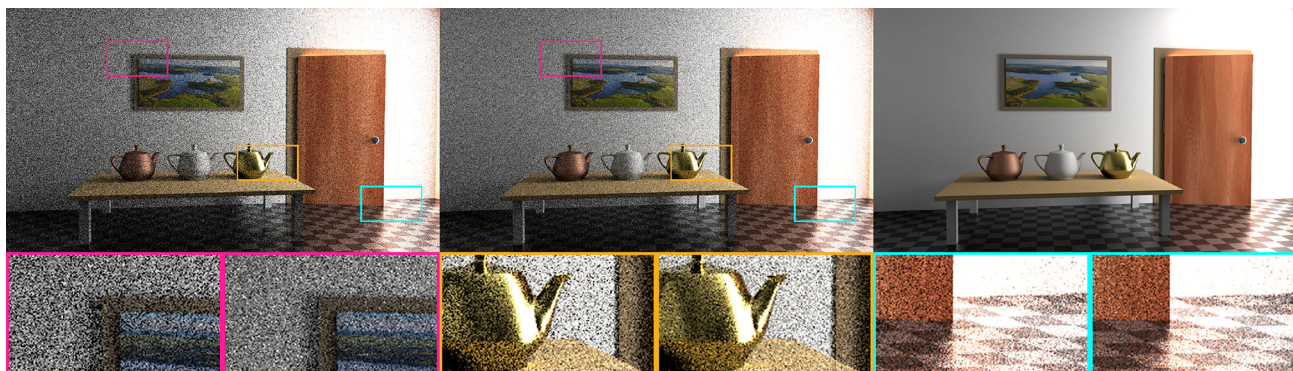
transport (MLT) [35] is a Markov chain Monte Carlo method which explores the path space based on vertex perturbation and neighbourhood searching of history path [13, 21]. Path guiding algorithms [23, 29, 31] appear to be practical solutions that iteratively update the spatial-directional radiance field based on valid path samples, continuously optimizing the sampling space for the next path. Among the above methods, many complex algorithms have specific application scenarios, and except for unidirectional path tracing, few algorithms are guaranteed to be universally applicable and require extra time-space overhead. Today, with the increased performance of GPU hardware and the rise of neural network applications, low-sample rendering and denoising techniques based on modern ray tracing pipelines [17] require higher-quality, faster-generated path samples. While some works have already achieved impressive progress [19, 24], the graphics pipeline still needs concise and memory-friendly implementations.

To solve the problem of difficult sampling and constructing effective light paths in complex scenes, several solutions have been proposed by researchers in recent years. Light portals are a technique to support light sampling in indoor or semi-open scenes. Some algorithms use light portals to

✉ Rui Wang  
ruiwang@zju.edu.cn

<sup>1</sup> State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China

<sup>2</sup> China Mobile (Hangzhou) Information Technology Co., Ltd, Hangzhou, China



**Fig. 1** 32 spp comparisons between BDPT(left) and our refined TDPT(middle) using an auto-generated light portal. (The 720p images were rendered on GPU, the maximum path depth was 4, and the rendering time is 4 s and 5 s, respectively. The ground truth image (right) was rendered by BDPT using 7.7k spp in 340 s.) Veach-ajar (door) scene presents difficult visibility due to the emitter and camera being placed in two separated rooms. Most indirect paths through the door slit contribute to the rendering result. BDPT suffers from insufficient path samples because the probability of successful connections between pre-

fix vertices on sub-paths is relatively low. Our refined TDPT algorithm is unbiased and dramatically increases the likelihood of constructing a valid path by sampling a portal segment and connecting their endpoints to sub-paths, resulting in more abundant samples and less image noise. Our experiments have shown that TDPT produced lower MAPE (maximum absolute percentage error) (56.7% to 67.1%) and higher PSNR (peak signal to noise ratio) (16.9 to 14.5), indicating the better applicability and robustness for such scenarios

support direct light sampling, including environment map oriented window portal projections [6] and Metropolis perturbation of portal paths [27]. There has also been some works on representing and generating light portals based on analysis of scene geometry, including light portal shape representations [25] and generalized light portals [26]. Recently, Rath et al. [30] proposed the focal path guiding algorithm, which is the first technique to combine path guiding and spatial analysis to augment path sampling in small regions.

Inspired by the above works, we consider the automated generation of light portals as an interesting problem, while extending the application of light portals to more general rendering tasks. Tri-directional path tracing (TDPT), proposed by Anderson et al. [3] in their domain-specific language (DSL), is an ideal method for light portals. TDPT is the successor to BDPT that increases the likelihood of obtaining paths through the small aperture by sampling an intermediate path segment. However, the algorithm is described only as an example case of their DSL and lacks implementation details, especially the light portal configurations. In this paper, we try to illustrate the process of constructing tri-directional paths based on light portals, and no longer regard TDPT as a subsequent process of BDPT by removing the samples from bidirectional connection. So that the different path tracing algorithms do not obtain repeated path samples, which helps us to compare the results more effectively and intuitively (Fig. 1). In summary, the main contributions of this article are threefold:

1. A refined TDPT algorithm implementation and in-depth analysis.

2. A novel workflow for the automated synthesis of light portals.
3. A combined theoretical framework for representing, generating and Monte Carlo rendering light portals.

## 2 Related works

### 2.1 Light portal

The light portal is a classic concept used early in computer graphics by Jones to remove hidden lines [15]. The portal was first introduced by Airey [1] and has been actively used in the context of occlusion culling for rasterization [2, 20, 22, 32]. These approaches focus on determining the primary visibility seen through the portal, given a viewpoint, such as a part of the room seen through a door. Light portals are now commonly adopted in major production renderers such as Arnold [4] and RenderMan [28]. Especially for indoor and semi-outdoor scenes, light portals can directly guide the next event estimation (NEE) through small openings such as doors and windows. Bitterli et al. [6] presented a technique for importance sampling distant, all-frequency environment maps in indoor scenes using portals with precise parameterization. Otsu et al. [27] first combined Metropolis light transport with pre-known light portals to perturb the path through the portal. We also find some works on automatic analysis, representation and generation of light portals. Nguyen [25] proposed an automated way to analyse a given 3D scene and create a specific portal shape. It requires light sources to emit rays directly towards the portal regions, which is diffi-

cult to accomplish in scenes with complex visibility. Shinji [26] introduced a new concept that allows existing polygon meshes with arbitrary shaders in the scene to be used as generalized light portals. We think that previous works have certain limitations. Firstly, different methods lack a unified representation and definition of light portals. Second, it is necessary to invest a large computational effort to generate light portals beyond the scene original geometry. Lastly, it would not fundamentally overcome the obstacle of path construction based on sampling and connection in rendering applications. Therefore, it is a challenging task to combine light portals with Monte Carlo path tracing and find a universal way to represent and generate optimal light portals for different scenes.

### 2.2 Tri-directional path tracing

Anderson et al. [3] designed Aether to simplify the sophisticated implementation of Monte Carlo rendering. They first proposed the pseudo code for tri-directional path tracing to demonstrate the flexibility and scalability of their DSL. The algorithm first performs BDPT and then samples a 2-vertex ray segment from a light portal. The tri-directional connection implies connecting one vertex of the portal segment to the vertices of the camera sub-path and the other to the light sub-path. Their paper gives a comparison of rendering results using the same 4 spp. However, we find the result inadequate because the TDPT algorithm continues to overlay new paths after the BDPT algorithm has finished. Since the Monte Carlo accumulative rendering converges more easily on pixel samples combined from more path samples, we cannot simply conclude that the TDPT algorithm using Aether’s description is necessarily better than BDPT in terms of performance. Furthermore, they did not show the configuration details of the light portals in their experiments, which motivates us to validate our results using generated light portals. In the following sections, we will clarify the details and give a refined implementation of the TDPT algorithm, including portal sampling, tri-connections and multiple importance sampling (MIS) calculations.

### 2.3 Focal path guiding

Focal effects are common in life, but are usually difficult to handle in rendering algorithms due to the stochastic nature of Monte Carlo path tracing. Rath et al. [30] constructed some tricky scenes to illustrate the phenomenon of focal effects, e.g. a hole is dug in a wall and an area light source is placed behind it, emitting rays towards the hole. If a path finally reach the light source, it must pass through the hole where the focal point is located. Combined with the idea of path-guiding, sampling rays towards the hole must be able to increase the probability of hitting the light source, so they extended the traditional path guiding cache from surface

local to scene global, and proposed a new focal path guiding algorithm based on spatial hierarchy. In principle, both focal effects and light portals attempt to produce similar light transport patterns. However, in practice, once the focal size exceeds a certain range, path guiding would be ineffective. In other words, a larger focal is still a light portal. We therefore believe that light portal based path construction addresses more general problems, and that light portals allow using relatively simple geometric representations without having to maintain spatial hierarchies at runtime.

$$I_{\text{light}} = L_e^{(0)}(y_0, y_1) \prod_{i=0}^{s-2} f_r(y_{i-1}, y_i, y_{i+1}) G(y_i, y_{i+1}) \tag{1}$$

$$I_{\text{portal}} = \underbrace{f_r(y_{s-2}, y_{s-1}, x_l) G(y_{s-1}, x_l)}_{C1} \underbrace{f_r(y_{s-1}, x_l, x_c) G(x_l, x_c) f_r(x_l, x_c, z_{t-1})}_{C2} \underbrace{G(x_c, z_{t-1}) f_r(x_c, z_{t-1}, z_{t-2})}_{C3} \tag{2}$$

$$I_{\text{camera}} = W_e^{(0)}(z_0, z_1) \prod_{i=0}^{t-2} f_r(z_{i-1}, z_i, z_{i+1}) G(z_i, z_{i+1}) \tag{3}$$

## 3 Refined tri-directional path tracing

In our work, we define the light portal as a planar polygon to simplify intersection and storage. Furthermore, the plane can divide the scene space into two regions, and the normal of the plane can statistically represent the average direction of the rays passing through the light portal. (We require that along the direction of light transport from the light source to the camera, the ray passing through the light portal is always located on the upper-hemisphere of the normal vector.) In the following subsections, we present our implementation details and refinements based on the original TDPT algorithm. First, our refined algorithm can be executed independently without bidirectional links, which significantly reduces the computational cost of ray tracing and visibility testing. Second, after introducing portal sampling and intersection, we properly handle multiple important sampling weight calculations, ensuring unbiasedness and robustness.

### 3.1 Tri-directional path construction

Following the surface form of the light transport equation, we express our path measurement contribution function as a product of three parts (Eqs. 1,2,3): a light sub-path  $\bar{y}$  with  $s - 1$  vertices ( $s \geq 0$ ), a ray segment through light portal with 2 vertices  $x_l$  and  $x_c$  ( $x_l$  locates near light source and  $x_c$  locates near camera) and a camera sub-path  $\bar{z}$  with  $t - 1$  vertices ( $t \geq 1$ ) (Fig. 2).

Specifically,  $I_{\text{portal}}$  is a product of the contribution from three connecting edges:

- C1: Edge between prefix vertex from a light sub-path and a light portal vertex.
- C2: Edge between 2 light portal vertices.
- C3: Edge between prefix vertex from a camera sub-path and a light portal vertex.

In addition, there are some boundary situations of Eqs. (1,2,3) that may affect the calculation of certain terms:

- $s = 0$ : The portal edge has a vertex on the light source.
- $t = 1$ : The portal edge has a vertex that can be directly projected to camera film.

We will explain them below in combination with specific path construction strategies.

### 3.1.1 Portal sampling

We first sample a point  $x_p$  on the surface of portal mesh, with a probability  $p_A(x_p)$  in area measurement and sample an outgoing direction  $\omega$  according to cosine distribution  $p_\Omega(\omega)$  in solid angle measurement. Then we trace the ray twice along  $\omega$  and  $-\omega$  and may get two closest intersection points  $x_l$  and  $x_c$ . The sampled portal segment is valid if and only if  $x_l$  and  $x_c$  are both on the surface. We formulate a joint probability density  $p_{AA}(x_l, x_c)$  to represent the sampling distribution of a portal segment:

$$p_{AA}(x_l, x_c) = \frac{p_A(x_p)p_\Omega(\omega)}{\cos(\theta_p)} G(x_l \leftrightarrow x_c) \tag{4}$$

where  $\theta_p$  is the angle between the surface normal  $N(x_p)$  and  $\omega$ ,  $G$  is geometry term of the portal segment. Additionally, we assume that  $x_c$  would never hit a light source and ignore the paths with depth=1. If the point  $x_l$  hits the light directly, it means that the prefix vertex of the light sub-path is not used, corresponding to the case  $I_{\text{light}}(s = 0)$ . If the point  $x_c$  can be projected to camera on a valid pixel, it means that we only need to connect the camera vertex  $z_0$ , corresponding to the case  $I_{\text{camera}}(t = 1)$ .

### 3.1.2 Sub-paths Generation

Starting from the origin position  $z_0$  of a camera and a randomly sampled vertex  $y_0$  on a light source, the random walk process of generating the camera and light sub-paths is trivial. For the camera sub-path, we need to trace no more than depth  $k$  because camera sub-path may finish at the light source as a valid path, and we apply NEE to vertices from  $z_1$  to  $z_{k-1}$  for direct illumination. We only need to store the first  $k - 1$

**Table 1** Computational costs of different stages

Stage	Cost
$\bar{y}$	$(k - 1)c_I + (k - 1)c_V$
$\bar{z}$	$kc_I + (k - 1)c_V$
Bi-connection	$\frac{(k-1)(k-2)}{2}c_V$
Tri-connection	$2c_I + (2k - 3)c_V + (5k - 7)c_P$

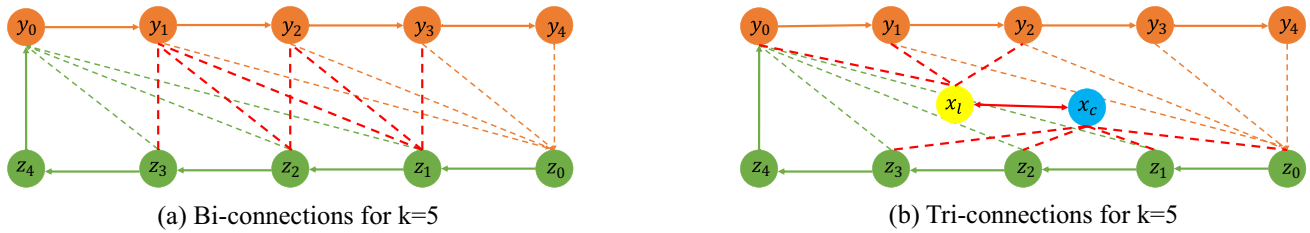
vertices as camera sub-path  $\bar{z} = (z_0z_1 \cdots z_{k-2}), k \geq 2$ , since the vertex  $z_k$  and the  $z_{k-1}$  do not participate in the following connections. Similarly, for the light sub-path, we trace paths no more than depth  $k - 1$  and connect vertices  $y_1$  to  $y_{k-2}$  with camera as samples splatted on the light image. We only store the first  $k - 2$  vertices as light sub-path  $\bar{y} = (y_0y_1 \cdots y_{k-3}), k \geq 3$ . During the tracing process, we need to record each intersection if a ray segment happens to pass the portal, which is essential for MIS calculation.

### 3.2 Tri-directional connections cost

We summarize the relationship between the path depth  $k$  and the rendering cost of different tri-directional path construction stages in Table. 1, where  $c_I$  is the cost of the closest intersection,  $c_V$  is the cost of the visibility test, and  $c_P$  is the cost of the ray-portal intersection test. Given that TDPT and BDPT share the same sub-path samples, the additional cost of tri-connection compared to bi-connection is two closest intersection detections (for tracing the two endpoints of the portal segment) and a few ray-portal intersection detections. In addition, tri-connection has linear complexity in visibility detection, whereas bi-connection has quadratic complexity. Due to the planar triangle mesh representation, the cost of ray-portal intersection is almost negligible, and since the light portal is a virtual geometry outside the scene hierarchies, the complexity of ray-scene intersection does not increase. Thus, the cost of path construction with independent TDPT is almost the same as with BDPT. Based on this, we perform the rendering test with the same number of samples at the same time, and then we can better distinguish the difference in performance between the two algorithms.

### 3.3 Unbiased estimator with MIS

Multiple-importance sampling is a reliable and efficient approach for combining different sampling strategies of Monte Carlo integral estimation. If a path sample  $\bar{x}$  is constructed using strategy  $a_i, i \in [1, N]$ , the calculation formula of the corresponding integral estimator is Eq. (5). Using balance heuristics for example, the MIS weight function can be written as Eq. (6). For BDPT, the strategy space of a  $k$ -length path is all bi-connection cases  $(s, t)$ , where  $k = s + t - 1$ .



**Fig. 2** The difference between BDPT and our refined TDPT. To construct a  $k=5$  path, tri-connections increase 1 visibility tests (red dashed line) and increase 2 intersection tests ( $x_l \leftrightarrow x_c$ ) compared to bi-

connections. With almost the same computational cost, TDPT greatly improves the probability of a connection between the camera sub-path and the light source sub-path

And the pdf of a path  $\bar{x}_{s,t}$  is Eq. (7), where the direction of arrow denotes the propagation direction of path from light to camera.

$$\langle I(\bar{x}) \rangle = w_{a_i}(\bar{x}) \frac{I(\bar{x})}{p_{a_i}(\bar{x})} \tag{5}$$

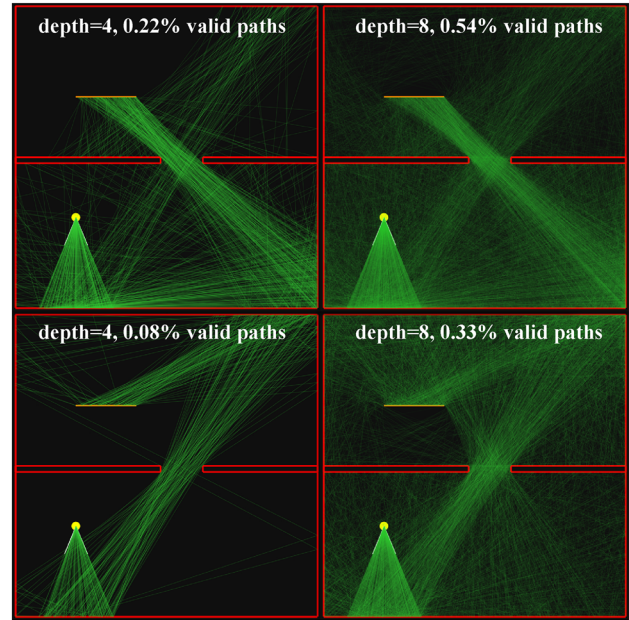
$$w_{a_i}(\bar{x}) = \frac{p_{a_i}(\bar{x})}{\sum_{j=1}^N p_{a_j}(\bar{x})} \tag{6}$$

$$p(\bar{x}_{s,t}) = \prod_{i=0}^{s-1} \vec{P}_A(y_i) \prod_{i=0}^{t-1} \overleftarrow{P}_A(z_i) \tag{7}$$

By combining the above formulas, we are able to derive an elegant calculation of the weight function of the BDPT MIS using an efficient technique [33]. However, the situation of TDPT is obviously different. One reason is that there must be a gap between the portal shape and the scene, so it is impossible to separate the scene into two ideal completely enclosed spaces, resulting in some path samples without any segment of ray passing through the portal, but still contributing to the rendering result. Another reason is that for a complete light path through the portal, only one or a few segments of rays can pass through the portal, making the entire strategy space discrete. For applying MIS to TDPT, we need to take special care to enumerate the different path construction probabilities as follows:

- $p_{camera}(\bar{x})$ :  $x$  is constructed by PT when generating the camera sub-path (The BDPT pdf is  $p(\bar{x}_{0,k+1})$ ).
- $p_{direct}(\bar{x})$ :  $x$  is constructed by NEE when generating the camera sub-path (The BDPT pdf is  $p(\bar{x}_{1,k})$ ).
- $p_{light}(\bar{x})$ :  $x$  is on the light image when generating the light sub-path (The BDPT pdf is  $p(\bar{x}_{k,1})$ ).
- $p_{portal}(\bar{x})$ :  $x$  is constructed by tri-connection. Specifically, for a portal segment  $x_l \leftrightarrow x_c$  connect a light sub-path with  $s$  vertices and a camera sub-path with  $t$  vertices, the pdf is Eq. (8).

$$p_{portal}(\bar{x}_{s,x_l,x_c,t}) = \prod_{i=0}^{s-2} \vec{P}_A(y_i) p_{AA}(x_l, x_c) \prod_{i=0}^{t-2} \overleftarrow{P}_A(z_i) \tag{8}$$



**Fig. 3** We designed a 2D scene for path tracing simulation by placing the area light(orange line segment) and the camera (yellow point) in separate rooms, with the diffuse reflecting walls(red line segments). We calculate the ratio of valid paths out of 32768 primary rays for different emitter directions and visualized the distribution of rays in space (green line segments). We see that the probability of constructing an effective path is very low in scenes with complex visibility. And the distribution of all effective paths in space is characterized by certain features. We consider the region with higher rays density to be the potential light portal region

Our refined TDPT algorithm exploits all possibilities in the strategy space and ensures unbiased estimators.

### 4 Generated light portal

We propose a novel workflow for the automatic generation of light portals based on the analysis of light transport in the scene. Our core idea is that the light portal must be located in some critical spaces in the scene, which are unavoidable for most light paths to pass through. As shown in Fig. 3,

we were inspired to find the appropriate location of the light portal by analysing the distribution of rays in some more complex scenes. The rays passing through more frequent spatial regions mean that the probability of sampling and constructing valid paths in these regions is higher. Our automatic generation method finds the surface pairs that would enclose the portal space and then finds the most likely location for the portal to be placed.

In Sect. 4.1, we construct a directed connectivity graph based on the shape factors of the scene and find the surface pairs by solving the minimum cut problem. Next, we sample and trace multiple rays between each surface pair and deposit the visible rays in a uniform grid, as described in Sect. 4.2 of the light grid. We use a practical method to analyse and fit a planar polygon (represented as a triangular mesh) from the light grid as a light portal in Sect. 4.3.

### 4.1 Form factor network flow

According to the network flow problem [9], we can construct a digraph  $G = (V, E)$ , where  $V$  represents the nodes in the graph,  $E$  represents the directed edges connecting the nodes, and each edge has a capacity value. There are two particular nodes in the graph, where the source node  $S$  with  $\text{deg}^-(S) = 0$  has only outgoing edges, and the sink node  $T$  with  $\text{deg}^+(T) = 0$  has only incoming edges. We believe that the light transport process in the scene is very similar to this digraph model. According to the Light Path Expressions (LPEs) [11], a light path starting from a light source, bouncing between the surfaces of the scene and finally ending at the camera can be expressed by  $L \cdot * C$ . Therefore we consider the light source and the camera as source and sink nodes in the network flow, respectively, and construct a digraph by calculating the form factor [10] according to Eq. (9) as edge capacity. Form factors are usually used to describe the fraction of diffusely reflected light that leaves one surface and reaches another. In our work, we did not need to subdivide the scene into tiny polygons, we just used Monte Carlo sampling to calculate the form factor between the original surface pairs of the scene. As shown in Eq. (10), we uniformly sample  $N$  points on surface  $A_i$  and  $M$  points on surface  $A_j$  according to the surface area.

$$F_{j \rightarrow i} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} dA_j dA_i \quad (9)$$

$$\langle F_{j \rightarrow i} \rangle = \frac{A_j}{NM} \sum_{i=1}^N \sum_{j=1}^M \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} \quad (10)$$

*Find primary and direct triangles set* Given a scene with specific light sources and camera configurations, we can find

a primary triangle set  $S_p$ , which consists of all the triangles directly visible to the sink node(camera), and a direct triangle set  $S_d$ , which consists of all the triangles directly illuminated by a light source. In our experiments, all light sources are area light with single-sided emission. Note that even if there are multiple light sources in the scene, there is only one direct triangle set. Hence we don't need to distinguish which triangles are illuminated by which lights.

*Build form factor graph* Once we finish constructing  $S_p$  and  $S_d$ , we first check if they have common triangles( $S_p \cap S_d \neq \Phi$ ). Suppose a triangle is directly illuminated by a light source and can be projected onto the current camera viewport. In that case, it means that there must exist a direct light path(the length is 2), and the graph has already connected. Otherwise, to build a connected graph, we need to further check relationships between triangle pairs( $t_i, t_j \in (S_d, S_p)$ ). To calculate the capacity of each edge, we need to distinguish between three cases with different situations:

1. For each  $t_j \in S_p$ , connect from a triangle to camera
2. For each light source, for each  $t_i \in S_d$ , connect from a light to a triangle
3. For each pair  $(t_i, t_j) \in (S_d, S_p)$ , connect from a triangle to another triangle

In most cases, we can construct indirect edges with a high probability even if there is no direct light path in the current field of view. We believe that the above strategy has two advantages, one is to simplify the form factor calculation that should be performed on all triangle pairs in the scene to two subsets of triangle pairs, which greatly reduces the computational cost, and the other is that our method is robust to the placement of scene light sources.

*Solve Minimum Cut* Our goal is to perform the minimum cut algorithm on the constructed digraph in order to solve the min-cut edges. All the nodes in the figure are split into two disjoint sets, and the minimum cut can get the maximum amount of flow through the edges connecting the two sets. Then, we can collect the triangles on the source side in the *Source Triangles Set* and the triangles on the sink side in the *Sink Triangles Set*. We see them as *Pivot Triangles Set* on critical light paths where the light portal could be placed in between. And these pairs of triangles act as bridges for visible ray samples. This step can also be thought of as a filter for  $S_d$  and  $S_p$ , reducing unnecessary triangles and easing the computational burden of subsequent steps. To further simplify the complexity, we can remove some smaller triangles in advance and sample rays according to area for efficient form factor calculation. Experiments in Section 5 show that proper simplification does not affect the stability of the results, but significantly accelerates the process of calculating pivot triangles.

## 4.2 Construct a light grid

We divide the scene into a uniform grid with a reasonable resolution and label empty voxels as *Air* by inserting the scene into the grid. For each pair of triangles in the *Pivot Triangle Set*, we sample rays between them and conduct ray marching through *Air* voxels. We maintain a counter in each voxel and increase by one as soon as a visible ray passes. After the counting process, we assume the light portal might be placed at the *Air* voxels with the most counter values. To obtain more accurate and stable counting results, we increase the number of ray samples iteratively and perform visibility tests and ray marching on GPU. With the help of compute shader, atomic addition operations can be performed on a 3D image.

## 4.3 Fit a light portal mesh

The remaining question is how to utilize the results of the light grid and generate a light portal with a proper shape for TDPT rendering. Based on our observations, we believe that a reasonable way to represent a light portal is to use a planar triangular mesh. The advantage is that it is convenient for rendering programmes to use a consistent geometric representation for ray tracing. It is also efficient for sampling ray segments on the surface of the light portal in a relatively straightforward way. Specifically, our method first performs a principal component analysis (PCA) on all air voxels locations where count values are above a certain threshold. We take the main axis direction of the voxels as the normal of the light portal plane. Next, we get the equation of the plane where the most counted voxel lies and project all voxels that intersect this plane onto it. Finally, we compute the convex hull of the projected points and triangulate to get the final light portal geometry. The light portal generated by the above process has several characteristics: (a) it must lie in the air space of the scene so that it does not overlap the original surfaces in the scene. (b) Compared to the manually placed light portal, it must be located where more rays can pass through, which is an optimal solution. Our auto-generated light portals can exhibit better robustness and practicality in the TDPT algorithm.

## 5 Implementations and experiments

We implemented our algorithms on a machine containing a core i7-9700K processor and an Nvidia GeForce 2080Ti GPU with 11GB of video memory. Solving the minimum cut [7], as well as the PCA, Delaunay triangulation, and convex hull algorithms used in the portal shape generation process, are all done using the APIs in Matlab R2020a. Our prototype

rendering programme of TDPT is implemented in C++ with Vulkan GPU ray tracing [17] for simplicity and efficiency.

*Scene configurations* We grabbed some scene models from [5] and edited them appropriately for portal experiments. For the *Cornell-box*, we dug a hole in the right wall and connected to a new box of the same size through this hole, moving the original light source into the new box. For the *Veach-ajar* scene, we reverse the emitting direction of the light source (Our rendering algorithms all use a single-sided diffuse area emitter) of the original scene. For the *Bathroom2* scene, similar to the *Cornell-box*, we removed the original direct-emitting light source placed on the window, built a channel through the window, and moved the light source to the new room connected.

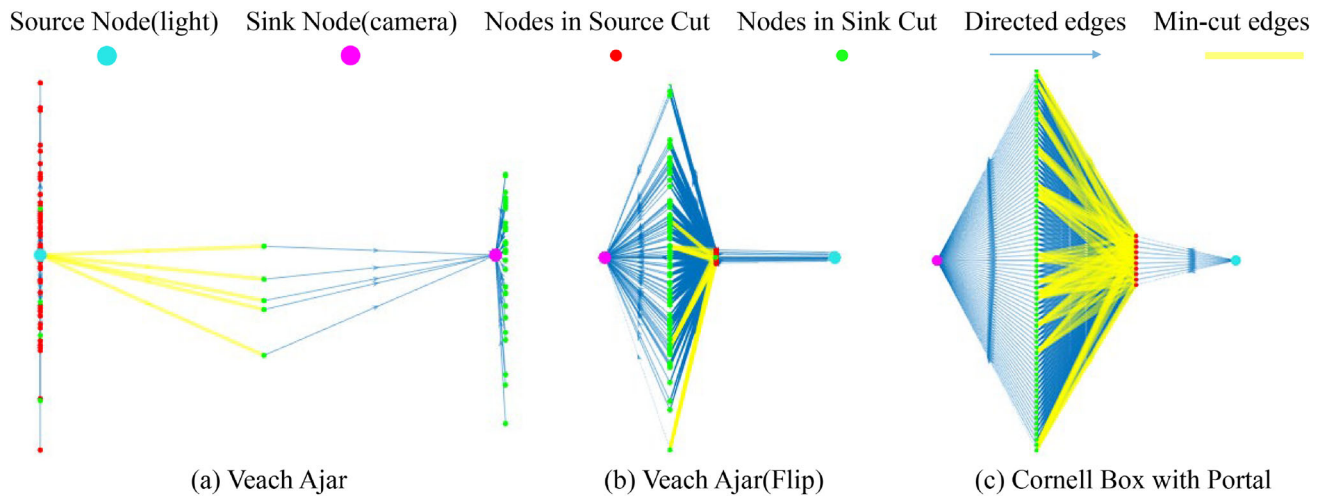
*Portal Generation* As shown in Fig. 4a, *Veach-ajar* has direct illumination, so the connected relationship of the digraph is very simple. The light source belongs to the source set, and the five triangles receiving direct illumination belong to the sink set. Note that there are many triangles in the graph that have become orphan nodes, and they will not participate in the calculation of portal generation. The case of indirect illumination could be complicated, as shown in Fig. 4b, c. The result of solving the minimum cut is in line with our expectations because all the connecting edge endpoints belong to different sets. Although the form factors between each pair of triangles need to be calculated, after solving the min-cut, only the connected triangles at both ends of the min-cut are retained for the light grid, which greatly reduces the amount of calculation.

*TDPT Rendering* We rendered our test scenes with different Monte Carlo path construction strategies. To facilitate an equitable evaluation of renderers, we follow the pseudocode of Aether [3] and replicate the original algorithm within our renderer as TDPT-ORG, while our refined version is TDPT-REF. We recorded the render time per frame for each algorithm and computed the mean value and the overall percentage of nonzero pixels of each frame as samples accumulated. We also evaluate the mean absolute percentage error (MAPE) and peak signal-to-noise ratio (PSNR) to compare the image quality.

### 5.1 1-SPP performance

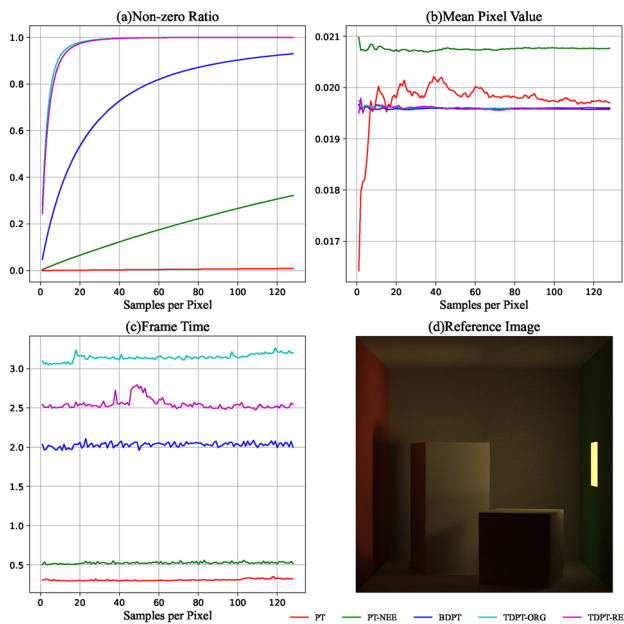
In Monte Carlo path tracing rendering, the percentage of nonzero pixel values among total pixels indicates the efficiency of various path construction strategies, the average value of RGB channel sums indicates the expectance of different integrators, and rendering time per frame reveals the calculation complexity. All contribute to evaluating rendering performance.

As shown in Fig. 5, the basic approaches (PT/PT-NEE) proved inefficient due to the limited construction strategy. Although BDPT explores the path space and shows a faster



**Fig. 4** Min-cut results of the test scenes. We visualized the digraph by colouring the nodes and highlighting the connecting edges between the cut sets. The test scenes are divided into two categories according to the orientation of the light sources. **a** contains direct light paths. **b** and **c** contain only indirect light paths. Our method can handle both cases and find suitable triangle pairs to generate light portals. For the first type,

the main energy transfer is between the light source and the triangles directly illuminated by the light source, and thus the number of nodes is small. For the second type of scenario, it is impractical to compute all pairs of triangles as there are no duplicate elements between the main set of triangles and the direct set of triangles. The minimum cut algorithm helps to filter the data set to find high impact candidates



**Fig. 5** Statistics of Cornell-box at depth 6 with 128 spp. **a** The starting value of each curve is the nonzero value percentage of the 1-spp image. In this scene, the percentage is 0.00619%(PT), 0.382%(NEE), 4.79%(BDPT), 27.6%(TDPT-ORG) and 24.4%(TDPT-REF), respectively. The curve's growth rate reflects the algorithm's efficiency. **b** The mean value of the five algorithms can converge to roughly the same result about 0.02 at 128spp, indicating that the implementations of all rendering algorithms are correct and unbiased. **c** The computation time per frame shows that our refined algorithm reduces the unnecessary bi-connections overhead and improves the performance by about 20%(3.1 to 2.5 s) compared with the original algorithm

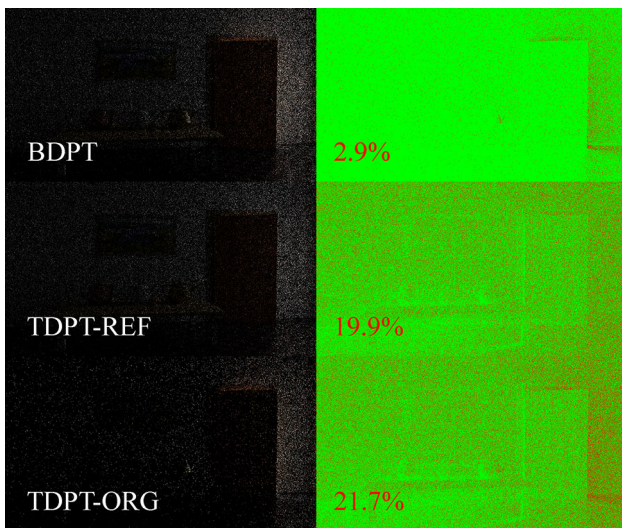
convergence rate from the experimental results, the proportion of nonzero pixels in each frame is still low, indicating a high probability of bidirectional connection failure in this kind of scene. In contrast, TDPT shows great advantages in such scenes. In the case that TDPT-ORG and TDPT-REF show similar trends in the nonzero pixel rate, our refined algorithm proves the effectiveness of using tri-connections with less computational cost.

We further illustrate the significance of the light portal sampling and tri-connections by visualizing the 1-spp image of *Veach-ajar(Flip)*, as depicted in Fig. 6. The presence of red pixels indicates the establishment of a valid path. Since the original TDPT algorithm relies on BDPT samples, it becomes evident that the concentration of red pixels around the door frame. From the statistical results, our refined algorithm has significantly improved the sampling efficiency, and the sampling area and range of the original algorithm are highly matched.

## 5.2 Image quality comparisons

Since the rendering scenes used in this article are quite complex, it is important to use appropriate error evaluation metrics. Although the TDPT path is easily acquired, its performance is limited by the availability of effective policies, and in extreme cases, TDPT's policies may still fail, resulting in the degradation of the algorithm to naive path tracing. At the same time, changes in multiple importance sampling (MIS) weights may result in different pixels performing differently in different areas of the rendered image. To measure





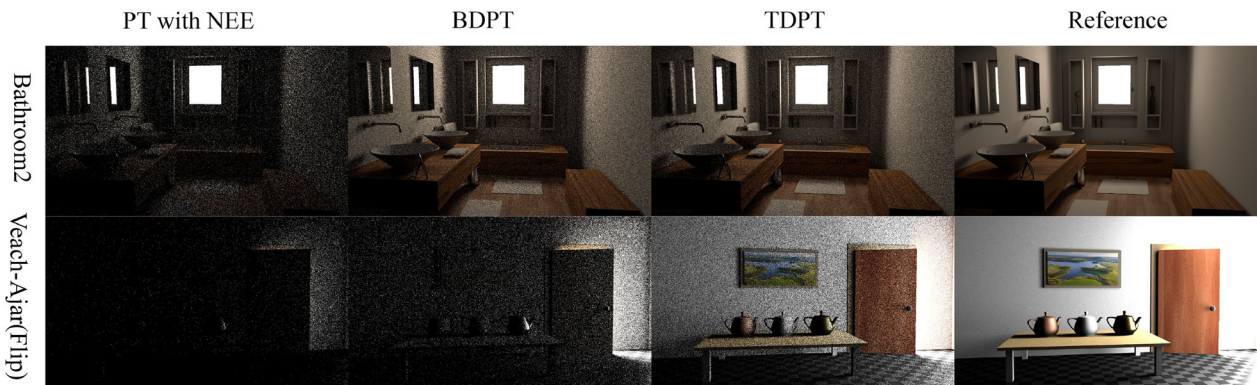
**Fig. 6** Visualizations of nonzero pixel values of *Veach-ajar(Flip)*(1-spp). The three algorithms achieved the same mean value (0.044). Since there is no direct light in this scene, it is hard to obtain valid path samples in a large number of room surfaces using BDPT. By marking zero-value pixels in green, we can visually observe the distribution of effective paths. TDPT provides access to light for locations that would otherwise be unsampled

the image noise level, we use PSNR, while MAPE is also used to quantify the deviation between the rendered result and the reference image. According to the experiments in Fig. 7, our algorithm achieves the expected performance.

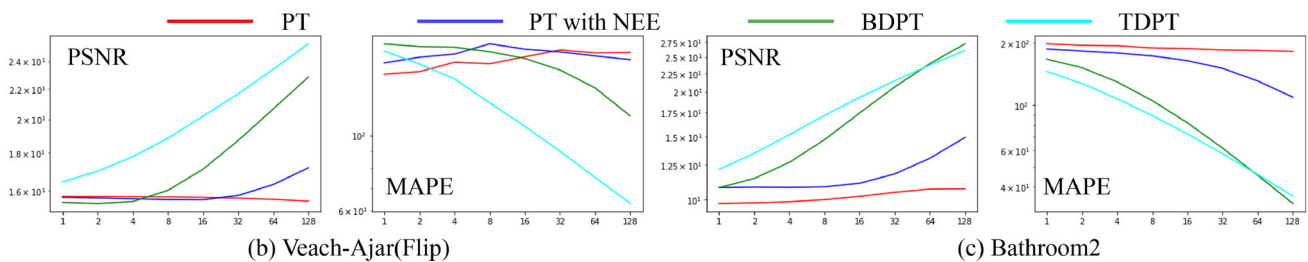
### 6 Conclusions

Our work proves that the automatic generation of portals is effective and can solve the problem of efficiently rendering complex scenes with indirect lighting. Our experiments with graph network flow have shown that we can perform statistical analysis of light distribution in space at a low cost. We have also implemented a refined, unbiased, and efficient TDPT algorithm, effectively reducing the computation cost of redundant path connections in the original work. In the future, the expression form of the portal and the path construction strategy will continue to inspire us to explore more general and concise algorithms.

*Complicate sampling distribution* We have used uniform and cosine distributions over the upper hemisphere for directional sampling and uniform distribution based on the portal area for positional sampling as in Eq. (4). However, when building the light grid, we only store the number of rays passing through each voxel. Suppose we still have enough



(a) 16 Spp Rendering Results with Maximum Depth of 5



**Fig. 7** We present some rendering results of different scenes and the image error metrics of PSNR and MAPE. Once we flip the *Veach-ajar* scene’s light source, the camera cannot receive any direct lighting. Hence, the metrics of TDPT occupy an overwhelming advantage. But

in the *Bathroom2* scene, the original window area is large enough that the bi-connection strategy still has a high probability of successful connection. Hence, the advantage of TDPT is not apparent in comparison

computing power to calculate the average direction of the rays in the voxel. In this case, we can determine the more accurate normal direction of the portal and use the directional distribution obtained at this stage to guide the portal ray sampling. At the same time, if we also consider the counts of other locations on the portal surface, calculating a normalized area distribution will also make position sampling more efficient.

**Diverse portal representation** We believe that portals represented by planar triangular meshes can avoid some unnecessary problems. For example, a planar shape can guarantee the uniqueness of the intersection points. In contrast, a 3D shape may result in a ray having more than one intersection with the portal surface, which affects the choice of construction strategy and the calculation of path probability. For this reason, we laboriously convert the most counted voxels into a planar shape. It is a promising work to combine focal path guiding and light portals and to use the spatial hierarchy structures for portal generation.

**Various connection strategies** Our algorithm does not cause the two endpoints of the portal segment to continue randomly outwards. It will be a more challenging path construction problem to include sampling two portal sub-paths and connecting their prefix vertices to the light and camera sub-paths. In the future, we can explore the guiding role of the portal in constructing two sub-paths in this type of scene.

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1007/s00371-024-03464-6>.

**Acknowledgements** The work was partially supported by National Key R&D Program of China (No. 2023YFF0905102), Key R&D Program of Zhejiang Province (No. 2023C01039).

**Author Contributions** Xuchen Wei wrote the main manuscript. All authors reviewed the manuscript.

**Data availability** No datasets were generated or analysed during the current study.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

## References

- Airey, J.M.: Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations. Ph.D. thesis, The University of North Carolina at Chapel Hill (1990)
- Aliaga, D.G., Lastra, A.A.: Architectural walkthroughs using portal textures. In: Proceedings. Visualization'97 (Cat. No. 97CB36155), pp. 355–362. IEEE (1997)
- Anderson, L., Li, T.M., Lehtinen, J., Durand, F.: Aether: an embedded domain specific sampling language for monte carlo rendering. *ACM Trans. Graph.* (2017). <https://doi.org/10.1145/3072959.3073704>
- Autodesk, Inc: Arnold node reference-lights-light portal. <https://help.autodesk.com/view/ARNOL/ENU/> (2024)
- Bitterli, B.: Rendering resources (2016). <https://benedikt-bitterli.me/resources/>
- Bitterli, B., Novák, J., Jarosz, W.: Portal-masked environment map sampling. *Computer Graphics Forum (Proceedings of EGSR)* **34**(4) (2015). 10/f7mbx7
- Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(9), 1124–1137 (2004). <https://doi.org/10.1109/TPAMI.2004.60>
- Georgiev, I., Krivánek, J., Davidovič, T., Slusallek, P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* **31**(6), 1921–19210 (2012). <https://doi.org/10.1145/2366145.2366211>
- Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28–30, 1986, Berkeley, California, USA (1986)
- Goral, C.M., Torrance, K.E., Greenberg, D.P., Battaile, B.: Modeling the interaction of light between diffuse surfaces. In: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '84, p. 213–222. Association for Computing Machinery, New York, NY, USA (1984). <https://doi.org/10.1145/800031.808601>
- Gritz, L., Stein, C., Kulla, C., Conty, A.: Open shading language. In: ACM SIGGRAPH 2010 Talks, SIGGRAPH '10. Association for Computing Machinery, New York, NY, USA (2010). <https://doi.org/10.1145/1837026.1837070>
- Hachisuka, T., Ogaki, S., Jensen, H.W.: Progressive photon mapping. *ACM Trans. Graph.* (2008). <https://doi.org/10.1145/1409060.1409083>
- Jakob, W., Marschner, S.: Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.* (2012). <https://doi.org/10.1145/2185520.2185554>
- Jensen, H.W.: Realistic Image Synthesis Using Photon Mapping. A. K. Peters Ltd, USA (2001)
- Jones, C.B.: A new approach to the 'hidden line' problem. *Comput. J.* **14**(3), 232–237 (1971). <https://doi.org/10.1093/comjnl/14.3.232>
- Kajiya, J.T.: The rendering equation. *SIGGRAPH Comput. Graph.* **20**(4), 143–150 (1986). <https://doi.org/10.1145/15886.15902>
- Khronos: Ray tracing in vulkan. <https://www.khronos.org/blog/ray-tracing-in-vulkan> (2020)
- Lafortune, E.P., Willems, Y.D.: Bi-directional path tracing (1993)
- Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C., Wyman, C.: Generalized resampled importance sampling: foundations of restrir. *ACM Transact. Gr.* **41**, 1–23 (2022). <https://doi.org/10.1145/3528223.3530158>
- Lowe, N., Datta, A.: A technique for rendering complex portals. *IEEE Trans. Visual Comput. Gr.* **11**(1), 81–90 (2005)
- Luan, F., Zhao, S., Bala, K., Gkioulekas, I.: Langevin monte carlo rendering with gradient-based adaptation. *ACM Trans. Graph.* **39**(4) (2020)
- Luebke, D., Georges, C.: Portals and mirrors: simple, fast evaluation of potentially visible sets. In: Proceedings of the 1995 Symposium on Interactive 3D Graphics, pp. 105–ff (1995)
- Müller, T., Gross, M., Novák, J.: Practical path guiding for efficient light-transport simulation. *Comput. Gr. Forum* **36**(4), 91–100 (2017). <https://doi.org/10.1111/cgf.13227>
- Müller, T., Rousselle, F., Novák, J., Keller, A.: Real-time neural radiance caching for path tracing. *ACM Trans. Graph.* (2021). <https://doi.org/10.1145/3450626.3459812>

25. Nguyen, P.: Light portals: light transport variance reduction. *Dissertations & Theses - Gradworks* (2014)
26. Ogaki, S.: Generalized light portals. *Proc. ACM Comput. Graph. Interact. Tech.* (2020). <https://doi.org/10.1145/3406176>
27. Otsu, H., Hanika, J., Dachsbacher, C.: Portal-based path perturbation for metropolis light transport. In: *Proceedings of Vision, Modeling and Visualization* (2020)
28. Pixar Animation Studios: Renderman 24 documentation-pxrportalLight. <https://rmanwiki.pixar.com/display/REN24/PxrPortalLight> (2021)
29. Rath, A., Grittmann, P., Herholz, S., Vévoda, P., Slusallek, P., Krivánek, J.: Variance-aware path guiding. *ACM Trans. Graph.* (2020). <https://doi.org/10.1145/3386569.3392441>
30. Rath, A., Yazici, O., Slusallek, P.: Focal path guiding for light transport simulation. *Association for Computing Machinery*, New York, NY, USA (2023). <https://doi.org/10.1145/3588432.3591543>
31. Ruppert, L., Herholz, S., Lensch, H.P.A.: Robust fitting of parallax-aware mixtures for path guiding. *ACM Trans. Graph.* (2020). <https://doi.org/10.1145/3386569.3392421>
32. Teller, S.J., Séquin, C.H.: Visibility preprocessing for interactive walkthroughs. *ACM SIGGRAPH Comput. Gr.* **25**(4), 61–70 (1991)
33. Veach, E.: Robust monte carlo methods for light transport simulation. Ph.D. thesis, Stanford, CA, USA (1998). AAI9837162
34. Veach, E., Guibas, L.: Bidirectional estimators for light transport. In: Sakas, G., Müller, S., Shirley, P. (eds.) *Photorealistic Rendering Techniques*, pp. 145–167. Springer, Berlin Heidelberg, Berlin, Heidelberg (1995)
35. Veach, E., Guibas, L.J.: *Metropolis Light Transport*, 1 edn. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3596711.3596744>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



**Xuchen Wei** is a PhD candidate at the State Key Lab of CAD&CG, Zhejiang University. He received his bachelor's degree in digital media from Zhejiang University in 2018. His research interests are in Monte Carlo rendering and physically based ray tracing.



**GuiYang Pu** is a Senior Engineer and Manager at China Mobile (Hangzhou) Information Technology Co., Ltd. He received a bachelor's degree and a PhD degree from Zhejiang University. His research interests are mainly in rendering, VR/AR, digital twins and service computing. Now, he leads a group working on transferring emerging rendering and AI techniques into products and services.



**Yuchi Huo** is a “Hundred Talent Program” researcher in the State Key Lab of CAD&CG, Zhejiang University. His research interests are in rendering, deep learning, image processing, and computational optics, which are aiming for the realization of next-generation neural rendering pipeline and physical neural computation.



**Hujun Bao** is a professor at the State Key Lab of CAD & CG and the College of Computer Science and Technology, Zhejiang University. He leads the 3D graphics computing group in the lab, which mainly makes researches on geometry computing, 3D visual computing, real-time rendering, and their applications. His research goal is to investigate the fundamental theories and algorithms to achieve good visual perception for interactive digital environments, and develop related systems.



**Rui Wang** is a professor at the State Key Lab of CAD&CG, Zhejiang University. He received a bachelor's degree in Computer Science and a PhD degree in Mathematics from Zhejiang University. His research interests are mainly in real-time rendering, realistic rendering, GPU-based computation, 3D display techniques, and AI-based rendering. Now, he leads a group working on the next-generation real-time rendering engine and techniques.