Volume xx (200y), Number z, pp. 1–15

Neural Path Sampling for Rendering Pure Specular Light Transport

Rui Yu[1,2] t-ruiyu@microsoft.com
Yue Dong[2†] yuedong@microsoft.com
Youkang Kong[3,2] t-ykong@microsoft.com
Xin Tong[2] xtong@microsoft.com

[1]University of Science and Technology of China [2]Microsoft Research Asia [3]Tsinghua University

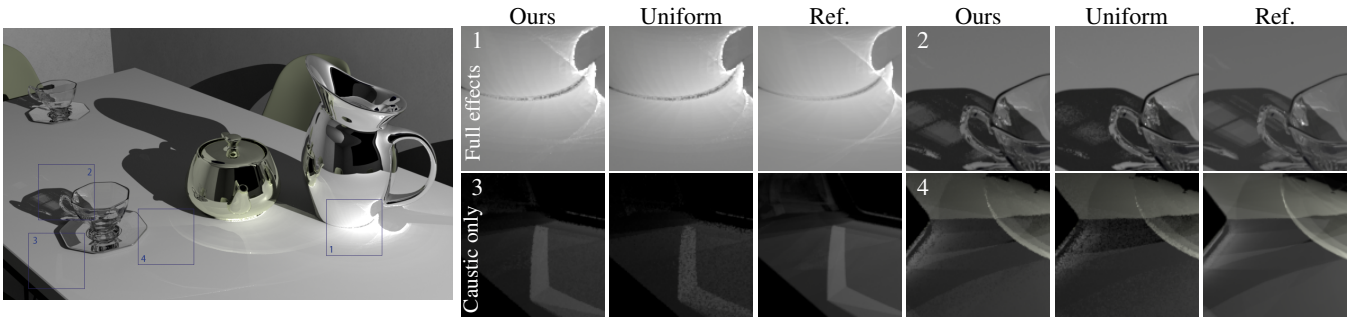

Figure 1: *Rendering of* THE BREAKFAST ROOM *scene that presented challenging pure-specular light transport paths. We tested a biased specular manifold sampling (SMS) [ZGJ20] estimator with uniform initial path sampling and our neural path sampling, all with a rendering time of 2.2 hours. Ground-truth images are included as references. To better visualize the results, we properly scaled the exposures of the cropped region. The bottom row shows the energy contributed by the specular paths only. Our neural path sampling method yielded convincing results, while the original biased SMS estimator exhibited energy loss and produced noisy images.*

Abstract

Multi-bounce, pure specular light paths produce complex lighting effects, such as caustics and sparkle highlights, which are challenging to render due to their sparse and diverse nature. We introduce a learning-based method for the efficient rendering of pure specular light transport. The key idea is training a neural network to model the distribution of all specular light paths between pairs of endpoints for one specular object. To achieve this, for each object, our method models the distribution of sparse and diverse specular light paths between two endpoints using smooth 2D maps of ray directions from one endpoint and represents these maps with a 2D convolutional network. We design a training scheme to efficiently sample specular light paths from the scene and train the network. Once trained, our method predicts specular light paths for a given pair of endpoints using the network and employs root-finding-based algorithms for rendering the specular light transport. Experimental results demonstrate that our method generates high-quality results, supports dynamic lighting and moving objects within the scene, and significantly enhances the rendering speed of existing techniques.

CCS Concepts

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

1. Introduction

Specular or transparent objects in a scene illuminated by point light sources, such as the sun or LED lights, can generate intricate glint highlights and caustics through multiple pure-specular light bounces. These effects are either caused by paths containing only specular vertices (glints) or by a single diffuse vertex surrounded by specular chains (caustics). Accurately simulating specular light transport is essential for numerous realistic rendering tasks. However, this remains a challenging task, as tracing the sparse and diverse specular light paths contributing to the rendering results is difficult using conventional path tracing methods [LW93, VG95].

Photon Mapping [Jen96] effectively simulates caustic effects by casting photons along specular light paths originating from the light source and converging on diffuse surfaces. A large number of photons can be sampled progressively [HJ09], such method can also be integrated with other bidirectional rendering algorithms [HPJ12, GKDS12]. However, due to their reliance on spatial co-

† Corresponding author

herency of light transport, these methods struggle with geometries with high-frequency visibility changes. Moreover, they fail to accurately simulate glint highlights through specular light paths directly connecting the light source and the viewpoint.

The specular light paths between nearby endpoints lie on a manifold, based on this key observation, a set of root-finding methods [JM12, HDF15, HKD15, KHD14, ZGJ20, JC22] has been developed to find a valid pure specular path between two endpoints via optimization from an initial light path. Although these methods greatly improve the efficiency to trace one specific light path, they either assume there is only one specular light path between two endpoints [HDF15] or rely on uniform inital path sampling [JM12, HKD15, KHD14, ZGJ20] to find all specular light paths between two endpoints. However, they either miss critical light paths and introduce bias, or a large number of initial samples are required to find all valid paths. Recently, Wang et. al. [WHY20] presented an efficient algorithm to find all specular light paths between two endpoints in a scene by searching the full combination of triangles filtered by a conservative cut. However, the performance of the method decreases significantly as the number of bounces increases.

Despite various approaches for estimating the energy of glints or caustics, a central challenge remains: identifying all valid paths with pure specular vertices connecting two given endpoints. These endpoints can be the camera and light source (glint) or a sample on the caustic receiver and the light source (caustics). Thanks to the root-finding optimization, we do not need a precise path, but a good initialization would significantly accelerate the entire root-finding-based rendering process.

To address this challenge, we introduce a learning-based method for representing the pure specular light transport of specular objects within a 3D scene. By training a scene-specific neural network that predicts the distribution of all specular paths between two endpoints, we can generate more efficient initial paths that are closer to the valid path, thus accelerating the root-finding-based rendering [ZGJ20, JC22].

Designing an efficient neural network-based solution presents several technical challenges. First, the specular light paths between two endpoints are sparse, and the number of specular light paths between different endpoint pairs varies. It is unclear how to encode these sparse and variable-length light paths using existing network structures designed for modeling fixed-length vectors or 2D signals over regular grids. Second, the neural network should be compact to minimize the computational cost of network inference during rendering. Simultaneously, the network should accurately model the distribution of specular light paths for any input endpoint pairs, reducing the computational cost of the root-finding algorithm. Finally, generating all specular light paths between two endpoints for network training is prohibitively expensive. Therefore, we need to design an efficient data generation and network training scheme to ensure that our rendering time speedup is not diminished by costly preprocessing.

To tackle these challenges, we encode the distribution of specular light paths between two endpoints with 2D maps and represent the 2D maps for all endpoint pairs in the scene with a 2D convolutional neural network (CNN). Our key observation is that no matter how diverse and complex the specular light paths between two endpoints are, each of them can be uniquely determined by the direction of its first ray segment from the starting point. (Similar to SMS [ZGJ20] refraction and reflection are treated as separated cases to avoid path splitting) As a result, all the valid specular light paths between two endpoints can be represented by a set of 2D positions (the UV-coordinate of the first hit point or the ray direction). Specifically, we encode the distribution by a distance map, which measures the 2D distance to the nearest 2D positions of valid paths. However, the distance map struggles to distinguish between multiple valid paths that are in close proximity, and it cannot indicate the number of valid paths locally. To address these limitations, we incorporate a density map and assign a fixed-sized Gaussian kernel to each valid path. With the 3D positions of the endpoints as input, the CNN generates both density and distance maps that represent the valid path distribution.

This 2D map representation provides several advantages. First, the 2D distribution map can model an arbitrary number of light paths. Second, by modeling the path distribution as the sum of 2D Gaussian functions and the distance map, we can efficiently encode those smooth 2D maps by a compact CNN and derive the sparse specular paths from them. Finally, since the value of 2D maps in a local region is only determined by nearby specular paths but not all of them, we can use patches of the 2D maps but not the whole ones for network training thus avoiding the requirement of finding all specular paths for each pair of endpoints. To this end, we design an efficient data generation and network training scheme, where the network can be trained from a collection of valid specular paths randomly sampled in path space, without including any complete set of valid paths connecting two endpoints.

Our training scheme leverages the fact that sampling a valid path without specifying the endpoints is much simpler than finding valid paths connecting given endpoints. Our method takes advantage of the coherency among all valid paths across different light and view configurations, instead of only learning from paths (on-line) sampled from the current frame, as used by most guided sampling methods [VKv*14, MGN17]. This approach results in a much more efficient training scheme.

Our method can be integrated with existing biased and unbiased root-finding algorithms for simulating specular light transport. After training, our method can support rendering with dynamic light and views, as well as moving objects in the scene. We evaluate our method with a set of scenes with complex specular light transport effects and compare our method with existing solutions. Experiments show that our method can efficiently encode the specular path distributions and generate convincing caustic and sparkle highlight effects for all scenes.

In summary, the contributions of our work can be described as:

- A learning-based method for efficiently simulating pure specular light transport of a scene;
- A 2D representation for modeling the distribution of all specular paths between two endpoints and a 2D CNN for encoding the distributions of specular paths of all endpoint pairs;
- An efficient data generation and network training scheme for learning the network from a collection of specular light paths with different endpoints.

2. Related Work

Generic Monte Carlo rendering Path tracing [Kaj86] serves as the foundation for a series of Monte Carlo rendering algorithms. Bidirectional path tracing [LW93, VG95] was introduced to improve sampling efficiency for difficult paths. However, sampling some pure-specular light transports, particularly when multiple valid paths connect two given endpoints, remains challenging and results in significant variance. In practice, the pure-specular interaction is relaxed by slightly increasing roughness and employing small area light sources. The specular paths can also be selectively mollified and sampled at the cost of introducing bias [KD13, WDH*21].

Photon-mapping-based approaches Methods based on photon mapping [Jen96] are effective at rendering caustics by gathering photons in a local region, which results in a spatial blur. The progressive photon mapping framework [HJ09] allows for tracing a large number of photons progressively, leading to a consistently converging result. This scheme can also be integrated with other bidirectional methods [HPJ12, GKDS12] and used in production [CK20]. To determine the optimal bandwidth for photon gathering, CPPM [LLZ*20] introduced a statistical test, improving both rendering time and result quality.

Although photon mapping is efficient for rendering certain caustics, it depends on the caustic receiver gathering a sufficient concentration of photons. When the caustic receiver is very thin and experiences high-frequency occlusion changes, photon casting becomes inefficient since the photons have a low probability of hitting the target. This results in an insufficient number of photons in a local region and noisy results. Furthermore, photon mapping-based methods cannot handle glint highlights when the specular path terminates at the camera vertex instead of on a diffuse surface. Our method focuses on finding valid paths connecting two endpoints, rather than relying on any light path or photon in spatially local regions. Consequently, our approach not only supports rendering glint highlights but also offers advantages when rendering caustics cast on objects with thin geometric features.

Root-finding-based approaches Mitchell and Hanrahan [MH92] introduced the Newton method to find valid paths based on Fermat's principle. To extend this to shading normals, Walter *et al.* [WZHB09] proposed optimization on the angular space by aligning the shading normal and half-vector. Loubet *et al.* [LZHJ20] proposed a sampling method that can connect two endpoints in space with exactly one specular interaction on a particular triangle. A global exploration with a hierarchical pruning scheme is also proposed for efficient sampling over all triangles. To find all valid paths connecting two given endpoints, PathCut [WHY20] performs optimization on the full combination of triangles filtered by a conservative cut. The efficiency of the path cut strategy decreases with multiple bounces and long paths due to the increased search space.

For Markov chain Monte Carlo (MCMC) [Vea98] rendering, Manifold exploration [JM12] is introduced for proposing mutations of existing pure-specular paths based on a method exploring the differential geometry of the manifold of specular paths. Kaplanyan *et al.* [KHD14] improved the constraint by introducing optimization in the half-vector space and later extended it to more difficult light paths with multiple glossy interactions [HKD15]. All those methods are designed for local perturbation based on one existing valid path. Finding the initial path by exploration in larger spaces still relies on classical sampling.

Manifold next event estimation (MNEE) [HDF15] applies root-finding optimization in a general Monte Carlo context. Starting from one deterministic initial admissible path, MNEE follows deterministic optimization based on iterative manifold explorations and projections. A similar vertex connection scheme is later applied in bi-directional path tracing [SHVJ18]. Due to the deterministic initial path generation, at most one admissible path can be found between two given endpoints. Thus, these methods fail to converge on complex light transport where multiple valid connections exist.

Specular Manifold Sampling (SMS) [ZGJ20] introduces a general scheme to simulate pure specular light transport within the Monte Carlo rendering framework, supporting both biased and unbiased estimation. Without proper prior knowledge, uniform random sampling is conducted over the specular surface to establish an initial set of paths. For accurate estimation, a large number of initial paths are required; otherwise, the results may be noisy (for the unbiased estimator) or experience energy loss (for the biased estimator). A two-stage sampling approach is proposed, as well as the use of LEAN mapping [OB10] to identify improved initial paths, though only for normal-mapped objects. Advanced root-finding optimization methods are also proposed [JC22].

Orthogonal to root-finding rendering algorithms that focus on designing estimators, our approach addresses the challenge of efficiently sampling initial paths. It can be integrated into existing root-finding-based rendering techniques, thereby improving their rendering efficiency.

Radiance caching and path guiding With existing sampled valid paths or photons from the current scene, radiance caching can speed up rendering by approximating the radiance in the scene. Early works only approximate reflections from diffuse surfaces [WRC88] and later extended to glossy surfaces [KGPB05] or participating media [JDZJ08, MJJG18]. Recently, Muller *et al.* [MRNK21] proposed an online training scheme for radiance caching, enabling dynamic scene support.

Path guiding methods learn the radiance distribution from existing samples rendered for the scene. Early works guided new samples based on stored rays or photon maps [LW95, Jen95]. The radiance can be better modeled by the Gaussian Mixture Model (GMM) [VKv*14, SHJD22], or by a spatial-directional tree [MGN17]. Reibold *et al.* [RHJD18] followed a similar progressive guiding and learning scheme as [MGN17], while building a GMM based on sampled high variance paths. Recently, Ruppert *et al.* [RHL20] proposed a parallax-aware representation which further improves path guiding efficiency. Deep learning models are also used for path guiding [HWZ*20, ZXS*21a, ZXS*21b] to learn the best path guidance from sampled photons or rays.

Both radiance caching and path guiding algorithms have difficulty dealing with high-frequency effects introduced by pure specular light transport. Difficult paths are sparsely distributed and need to be sampled in the first place before the caching or guiding steps. To address this, Li *et al.* [LWT*22] recently introduced a path guiding scheme based on representative specular paths found by a global solver [WHY20]. Such a guiding scheme is proven effi-

cient on rendering caustics cast by objects with non-zero roughness. However, for pure specular objects, the representative paths already include all the valid paths in the scene, thus no further guided sampling is needed.

Path guiding and radiance cache are designed for rendering single-frames in static scenes with fixed lighting and viewpoints. Due to the sparse nature of pure specular paths, once established, their energy is known, and no nearby paths will be valid due to the impulse nature of specular reflection. This makes path-guiding based on current frame samples ineffective without special considerations.

Concurrent to our work, two methods [XWW23, FHG*23] have been proposed for leveraging existing valid pure specular paths for efficient rendering. Xu *et al.* [XWW23] collect and reuse both spatial and temporal valid paths for initialization and also help to determine a spatially varying number of trials. Fan *et al.* [FHG*23] proposed a systematic approach by reconstructing continuous energy distributions from historical and coherent sub-paths in an online training fashion. The method begins with an initialization stage, sampling seed paths, followed by an online training stage that reconstructs the distribution. This distribution is then used for importance sampling discrete specular chains in regular Monte Carlo rendering.

Unlike those methods that leverage the coherency for paths within the current rendered frame or in the temporally nearby frames, our method learns all valid paths connecting endpoints in a large predefined region. Relaxing the endpoints not only introduces much more coherency but also leads to efficient training path sampling that covers more complex, difficult paths. Our method better avoids online training pitfalls where critical valid paths go missing from the initialization sampling phase by adopting an offline training scheme. The extra offline training time cost can be alleviated when rendering multiple frames since the training only needs to be performed once.

3. Method Overview

Given a 3D scene that consists of specular or transparent objects, for each object, our method takes two stages to learn and render its specular light transport in the scene (shown in Figure. 2).

In the offline training stage, we first determine a 3D valid region of endpoints that encloses all specular light paths based on the scene layouts and a predefined range of views, lights, and object movements used in the rendering. Without loss of generality, we assume that a pure specular light path starts from a caustic receiver (non-pure-specular objects) surface point or the viewpoint (for sparkle highlight) and ends at a point light source (or a point sample of light sources). Thus the valid region of endpoints is determined by the possible position of the light source and the viewpoint during the rendering and the position of caustic receivers surrounding the specular objects. After that, our method samples a collection of specular light paths between endpoints in the 3D region and trains the network with the sampled light paths.

In the rendering stage, given the light and view positions, we first perform conventional path tracing to determine all pairs of endpoints in the 3D region between them where the specular light paths

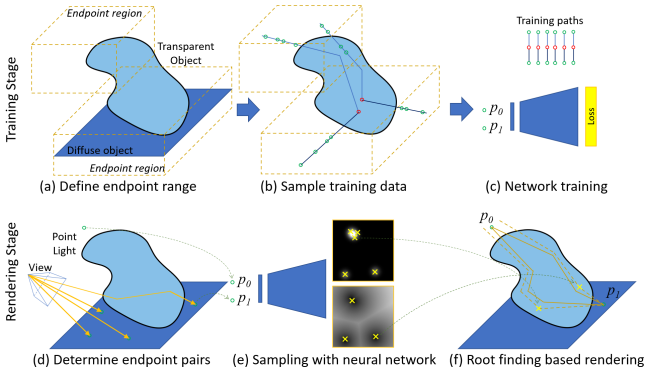

Figure 2: *Overview. During the training stage, (a) we first determine the required region of endpoints (orange dashed lines) based on scene layout, view, and lights. (b) We then randomly sample specular paths starting from the specular surface point (red circle) to generate training data of the endpoints (green circle) and first ray segment (dark blue line) (c) and train the network. During rendering, (d) a regular path tracing pass is rendered to determine all endpoint pairs (green circle) where the contribution of all specular paths between them needs to be evaluated. (e) For each pair of endpoints, we take their 3D position as the neural network input and infer the distribution of valid specular paths encoded in 2D maps and sample initial paths (yellow 'x') based on the 2D maps. (f) Finally, we perform root-finding methods for each initial path (orange dash) to find the valid path (orange line) and evaluate its contribution.*

will contribute to the rendered images. For each pair of endpoints, we query the network to get the distribution of specular light paths and then sample the initial paths based on the distribution map for guidance. Finally, we perform root-finding for each initial light path to obtain the exact specular light paths between queried endpoint pairs. We repeat this process for all pairs of endpoints to obtain the specular light transport.

In the following, we first describe the 2D representation of light path distributions and the technical details of the rendering stage in Section 4. After that, we introduce our data generation and network training scheme in Section 5. Finally, we validate our method and compare it to existing works in Section 6.

4. Neural Path Sampling

In this section, we first describe the 2D representation for modeling the distribution of specular light paths between two endpoints. After that, we describe how to generate the initial path from the inferred distribution maps and compute the specular light path with the unbiased and biased root-finding algorithms respectively.

4.1. Representation

We represent all valid paths for a given set of endpoints p_0 and p_1 by two maps defined on a 2D parameterization of the specular surface. Thanks to the fully deterministic nature of pure specular paths, one pure specular path connecting the two given endpoints

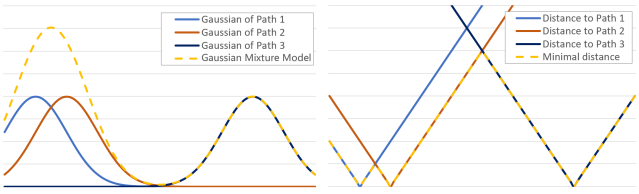

Figure 3: *1D illustration of our Gaussian and distance map representation. For the density map (left), each valid path is represented by a fixed-sized Gaussian function centered at the valid path position. The density map is the sum of all individual Gaussian functions and effectively represents the number of valid paths in a local region. Path 1 and 2, which are closer, can be represented by a higher value, indicating multiple paths in this local region. However, the density map cannot encode the accurate position of the paths, especially when multiple paths are in close proximity. On the other hand, the distance map excels at encoding the position of each path, as the minimum distance map is always zero at the valid path position. However, the distance map struggles to encode the number of valid paths, particularly when the valid paths are near each other.*

p_0, p_1 can be solely determined by the first ray segment that hit the pure specular surface. As a result, all valid paths are corresponding to a set of 2D points defined on the pure specular surface.

To represent those sparse 2D points, we parametrize the specular surface as a 2D UV-domain and use a density and a distance map to represent those 2D points. For each valid path, we assign a fixed-sized normalized Gaussian function centered at its path position and form a density map. This density map approximates the probability of having a nearby valid path, in addition, since each Gaussian function is normalized, the integral of the density represents the total number of valid paths for this pair of endpoints. The distance map encodes the 2D distance from the current position to the nearest valid path position. Specifically, the density map $\mathbf{K}(\mathbf{x})$ and the distance map $\mathbf{D}(\mathbf{x})$ are defined as:

$$\mathbf{K}(\mathbf{x}) = \sum_{\mathbf{v}_i \in \mathbb{P}} G(\mathbf{x}, \mathbf{v}_i) \tag{1}$$

$$\mathbf{D}(\mathbf{x}) = \min_{\mathbf{v}_i \in \mathbb{P}} \|\mathbf{x} - \mathbf{v}_i\|_2 \tag{2}$$

Where $\mathbb{P}$ is the set of all valid paths with each path represented by the 2D position of the first hit point $\mathbf{v}_i$; and $G(\mathbf{x}, \mathbf{v}_i)$ is a Gaussian-kernel function centered at $\mathbf{v}_i$.

Figure 3 illustrates one example of the density and distance map representation. Although in theory, either map is sufficient to encode all the valid paths effectively, in practice, we find the density map can better encode the probability and the distance map is good at pinpointing the accurate location of a valid path. As illustrated in the 1D example, since the Gaussian has a smooth peak and is mixed using *sum*, nearby Gaussian peaks will be merged into one higher peak that indicates more valid paths locally. The higher peak, however, makes it difficult to set one fixed threshold to find all peaks. On the other hand, the distance is mixed using *min*, allowing a fixed threshold to find all the minimums, pinpointing the locations of the valid paths regardless of how many valid paths exist locally. Combining both maps results in the best tradeoff for the proposed algorithm.

The full representation is 8-dimensional, consisting of two endpoints in 3D space and a 2D map as output. Although it is possible to fit both the distance and density maps with classical data structures like high-dimensional trees or Gaussian Mixture Models, the high-dimensional nature makes the fitting unstable. Additionally, preparing the complete map for fitting is costly (see Section 5 for further discussion), so we only have a partial evaluation of the map. As a result, a flexible fitting scheme that supports arbitrary loss functions is preferable. Finally, when generating initial path samples, we require fully evaluated distance and Gaussian maps, which means that the representation should efficiently output a complete 2D slice rather than evaluating a single point. A convolutional neural network is ideal for producing 2D maps that utilize the spatial coherence of the 2D output. As a result, we choose to employ a 2D convolutional neural network to represent both maps.

In practice, we select a basic decoder network to model the mapping from the locations of the two endpoints, p_0 and p_1, onto the density and distance maps. Our network structure is based on the DC-GAN generator [RMC15], where the 6D vector of p_0, p_1 is processed by a MLP with Leaky ReLU layer to get a 16384D feature vector, the feature vector is reorganized into a $8 \times 8 \times 256$ feature map, followed by a series of deconvolutions to generate the final density and distance map in a 256×256 resolution. We normalize the UV-domain to $[0, 1]$ and use the Gaussian function $G(\mathbf{x}, \mathbf{v}_i)$ with variance $\sigma = 0.01$. For a detailed illustration of the network structure, please refer to the supplementary material. Unlike previous methods that try to represent density with fixed number of Gaussians [CDAS20], the number of output Gaussians are conditioned to the two endpoints, as a result, we represent our density model with a regular 2D image generator network, instead of networks designed for fixed-numbered density.

4.2. Rendering with Neural Path Sampling

The neural path sampling provides an efficient way to sample seed paths that are close to the admissible paths, which can be used for root-finding based rendering algorithms. Here we describe how to integrate neural path sampling for unbiased or biased SMS [ZGJ20] for caustics rendering, or follow PathCut [WHY20] to compute energy contribution for glint rendering.

Unbiased Neural Path Sampling Unbiased specular manifold sampling [ZGJ20] starts by random sampling an initial position on the specular object and performing a manifold walk to find a valid path. Then the inverse probability of such a sample is estimated by performing the same random sampling and manifold walk, and logging the number of trials until the same valid path is found again. We can simply replace the uniform sampling for the initial position by an importance sampling scheme where for each surface position $\mathbf{x}$, its sampling probability follows the density map while normalized:

$$p(\mathbf{x}) = \mathbf{K}(\mathbf{x}) / \int \mathbf{K}(\mathbf{y})\, d\mathbf{y}. \tag{3}$$

The importance sampling process is unbiased when the probability map is non-zero everywhere. As a result, to avoid zero-values

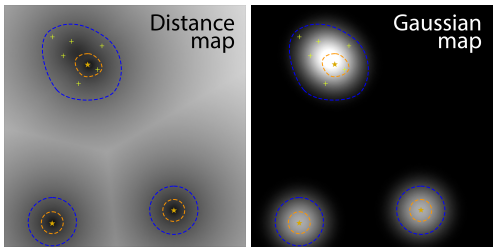

Figure 4: *Biased SMS rendering with neural path sampling. A large distance threshold (blue line) provides a conservative mask Ω_L for possible valid paths. An estimate of the number of valid paths for each region is calculated based on the volume in the density map. A more precise set of valid paths are estimated in the confident region Ω_S defined by a small distance threshold (orange line). We will first sample the precise set (orange star), and then randomly sample (yellow cross) within each conservative mask region based on the estimated valid paths numbers.*

falsely introduced by the network training error, we added a small offset of 0.001 to the whole map before normalization.

Biased Neural Path Sampling Specular manifold sampling has a more efficient mode for biased rendering, which trades bias for low variance [ZGJ20]. Biased SMS also starts by randomly sampling over the specular object and performing a manifold walk to find the valid paths, and the final energy is computed by accumulating contributions from all found unique solutions after N random trials.

We combine our neural path sampling for biased SMS by proposing a better sampling scheme for the N trials. With the density and distance map estimated by the neural network, the path sampling needs to generate a set of $N < N_{max}$ initial paths, where N_{max} is the user-determined maximal number of trials for one specular manifold sampling step.

We first find valid paths encoded by the distance map. Ideally, all valid paths should have a zero value on the distance map, thus we threshold the distance map and mask the region Ω with its distance smaller than a predefined threshold σ_D. Then the number of valid paths within the region can be estimated by:

$$N_{est} = \int_{\mathbf{x}\in\Omega} \mathbf{K}(\mathbf{x})\,\mathrm{d}\mathbf{x} \,/\, \int_{D(\mathbf{y})<\sigma_D} G(\mathbf{y})\,\mathrm{d}\mathbf{y} \tag{4}$$

which is the volume of the $\mathbf{K}$ within that region normalized by the volume of a single Gaussian-kernel G within the distance $D(\mathbf{y})$ defined by the threshold σ_D. Although Eqn. 4 only provides a rough estimate of the number of valid paths within that region Ω, it already acts as a good indicator for determining the number of samples for different cases. In practice, to compute Eq. 4, we first followed the Connected-Component Labeling (CCL) method [ST88] to label each region within the predetermined threshold σ_D, we then performed a reduce-sum operation to compute the integral over the labeled region.

When the estimated number of valid paths is equal to 1, we call it a *single solution region*, since it indicates the distance map and the density map agree with each other, and there is only one valid path within this region. A single trial initialized from the center of the region should be sufficient to find the only valid path. The setting of the σ_D determines the number of single solution regions. A smaller threshold will yield a smaller region and a higher probability to get an agreed single solution. However, setting a small threshold will have the potential risk of missing valid paths, so we designed a two-scale sampling scheme that combines two thresholds.

As illustrated in Figure 4, the algorithm starts by applying a large threshold on the distance map $\sigma_D = 0.05$ and estimates the number of valid paths within each region Ω_L. Then, we apply a smaller threshold $\sigma_D = 0.03$ and get another set of regions Ω_S, which includes more single solution regions. For each region from the large threshold Ω_L, we can estimate its valid solution number N_{est}. We then count the number of single solution regions N_{ssr} among all small threshold regions. Finally, each single solution region will only need a single trial at its center. The remaining number of paths $N_{est} - N_{ssr}$ will need much more effort since we do not have accurate guidance for them. In practice, we will sample $M \times (N_{est} - N_{ssr})$ initial paths within this region Ω_S for each estimated valid path not covered by N_{ssr}. The probability is proportional to the $N_{est} - N_{ssr}$ belonging to each region cut by the large threshold, and the total number of samples are scaled by M. All the initial paths provided by the single solution region and generated by sampling are concatenated together with the single solution region path at the front, forming the final set of initial paths for biased SMS rendering. Note that the number of our proposed initial paths varies and is generally proportional to the number of valid paths, which is reasonable and will be much more efficient than a uniformly predefined number of trials. Figure 7 demonstrates the variation in the number of samples corresponding to the caustic complexity.

Neural Path Sampling for Glint Highlights PathCut [WHY20] introduced a framework for rendering glint highlights resulting from specular light paths connecting the light source and the camera. This process begins by finding all valid paths between the camera and the light, then computing the contributions of these valid paths using the generalized geometry term and assigning them to the rendered pixels.

We adopt this framework; however, instead of using PathCut, we utilize *biased neural path sampling* to find valid paths by setting the camera and light as the two endpoints. Once the valid paths are identified, we employ the same energy computation and rendering scheme as in PathCut.

5. Training

Ideally, to train the path sampling network, we need paired training data which includes all the valid pure specular path that connect the two given endpoints. However, finding all valid paths incurs prohibited high computation cost for generating a large amount of training data. As a result, we proposed a training scheme which only needs valid paths in a local range, thereby significantly reduces the cost of generating training data.

Independent Valid Path Sampling Although it is difficult to find all valid paths that connect two given endpoints, randomly selecting a valid path regardless of the endpoints is much simpler. We can construct a valid pure specular path by sampling a *seed ray segment*

starting from the surface, with random position and direction, then performing ray-tracing on both directions of the sampled ray segment following pure-specular reflection to determine the full path. Note that, similar to SMS [ZGJ20], the type of pure specular interaction, such as reflection or refraction, must be predetermined or enumerated and treated as distinct cases. When a ray traced from either direction intersects the region of endpoints, positions along the segment within the endpoint region are randomly sampled to establish the two endpoints of this path, and the initial ray-segment originating from one of the endpoints is recorded. In this instance, the first ray-segment is determined when the ray intersects the endpoint region, which differs from the seed ray segment. In this way, a valid path for a pair of endpoints can be sampled independently.

Curvature-based sampling Specular paths change rapidly around high curvature surface regions, as a result, instead of uniformly sample the *seed ray segment*, we include an importance sampling scheme based on the surface curvature. The sampling rate over the surface is proportional to $1+k|c|$, where c represents the surface curvature and k is a predefined scale factor.

In our implementation, we set $k = 10240$. When sampling endpoints, endpoints that are close to each other with their distance smaller than 0.02 are discarded. When sampling over triangle meshes, to avoid numerical instability in curvature computations or very large curvature values, we limit the maximal number of sampled seed ray segments per triangle to 24.

Curvature-based sampling is only applicable for generating training data. It works only when the full path is generated by tracing pure specular rays from both directions of the seed ray segment. One cannot perform curvature-based sampling for a regular unidirectional path tracing. For example, consider a curved reflector positioned underneath a flat transparent slab. In the case of unidirectional or bidirectional sampling, the ray will first hit the flat slab, there is no closed-form solution available for determining the correct samples follows curvature-based sampling on the curved reflector after the first refraction on the flat slab.

Local Valid Paths Search Manifold walk [JM12,ZGJ20] provides an efficient way to explore local properties around an existing valid path. We use manifold walk to enhance our training data by performing a local search. For each sampled endpoint pair in one independently sampled path, we fix the endpoints and perturb the first interaction position around the current path, following a 7×7 regular grid pattern in the UV domain around the first interaction position. For each perturbed path, we run manifold walk optimization to find a valid path and record all unique solutions as training data.

Sampling Negative Examples The valid path search only finds valid light paths that successfully connect two endpoints. However, there are certain pairs of endpoints that are not connected by any valid path. Our network needs to detect such cases and indicate that there is no valid path. To find those negative examples, we randomly sample endpoints within the predefined ranges and find the nearest valid path among all sampled valid paths. If the distance to the nearest valid path is larger than a threshold d_λ, we mark the sampled endpoint pair as negative, indicating that there is no valid path connecting them. The distance between valid paths is measured by the 6D Euclidean distance between endpoints, with the 3D positions of the two endpoints concatenated as a 6D vector. The threshold d_λ is determined by the maximal nearest distance between the sampled valid paths, denoted as d_{max}. In practice, we set d_λ to be $4\times d_{max}$.

Training with Local Valid Paths Before training, the sampled valid paths and negative examples are first converted to ground truth 2D maps following Equation 1 and 2. If there is no valid path for the given endpoints, we set a constant value of $\mathbf{K}(\mathbf{x}) = 0$ and $\mathbf{D}(\mathbf{x}) = 10.0$.

To train the neural path guiding network, we use an L1 loss which compares the difference between the estimated map and ground truth map. Since our training data only consists of valid path information in a local region, the L1 loss is masked by a Gaussian function $M(\mathbf{x})$ with $\sigma = 0.1$ centered at the local valid path position. For the remaining region, we simply apply a regularization term. We regularize the density map with a value constraint $R_K(\mathbf{x}) = |\mathbf{K}(\mathbf{x})|$ that ensures the density map at positions without a valid path has a zero value. Since a valid unsigned distance map should have a unit gradient anywhere except near zero distance, the distance map is regularized with an Eikonal constraint [BW99] $R_D(\mathbf{x}) = \big||\nabla\mathbf{D}(\mathbf{x})| - 1\big|$. Both regularization terms are applied only in the region outside valid paths, using the inverted mask $1-M(\mathbf{x})$. The total loss function is a weighted sum of all the loss functions and regularization terms as:

$$L_{total} = M\left(\lambda_0 L_D + \lambda_1 L_K\right) + (1-M)\left(\lambda_2 R_K + \lambda_3 R_D\right), \quad (5)$$

with weights $\lambda_0 = 0.5$, $\lambda_1 = 0.5$, $\lambda_2 = 0.005$, and $\lambda_3 = 0.005$. Note that the pixel position $\mathbf{x}$ is omitted for simplicity, for example M stands for $M(\mathbf{x})$.

The network is trained with the Adam optimizer, using a learning rate of 0.0001. The batch size is 128 for outputting 256×256 result maps. We follow [KALL18] to apply the progressive training scheme. We start by outputting a pair of 64×64 maps, and adjust the batch size and learning rates following a standard progressive training scheme. Note that during the progressive training process, the distance map always uses the distance metric defined by the pixel size of the finest resolution.

6. Results

We train the path guiding neural network with TensorFlow [AAB*15]. The network inference is accelerated with TensorRT [NVI22] using half-precision (FP16) computation. The ray-tracing is implemented with DirectX Ray-tracing, and the specular manifold walk is implemented with the autodiff C++ library [Lea18]. To reduce context switch overhead, we pack 16×16 pairs of endpoints into one batch and perform the network inference and initial path generation for the whole batch. We will release our code and data upon acceptance.

Our system is trained and tested on a workstation equipped with an Intel Core i9-10900X CPU and an NVIDIA RTX 3090 GPU. The neural network inference and ray-intersection processes are executed on the GPU, while the remaining computations, such as root-finding, are performed on the CPU.

To validate the efficiency of our system, we examine 4 scenes

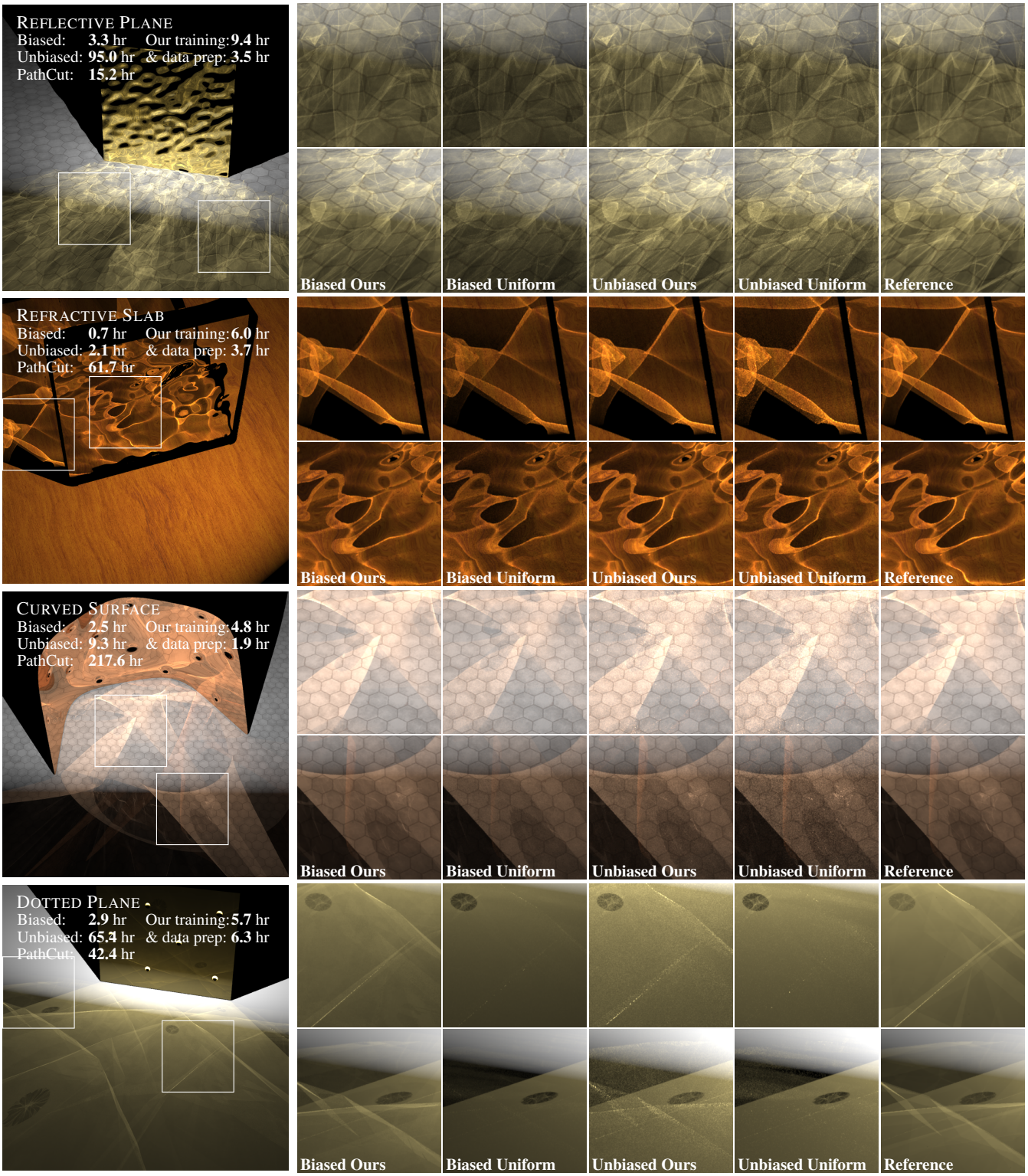

Figure 5: *Equal rendering time comparison between original SMS [ZGJ20] using uniform initial path sampling and our neural path sampling, both unbiased and biased versions are included. The rendering time for PathCut [WHY20] to find all valid light paths are also listed. Our method converges much faster and matches the reference well. Using the uniform initial path sampling, the biased SMS suffers from significant energy loss, resulting in a darker image; however, the unbiased SMS generates strong noise.*

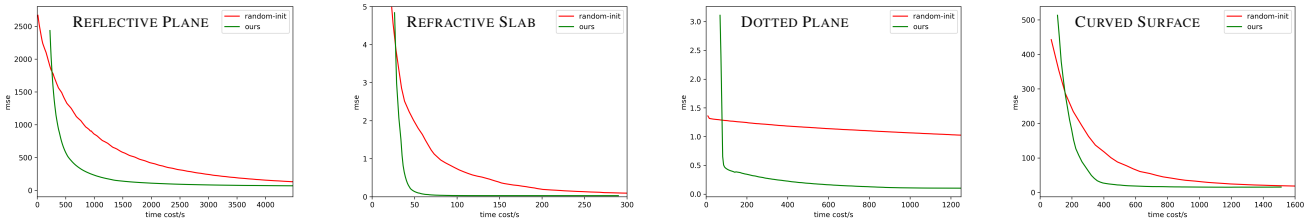

Figure 6: *Convergence plot with mean square error (MSE) over time for biased SMS estimation with initialization and neural path sampling. Our result outperforms SMS with uniform initial path sampling by a large margin. For the neural path sampling, the network inference time and sampling time are included, which contributes to the horizontal offset of the first point of our method.*

with challenging caustics and sparkle highlights due to pure specular light paths. The REFLECTIVE PLANE and REFRACTIVE SLAB scenes are classical examples with difficult specular paths in refraction and reflection cases. The CURVED SURFACE scene exhibits long-range, multi-bouncing reflection paths. Some of the caustics on the ground are caused by pure specular paths having more than 2 pure-specular bounces, which are difficult to sample. Some man-made objects have simple shapes decorated with sparse fine geometry details, such as the DOTTED PLANE scene. (The specular coefficient of the planar region of this specular object is scaled down to emphasize the caustics caused by the dotted patterns.) Aside from the simple caustic cast by the planar region, many detailed caustic patterns are produced by those small dots. However, because those dots are very small and sparsely distributed, uniform sampling requires a lot of trials to reach those dotted features. Aside for the 4 challenging scene commonly used for testing pure specular rendering algorithms, to evaluate the performance of our method on large-scale real-world scenes, we also incorporated THE BREAKFAST ROOM, designed by 'Wig42', which is a widely-used benchmark for testing rendering algorithms. We made modifications to the materials in the scene to include transparent and specular objects.

6.1. Comparison

Our test scene contains both complex caustics and specular highlights. The only methods that can support both cases are specular manifold sampling (SMS) [ZGJ20] and PathCut [WHY20]. The original SMS uses uniform sampling to generate the initial paths for the root-finding, whereas our method improves SMS with initial paths generated by the neural path sampling. Here we compare SMS with uniform initial path sampling and neural path sampling with both biased and unbiased estimators being tested. While PathCut is originally designed for finding pure specular paths directly from the light to the camera, it can be extended to render caustics by running PathCut between all endpoint pairs that require pure specular connections. Please reference the supplementary materials for the implementation details about SMS and PathCut.

In Figure 5, we provide equal-time rendering results for both biased and unbiased SMS, with and without our neural path guiding. A reference image rendered with 1 Mspp unbiased SMS is also provided. All images are rendered in a 1024×1024 resolution. For biased estimators, an 8-spp image is rendered, with each *spp* ren-

Table 1: *Rendering time (hours) required to achieve equal quality results. Training data preparation time and training time are also listed. Note that network training is only necessary once, as it enables the object to be rendered under various viewing and lighting conditions.*

	Biased		Unbiased		Data	Net
	Ours	Uniform	Ours	Uniform	Prep	Train
REFLECTIVE PLANE	3.3	10.1	31.7	95.0	3.5	9.4
REFRACTIVE SLAB	0.7	4.2	0.6	2.1	3.7	6.0
CURVED SURFACE	2.5	4.1	3.9	9.3	1.9	4.8
DOTTED PLANE	2.9	83.9	14.5	65.4	6.3	5.7

Table 2: *We compare the trials and SPP number statistics for the biased and unbiased estimators respectively, using either uniform or neural initial path sampling methods. The numbers for equal rendering time (ET) correspond to Figure 5, while the numbers for equal quality (EQ) correspond to Table 1.*

	Trials (Biased)			SPP (Unbiased)		
	Ours	Uni.ET	Uni.EQ	Ours	Uni.ET	Uni.EQ
REFLECTIVE PLANE	273.8	697	2135	4096	4562	13628
REFRACTIVE SLAB	32.9	140	840	512	643	2179
CURVED SURFACE	255.3	388	620	3072	4186	9982
DOTTED PLANE	284.4	653	19280	2048	2201	9937

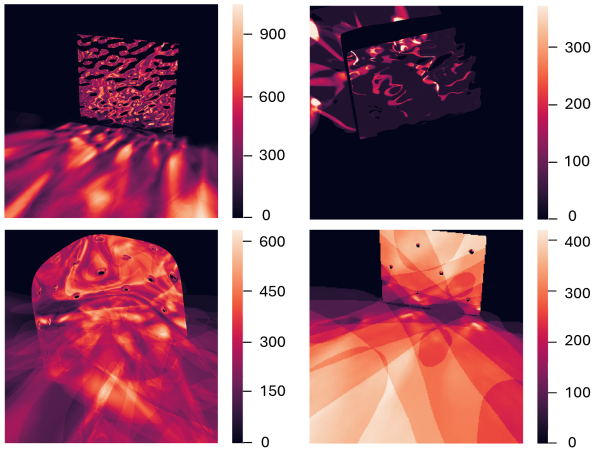

Figure 7: *Visualizations of the number of trials using biased SMS with neural initial path sampling, the heat map legend is displayed to the right of each image. Note that the number of trials correlates well with the caustic distribution.*

dered independently, while different methods uses different number of trials in equal-time. The *spp* for unbiased estimators varies due to equal allocation of rendering time. The rendering times for each scene are listed in the figure. Note that with the same rendering time, our method can correctly render the complex caustics with both biased and unbiased estimators. However, using the same amount of rendering time, the original SMS method with uniform initial path samples [ZGJ20] suffers from either loss of energy for the biased version or strong noise for the unbiased version. This is especially pronounced in shapes with regular patterns rather than noise-like normal maps, as in the DOTTED PLANE scene, the caustic pattern caused by those concave dots is missing even for the unbiased version rendering results.

We also compare the number of trials and SPPs in Table 2. For biased SMS estimators, the number of trials is listed. Since our method generates samples adaptively based on the predicted number of valid paths, we only list the average number of trials and provide a visualization of the distribution in Figure 7. Our method converges quickly with fewer trials, thanks to the improved initial path sampling. Additionally, as illustrated in Figure 7, our number of trials adapts according to the number of valid paths estimated by the neural network. This results in the number of trials correlating with the caustic complexity, allowing us to concentrate computation only on those pixels that are difficult to render. The same applies to the unbiased SMS estimator, where our method converges faster with fewer SPPs required to achieve similar quality results.

The rendering results of THE BREAKFAST ROOM using the biased SMS estimator, with and without neural path guiding, are shown in Figure 1. Both methods have the same rendering computation time of 2.2 hours. To achieve this result, we train separate neural networks for each object and iteratively sample paths for each object separately. Notably, our method generates noise-free results while the original SMS [ZGJ20] suffers from noise and energy losses.

Since PathCut [WHY20] has to find all possible solutions, we cannot provide an equal-time comparison. Instead, we list the rendering time of PathCut in Figure 5 for reference. Although for scenes dominated by single bounce interactions, PathCut shows its advantage in the total rendering time compared to unbiased SMS with path guiding. For scenes with multiple bounces caustics, like the CURVED SURFACE scene, PathCut takes a much longer time to render. In addition, our rendering results combined with biased SMS also matches the reference well, yet runs significantly faster than PathCut. Since the PathCut finds all possible paths, its rendering result is the same as the reference image; as a result, we did not include additional rendering results in the figure.

We also included an equal-quality comparison listed in Table 1 and showed how much time is needed for SMS with uniform sampling to achieve the same MSE results as our results that are shown in Figure 5. For biased SMS rendering, our neural path guiding scheme provides significant speedup with up to a 28 times faster rendering time for the DOTTED PLANE scene. Taking into account the training time and rendering of a single frame, our method remains 5 times more efficient in this particular scenario. The acceleration also confirms with the convergence curve plotted in Figure 6, which shows that the biased SMS estimator converges much faster with initial paths sampled by our method compared to uniformly sampled initial paths. For unbiased SMS, our neural path guiding scheme also leads to an average speedup of 3.4 times. Considering that unbiased rendering is much slower, even such a speedup factor also leads to considerable time savings.

Since our method needs to infer the neural network and generate samples based on the network output alongside the rendering process, we have collected such pre-computation time and marked it as the offset at the beginning of the plot in Figure 6. Please note that this is merely for illustration purposes, so that all extra computations can be represented in one place. Among this pre-computation time, the network inference, CCL [ST88], reduce-sum, and sample generation contribute $52.7\%, 37.7\%, 1.5\%, 2.2\%$ to the total duration, respectively. Generally, the pre-computation time is significantly shorter compared to the more time-consuming root-finding rendering process. Here, we primarily discuss the extra computation time in our algorithm that scales with the number of primary ray samples. The preparation time for training data and the actual training time are intended for multiple rendering frames and remain constant regardless of rendering resolutions and SPPs. Consequently, we omitted these times from the plot and discussed their impact on performance in the subsequent section.

In Figure 8, we compare our method with stochastic progressive photon mapping (SPPM) [HJ09] using two different configurations. The default configuration employs radius reduction with $\alpha = 0.7$, which automatically reduces the gathering radius based on samples. To achieve sharp caustics with SPPM, we follow [ŠOHK16] and set the radius small enough according to its pixel footprint (SPPM-FR) instead of using radius reduction. For the scene setup, we place a sieve, a fine mesh strainer, beneath the specular object to receive the caustic instead of using a plane as the caustic receiver. Due to the high-frequency nature of the sieve shape, SPPM fails to produce satisfactory results, yielding either blurred (with large, automatically reduced radius) or noisy (with small, fixed radius) images. In contrast, our method does not rely on the spatial coherence of the caustic receiver, thus generating clean caustic renderings. Note the results generated by photon mapping exhibit significant energy loss. Therefore, the images are manually rescaled for an optimal viewing experience. Due to this energy loss, numerical comparisons are not included. Additionally, we provide further comparisons between GPU-based SPPM in the supplementary material.

In Figure 9, we present results featuring strong glint highlights that have a specular path connecting the view directly to the light source. The object is illuminated by an environment map and a circular light strip positioned in front of it. The lighting pattern is depicted in the inset illustration. To render the thin light strip, we uniformly sampled multiple point light samples and rendered the glint highlights. Please refer to the supplementary video for the full animation, showcasing the rotating lit-up segment on the strip. To prevent flickering, we allowed all methods to produce visually converged results and compared their rendering times. Although our method requires additional training time, it renders glints much faster than PathCut [WHY20] and SMS [ZGJ20], enabling applications such as lighting design to receive quicker feedback.

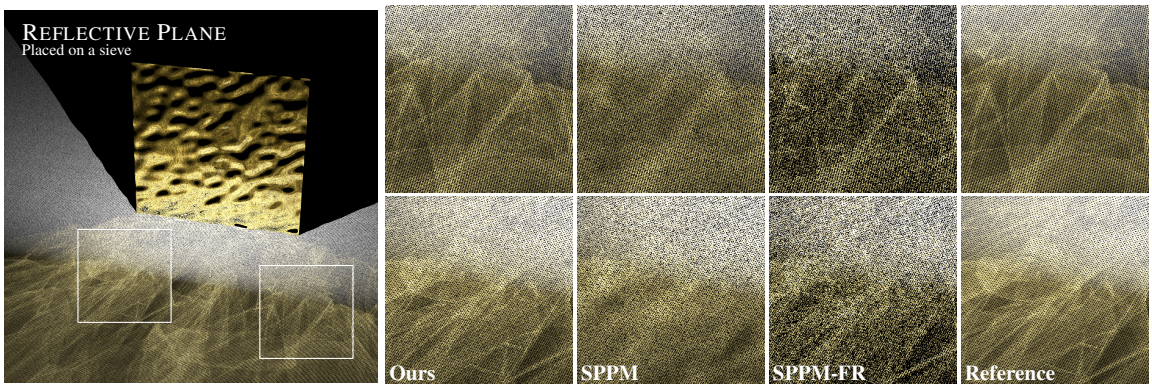

Figure 8: *We compared the equal rendering times (5.5 hours) of our neural path sampling method with biased SMS and SPPM [HJ09]. The default configuration of SPPM, employing automatic radius reduction, yields blurred results. In contrast, using a fixed radius (SPPM-FR) based on pixel footprint results in sharp, albeit noisy, images. Our approach generates high-quality caustics that closely match the reference.*

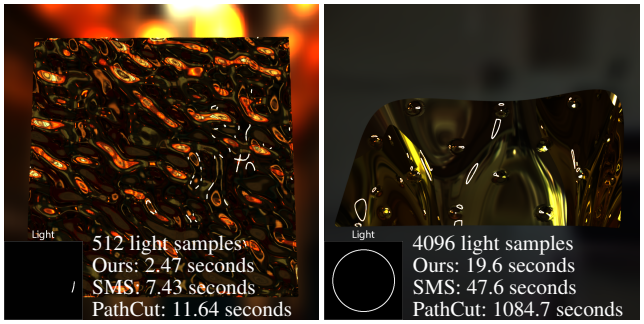

Figure 9: *Rendering of the glinty specular highlights of the* REFLECTIVE PLANE *and the* CURVED SURFACE *scene illuminated by an environment map and a circular light strip (as the insert) positioned in front it. All methods yield visually converged results; therefore, visual comparisons and references have been excluded.*

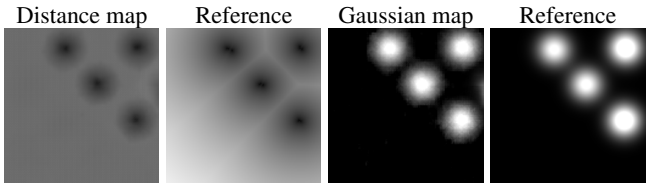

Figure 10: *Visualizations of the neural network output compared with the reference distance and Gaussian maps.*

6.2. Validations and Ablation Studies

Visualizing network output. Figure 10 visualizes one example of the output map from our neural network, trained for the DOTTED PLANE scene. Both the distance map and the Gaussian map roughly match the reference map. The predicted Gaussian map has some noise, but it has little effect when calculating the integral to determine the number of valid paths. The predicted distance map has a larger difference when the distance value is larger, as the local valid path training loss is concentrated in the region close to the valid path. However, since the distance threshold used in the sampling step is much smaller, distance values larger than the thresholds do not affect the quality of the generated initial samples either.

Training with local valid paths. One major advantage of our training scheme is that it does not require dense sampled endpoints. It also does not need a complete set of all valid paths connecting the two endpoints. Finding all valid paths between two endpoints is significantly slower (taking 5-20 seconds) compared to sampling a valid path using local valid path search (which takes 0.01 seconds).

To validate the efficiency of our training scheme, we designed a configuration where the computation of all valid paths is permissible. We use the CURVED SURFACE for this experiment. The point light source is limited to a 1D line, and the other endpoint range is a 2D plane on the ground where the caustic is cast. For this configuration, we prepared three different training datasets. The COMPLETE DATASET includes uniformly sampled endpoints and all valid paths connecting those endpoints; the INDEPENDENT SAMPLED DATASET includes valid paths with uniform path space sampling, following section 5, and the LOCAL SEARCH DATASET includes valid paths with uniform path space sampling and local search, as described in section 5. We then train our path guiding network on the three datasets with the same configuration and compare the rendering results. Figure 13 (left) plots the MSE over time, our incomplete training scheme combined with LOCAL SEARCH produces similar quality results compared with the COMPLETE DATASET, while simply INDEPENDENT SAMPLING leads to inferior training results.

Curvature-based sampling. Instead of sampling the training data uniformly on the object, we focus the training data on geometry features with high curvatures, leading a more efficient training dataset. Figure 11 compares equal-time rendering of biased SMS based on networks trained with and without curvature-based sampling. This sampling scheme is designed for training data generation, since the sampling is not limited to the first pure-specular bounce. A similar scheme can be applied for sampling the first bounce light path, but will fail when the high curvature feature appears on the second or higher bounces.

Training time. Although one trained neural network can be used

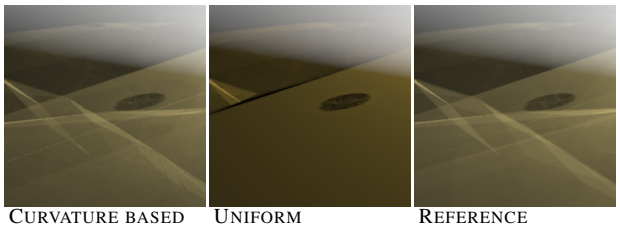

Figure 11: *Comparing biased estimators with the path sampling network trained with training data generated by curvature-based sampling or uniformly sampled training data. Note that for challenging cases like the* DOTTED PLANE *scene, curvature-based sampling helps generate efficient training data and the trained network can sample critical paths efficiently. On the other hand, when generating the same amount of training data, uniformly sampling training data generates a large amount of data for the planar region. The trained network cannot provide efficient sampling, and the rendering result still misses critical caustics from the high curvature dotted region.*

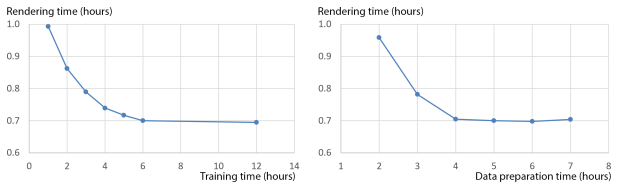

Figure 12: *Equal quality rendering time plot with path sampling network trained with different training time budgets and different training data generation time budgets. The rendering time is measured based on the biased SMS with neural path sampling for the* REFRACTIVE SLAB *scene, as shown in Figure 5.*

for rendering arbitrary number of images, the training still takes a significant amount of time if only a few rendering images are needed. In such case, one might choose to trade image rendering cost for training cost by training the neural network with fewer iterations or generating fewer number of training data. We choose the REFRACTIVE SLAB scene as an example and train different neural path sampling network with different training configurations. We then measure the rendering time of biased SMS estimator when reaching a noise-free result (with a 0.2 MSE) to indicate the quality of the trained neural network.

To this end, we analyze the effect of training iterations and number of data separately. We first fix the training data size to 2.2 million sampled paths and train the network with a different number of iterations. Figure 12 plots the rendering time for reaching a noise-free image against the network training time. Generally, the converged network provides the best acceleration for rendering multiple frames. However, a network trained with fewer iterations still provides significant benefits when only a few images need to be rendered. We then fix the number of training iterations but vary the size of generated training data. Since the size of training data is proportional to the time used for generating the training data, we plot the noise-free rendering time against the time used for generating training data. As shown in Figure 12, the rendering time increases as the training data generation time decreases.

Note that rendering challenging pure-specular light transport usually takes several hours; However, thanks to the efficient data-generation scheme the training data generation and network training time are not significantly longer compared to the lengthy rendering time. For instance, in the DOTTED PLANE scene, the training data preparation and neural network training take 12 hours in total. The combined training and rendering time using our method takes 14.9 hours for a single frame. On the other hand, the original SMS with uniform initial path sampling takes 83.9 hours to reach an equal-quality result. For a short 90-frame video, as shown in the supplementary video, the rendering takes hundreds of hours even with the faster biased estimator, diminishing the training time cost. For the four test scenes listed in Table 1, our method begins to show speed benefit when rendering only $2, 3, 5$, and 1 frames, respectively.

Training for a single frame Although our method is not specifically designed for training and rendering a single frame with a fixed, single point light source configuration, we evaluated the performance of our system under extreme conditions by training and testing it for rendering a single frame of the REFRACTIVE SLAB scene. With this simplified configuration, we were able to reduce the training data preparation and training time to 0.6 and 2.4 hours, respectively, while still maintaining a significant speed increase with the trained network. Utilizing a guided initial path, the biased estimator took 0.8 hours to achieve equal quality results, as demonstrated in Table 1. The combined time for data preparation, training, and rendering (3.8 hours) remains shorter than the biased estimator using uniform sampling (4.2 hours).

Parameterization. In line with previous root-finding-based algorithms [ZGJ20], the initial path sampling in our method is performed in a parameterized 2D domain. To ensure a fair comparison, we provide the same UV-parameterization for all tested methods. All the specular objects in the THE BREAKFAST ROOM scene have their UV texture charts automatically generated by Blender using its default settings. The network will learn the maps defined on disconnected texture charts. Nevertheless, our approach is not bound to a specific 2D UV-parameterization.

In this experiment, we opted for a view-based parameterization instead of relying on a fixed, predefined UV-parameterization. Specifically, we define the 2D map domain as the angular domain centered at one of the endpoints, p_0. We use the spherical coordinates defined on the unit sphere and restrict the θ and ϕ ranges by the object's bounding box viewed from p_0. This view-based parameterization overcomes the limitations of UV-parameterization while encompassing all feasible paths that originate from p_0. We evaluated our method using this view-based parameterization on the concave object, CURVED SURFACE. Our view-based implementation takes 2.6 and 4.4 hours respectively, for biased and unbiased estimators, to achieve similar quality results as shown in Table 1. This result suggests that our method performs well, even without a predefined 2D parameterization.

Different network output maps. Our network design and sampling scheme is based on both a distance map which models the position of individual valid paths and a density map which models

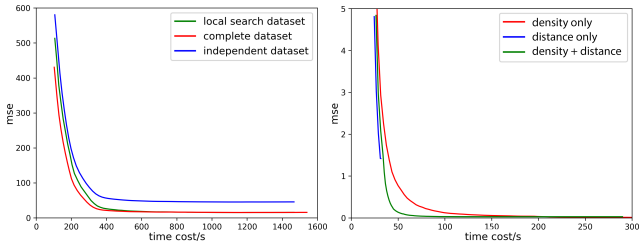

Figure 13: *Convergence plot with mean square error (MSE) for biased SMS with neural path sampling using different configurations. The plots are measured on the* CURVED SURFACE *scene (left) and the* REFRACTIVE SLAB *scene (right) respectively.* ***Left:*** *Neural path sampling network trained with only independent sampled training data (blue) misses critical valid paths leading to slow convergence. Our local search scheme (green) produces a similar convergence speed as the network trained with the full dataset (red) while taking much less time for training data generation.* ***Right:*** *Generating the initial paths with only the distance map (blue) can rapidly improve the result but failed to converge; with only the density map (red) the early sampled paths are less efficient, thus the error decreases slower; our sampling scheme combines both the distance map and density map (green) leads to fast convergence.*

the density of the valid paths. To validate the effect of each component, for the REFRACTIVE SLAB scene, we trained two variants of the network that only output either the distance map or the density map and plotted the rendering error decrease over time. With only the distance map, the neural path sampler first samples all the local minimal peaks of the distance map and then uniformly sample around those local minimal peaks. With only the density map, the sampler will uniformly sample based on the density map. As shown in the plot in Figure 13 (right), the distance map provides good location information of valid paths thus the rendering error decreases rapidly with early samples. However, without the density map providing the density information (following Eqn. 4) it stops without converging to the correct result. The density map based sampling works well when the sampling number is large, however, without the accurate location provided by the distance map, its early samples are not very efficient. By combining both the location and density information, our sampling scheme achieves efficient rendering on both the early and later stages of the sampling.

Dynamic scene and temporal stability. Our path sampling network is trained for the specific pure-specular object while the rest of the scene can be dynamically changed. In the supplementary video, we show rendering sequences under dynamic lighting conditions, dynamic scene, and moving of the pure-specular objects. To show our method is temporally consistent, the video is rendered with biased SMS, where any inconsistency will be more pronounced. The supplementary video shows our method supports putting the pure-specular object in a dynamic scene and produces temporally stable results.

7. Conclusion

We introduced neural path sampling, a learning-based method to generate initial path samples for rendering pure specular light transport. We utilize a density map and a distance map to accurately represent the position and distribution of valid paths connecting two endpoints. Such a representation can be used to generate initial path samples for both unbiased and biased specular manifold sampling [ZGJ20]. We also designed a unique training scheme that does not require finding all valid paths connecting a pair of endpoints, significantly reducing the computation cost for training data generation.

Our results show the neural path sampling method can significantly improve the rendering performance of pure specular light transport. Since our trained network works for any new lighting and viewing conditions, the network can be used to accelerate the rendering of any new frame.

We specify the endpoints as 3D points throughout this paper, however, our representation is not limited to connecting only 3D points. Our method and network structures can be easily modified to support directional lighting where the endpoints are parameterized in a 2D space instead of 3D, thus providing efficient sampling for root-finding algorithms designed for distant lighting conditions.

Our guiding network is trained for one specific object, thus, we only support rigid transformations of the specular object relative to the scene. Like other scene-specific neural network models, one possible extension would be introducing additional conditional control parameters, like the thickness of the object, covering the light transport for all different deformations of the object within one network.

Our path sampling scheme is designed for root-finding-based pure specular light transport rendering, however, the representation could also be beneficial for encoding other sparsely distributed and difficult paths. One direct solution would be combing the idea of representative specular paths [LWT*22] to support rough surfaces. We wish to pursue these future directions and envision a unified path sampling strategy that leverages machine learning models for efficient generic Monte Carlo rendering.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/. 7

[BW99] BORN M., WOLF E.: *Principles of Optics*. Cambridge University Press, 1999. 7

[CDAS20] CURRIUS R. R., DOLONIUS D., ASSARSSON U., SINTORN E.: Spherical gaussian light-field textures for fast precomputed global illumination. *Computer Graphics Forum 39*, 2 (2020), 133–146. 5

[CK20] CONTY A., KULLA C.: Adaptive Caustics Rendering in Production with Photon Guiding. *EGSR Industry Track* (2020). 3

[FHG*23] FAN Z., HONG P., GUO J., ZOU C., GUO Y., YAN L.-Q.: Manifold path guiding for importance sampling specular chains. *ACM Trans. Graph.* (2023). 4

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph. 31*, 6 (nov 2012). 1, 3

[HDF15] HANIKA J., DROSKE M., FASCIONE L.: Manifold next event estimation. In *Computer graphics forum* (2015), vol. 34, Wiley Online Library, pp. 87–97. 2, 3

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, Association for Computing Machinery. 1, 3, 10, 11

[HKD15] HANIKA J., KAPLANYAN A., DACHSBACHER C.: Improved half vector space light transport. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 65–74. 2, 3

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Trans. Graph. 31*, 6 (nov 2012). 1, 3

[HWZ*20] HUO Y., WANG R., ZHENG R., XU H., BAO H., YOON S.-E.: Adaptive incident radiance field sampling and reconstruction using deep reinforcement learning. *ACM Trans. Graph. 39*, 1 (jan 2020). 3

[JC22] JHANG J.-W., CHANG C.-F.: Specular manifold bisection sampling for caustics rendering. *Computer Graphics Forum 41*, 7 (2022), 247–254. 2, 3

[JDZJ08] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. *ACM Trans. Graph. 27*, 1 (Mar. 2008). 3

[Jen95] JENSEN H. W.: Importance driven path tracing using the photon map. In *Eurographics Workshop on Rendering Techniques* (1995), Springer, pp. 326–335. 3

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (Berlin, Heidelberg, 1996), Springer-Verlag, p. 21–30. 1, 3

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (July 2012). 2, 3, 7

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 143–150. 3

[KALL18] KARRAS T., AILA T., LAINE S., LEHTINEN J.: Progressive growing of gans for improved quality, stability, and variation. *ICLR* (2018). 7

[KD13] KAPLANYAN A. S., DACHSBACHER C.: Path space regularization for holistic and robust light transport. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 63–72. 3

[KGPB05] KRIVANEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 550–561. 3

[KHD14] KAPLANYAN A. S., HANIKA J., DACHSBACHER C.: The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph. 33*, 4 (July 2014). 2, 3

[Lea18] LEAL A. M. M.: autodiff, a modern, fast and expressive C++ library for automatic differentiation, 2018. URL: https://autodiff.github.io. 7

[LLZ*20] LIN Z., LI S., ZENG X., ZHANG C., JIA J., WANG G., MANOCHA D.: Cppm: Chi-squared progressive photon mapping. *ACM Trans. Graph. 39*, 6 (Nov. 2020). 3

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques* (1993). 1, 3

[LW95] LAFORTUNE E. P., WILLEMS Y. D.: A 5d tree to reduce the variance of monte carlo ray tracing. In *Eurographics Workshop on Rendering Techniques* (1995), Springer, pp. 11–20. 3

[LWT*22] LI H., WANG B., TU C., XU K., HOLZSCHUCH N., YAN L.-Q.: Unbiased caustics rendering guided by representative specular paths. In *SIGGRAPH Asia 2022 Conference Papers* (New York, NY, USA, 2022), SA '22, Association for Computing Machinery. 3, 13

[LZHJ20] LOUBET G., ZELTNER T., HOLZSCHUCH N., JAKOB W.: Slope-space integrals for specular next event estimation. *ACM Trans. Graph. 39*, 6 (Nov. 2020). 3

[MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum 36*, 4 (July 2017), 91–100. 2, 3

[MH92] MITCHELL D., HANRAHAN P.: Illumination from curved reflectors. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, Association for Computing Machinery, p. 283–291. 3

[MJJG18] MARCO J., JARABO A., JAROSZ W., GUTIERREZ D.: Second-order occlusion-aware volumetric radiance caching. 3

[MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Trans. Graph. 40*, 4 (July 2021). 3

[NVI22] NVIDIA: Tensorrt, 2022. URL: https://developer.nvidia.com/tensorrt. 7

[OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, Association for Computing Machinery, p. 181–188. 3

[RHJD18] REIBOLD F., HANIKA J., JUNG A., DACHSBACHER C.: Selective guided sampling with complete light transport paths. *ACM Trans. Graph. 37*, 6 (Dec. 2018). 3

[RHL20] RUPPERT L., HERHOLZ S., LENSCH H. P.: Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 147–1. 3

[RMC15] RADFORD A., METZ L., CHINTALA S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015). 5

[SHJD22] SCHÜSSLER V., HANIKA J., JUNG A., DACHSBACHER C.: Path guiding with vertex triplet distributions. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 1–15. 3

[SHVJ18] SPEIERER S., HERY C., VILLEMIN R., JAKO W.: *Caustic Connection Strategies for Bidirectional Path Tracing*. Tech. rep., Pixar Technical Memo, 2018. 3

[ŠOHK16] ŠIK M., OTSU H., HACHISUKA T., KŘIVÁNEK J.: Robust light transport simulation via metropolised bidirectional estimators. *ACM Trans. Graph 35*, 6 (2016), 245. 10

[ST88] SAMET H., TAMMINEN M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 4 (1988), 579–586. 6, 10

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 3

[VG95] VEACH E., GUIBAS L.: Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 1995, pp. 145–167. 1, 3

[VKv*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. 33*, 4 (July 2014). 2, 3

[WDH*21] WEIER P., DROSKE M., HANIKA J., WEIDLICH A., VORBA J.: Optimised path space regularisation. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 139–151. 3

[WHY20] WANG B., HAŠAN M., YAN L.-Q.: Path cuts: Efficient rendering of pure specular light transport. *ACM Trans. Graph. 39*, 6 (Nov. 2020). 2, 3, 5, 6, 8, 9, 10

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, Association for Computing Machinery, p. 85–92. 3

[WZHB09] WALTER B., ZHAO S., HOLZSCHUCH N., BALA K.: Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph. 28*, 3 (July 2009). 3

[XWW23] XU X., WANG L., WANG B.: Efficient caustics rendering via spatial and temporal path reuse. *Computer Graphics Forum* (2023). 4

[ZGJ20] ZELTNER T., GEORGIEV I., JAKOB W.: Specular manifold sampling for rendering high-frequency caustics and glints. *ACM Trans. Graph. 39*, 4 (July 2020). 1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13

[ZXS*21a] ZHU S., XU Z., SUN T., KUZNETSOV A., MEYER M., JENSEN H. W., SU H., RAMAMOORTHI R.: Hierarchical neural reconstruction for path guiding using hybrid path and photon samples. *ACM Trans. Graph. 40*, 4 (jul 2021). 3

[ZXS*21b] ZHU S., XU Z., SUN T., KUZNETSOV A., MEYER M., JENSEN H. W., SU H., RAMAMOORTHI R.: Photon-driven neural reconstruction for path guiding. *ACM Trans. Graph. 41*, 1 (Nov. 2021). 3