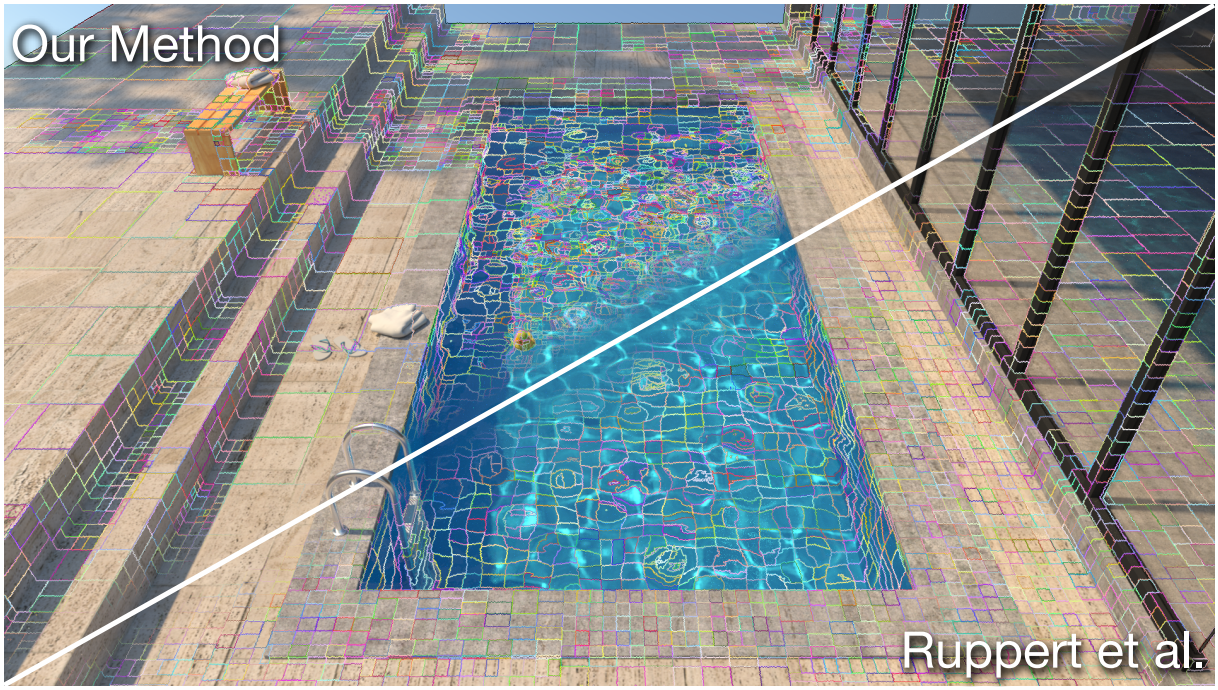
Spatial Subdivision for Path Guiding



Fengshi Zheng

Master's Thesis
July 2024

Overseen by
Prof. Dr. Markus Gross
Supervised by
Sebastian Herholz
Dr. Marco Manzi
Dr. Marios Papas

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl
computer graphics laboratory

Abstract

Path guiding is a technique to facilitate Monte-Carlo-based rendering algorithms by incorporating the scene's spatial-directional incident radiance information into local sampling decisions. State-of-the-art path-guiding methods typically use a k-d tree to subdivide the scene space, and then cache the directional incoming light distribution at the tree's leaf nodes. While significant work has been done to improve the representation of directional components, the spatial structure remains underexplored, often relying on simple subdivision heuristics based on the number, mean, and variance of samples arriving at each node.

This thesis mainly focuses on k-d-tree-based spatial subdivision schemes for path guiding. We decompose the problem into "when and where to split" and address them separately. For "when to split", we present two adaptive splitting algorithms either based on divergence estimates or multiple importance sampling, which make split decisions responsive to the light change at a node. For "where to split", we propose several geometry- and lighting-aware algorithms derived from decision trees metrics. We demonstrate their ability to improve the subdivision quality by fitting the shapes of geometries or lighting in the scene.

Finally, we introduce our comprehensive models by combining the "when and where to split" algorithms. Rendering experiments across diverse scenes show that our best models consistently outperform previous methods while maintaining an equal or lower tree node budget. Additionally, we explore alternative spatial structures beyond k-d trees.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Background

In the world of computer graphics, physics-based rendering is the science of synthesizing realistic images from scene descriptions. Monte Carlo rendering is the governing algorithm that simulates light transport from the sources to the camera by generating random samples of paths in the scene. To accelerate Monte Carlo rendering, path guiding is a technique that collects path data on the fly gained during the rendering, which can be later used to guide local sampling decisions. A fundamental problem in path guiding is the representation of the scene's spatial-directional lighting distribution. Common path-guiding methods first subdivide the scene into a 3D structure and then store some cache per local region to approximate the 2D directional incoming light distribution.

While numerous works have been focused on improving directional representations such as histograms, quadtrees, and Gaussian mixture models, very few addressed the optimization of spatial structures and their subdivision schemes. Mainstream path-guiding methods use a k-d tree with a simple sample-count criterion as its subdivision scheme. Since this criterion does not account for the underlying characteristics of the lighting, it leads to suboptimal spatial structures for guiding, appearing as inefficient sampling during rendering. Instances are regions with highly spatially varying lighting, such as caustics and shadows, being insufficiently subdivided (see the red square in fig. 1.1), uniformly lit areas being over-subdivided, or points from distinct surfaces being grouped into one region (see the blue arrows in fig. 1.1).

This thesis strives to explore some commonly used spatial subdivision approaches in computer graphics and find ones suitable for path guiding. We tailor our proposed methods to address the aforementioned subdivision problems while maintaining high compatibility with existing path-guiding algorithms.

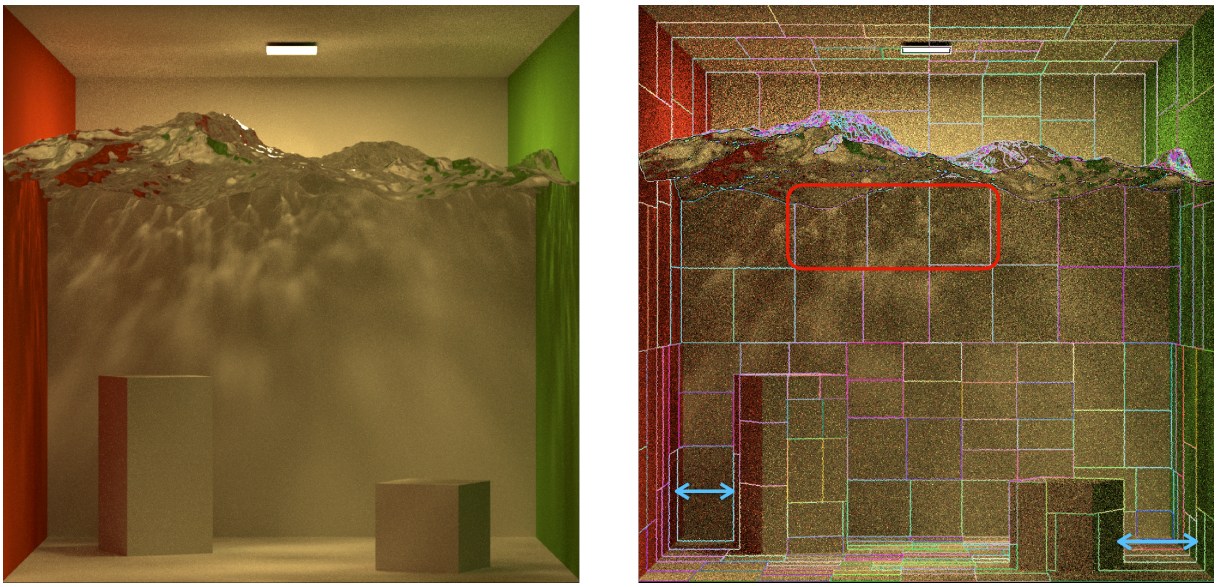


Figure 1.1.: *Demonstration of problems in a sample guiding structure in Water Caustics scene. Left: reference image. Right: rendering overlaid with guiding region boundaries. The red square highlights some regions that do not consider the high-frequency caustics pattern, which should have been subdivided more finely. The blue arrows indicate the regions that span across opposing surfaces, mixing up the guiding distributions from two geometries.*

1.2. Contribution

Our main contribution comprises the following aspects:

- **Summary of existing problems**: we study the suboptimal subdivisions resulting from existing methods. Based on this, we outline a list of requirements for the design of improved subdivision schemes.
- **Adaptive k-d trees**: we enhance the k-d tree subdivision scheme with adaptive criteria, based on either divergence estimates or multiple importance sampling.
- **Geometry- and lighting-aware subdivision**: we develop a family of efficient algorithms to find the split position based on sample variance and information gain.
- **Theoretically optimal subdivision**: we give a lower bound of rendering error due to spatial marginalization at a guiding region, and based on which, a theoretically optimal split strategy for k-d tree subdivision.
- **Bottom-up subdivision approaches**: we explore several bottom-up subdivision approaches that are based on clustering or bounding volume hierarchy.
- **Comprehensive analysis**: we experiment with our proposed algorithms with modular and rendering experiments to understand and analyze their behaviors.

1.3. Overview

This thesis is organized into seven chapters.

In chapter 1, we introduce readers to the problem we consider and summarize the key contributions.

In chapter 2, we review some fundamentals about light transport, Monte Carlo integration, and sampling, which form the basics for path guiding and spatial subdivision. We also introduce a collection of commonly used spatial structures in graphics.

Chapter 3 summarizes a few representative path-guiding works with a focus on their spatial subdivision schemes if applicable. Additionally, we include the part of their methodologies which we draw inspiration from. To motivate the proposal of new spatial subdivision schemes, we present a list of example scenes where current methods result in suboptimal spatial subdivision.

In chapter 4, we first provide information about the spatial subdivision problem in the context of path guiding and rendering. Then, we introduce the reader to a few 2D examples to inform the challenges and important aspects of a spatial subdivision.

Chapter 5 is the central part of the thesis, where we present our core methodologies based on k-d tree subdivisions from two perspectives: “when to split”, which determines when a subdivision happens, and “where to split”, which considers the axis and position to split. Next, we conquer these two problems separately. Finally, we combine the when- and where-to-split solutions into our full models.

In chapter 6, we evaluate our proposed algorithms in 2D and rendering experiments. The 2D experiments demonstrate the behavior of the algorithms and offer a detailed analysis. In the rendering experiments, we integrate our methods into a state-of-the-art path guiding library. We then evaluate their performance by comparing them against previous works’ spatial subdivisions.

In chapter 7, we showcase several of our less successful attempts, all of which construct the spatial structure in a bottom-up manner. We give an outline of their pros and cons.

In the final chapter 8, we summarize the proposed methods and the key findings of our research. We provide a list of limitations of our work and suggest future avenues for further investigation.

2

Preliminaries

2.1. The Light Transport Equation

The cornerstone of *Monte Carlo rendering*, or *path tracing*, is the *light transport equation* (LTE) that describes the propagation of light in a scene [Kaj86] . It quantifies how a beam of light transfers from one point to another based on *radiance*, a principled measurement of light. The radiance from surface point $\mathbf{x}$ to an outgoing direction ω_o is given as an integral over all the incoming directions ω_i by

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_s(\mathbf{x}, \omega_o) \tag{2.1}$$

$$L_s(\mathbf{x}, \omega_o) = \int_\Omega f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n}) \mathrm{d}\omega_i \tag{2.2}$$

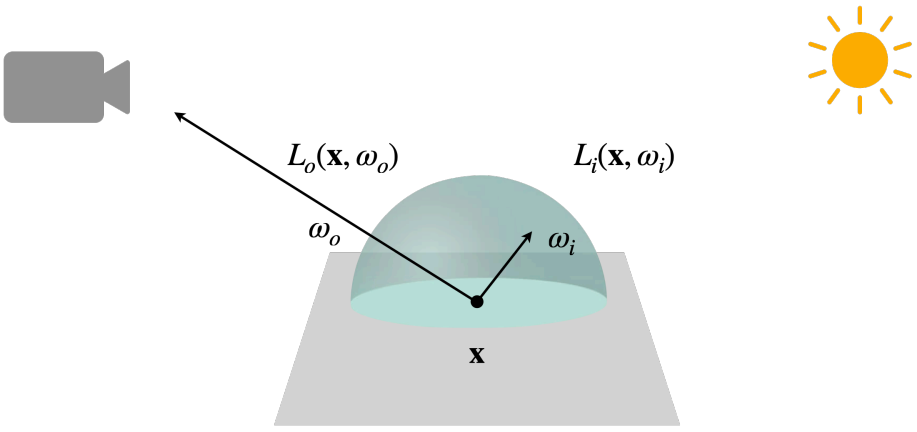


Figure 2.1.: *Illustration of the light transport equation*

The related terminologies are listed as follows:

- L_o is the *outgoing radiance* leaving from $\mathbf{x}$ toward ω_o.
- L_e is the *emitted radiance*.
- L_s is the *scattered radiance*.
- f is the *bidirectional scattering distribution function* (BSDF) that entails the material properties at the scattering surface.
- L_i is the *incident radiance* arriving at $\mathbf{x}$ from ω_i.
- $\omega_i \cdot \mathbf{n}$ is the cosine between the incoming direction and the normal.
- Ω is the incoming directional domain. For reflection-only materials, it is the unit hemisphere of $\mathbf{n}$ on the same side of ω_o. For materials with transmission (e.g. glasses), it is the full unit sphere.

Notably, eq. (2.2) only deals with scenes without participating media, i.e. surface rendering. Hence, the radiance fields are defined on the spatial domain of geometry surfaces, essentially 2D manifolds in the 3D space. In a scene with media, however, LTE takes a more complicated form. In the scope of this thesis, we only focus on surface rendering.

LTE is an integral equation that cannot be solved analytically in most real 3D scenes. Numerical methods such as *Monte Carlo integration* (MC) come in handy in finding approximate solutions.

2.2. Monte Carlo Integration

Consider a general integration problem of a function $f : D \to \mathbb{R}$

$$I = \int_D f(x)\mathrm{d}x \tag{2.3}$$

If we have a random variable X that follows a continuous distribution p, the integral can be estimated by

$$\text{One-sample MC estimator} \quad \langle I(X)\rangle \coloneqq \frac{f(X)}{p(X)} \tag{2.4}$$

$$I = \int_D \langle I(x)\rangle p(x)\mathrm{d}x = \mathbb{E}_{X\sim p}[\langle I(X)\rangle] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \tag{2.5}$$

Eq. (2.5) allows us to approximate the integral eq. (2.3) by taking the average of N random samples from distribution p, each of which has a contribution of $f(X_i)/p(X_i)$. An important property of MC integration is that it makes an *unbiased* estimate of the integral if $f(x) \neq 0 \implies p(x) > 0, \forall x \in D$, which means the math expectation of the estimator equals the true integral. The average of multiple one-sample estimators is called the *N-sample estimator*:

$$\langle I(X)\rangle_N \coloneqq \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}, \quad X_i \sim p \tag{2.6}$$

whose variance is in inverse proportion to the number of samples, if X_i are independent:

$$\mathrm{Var}[\langle I(X)\rangle_N] = \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\langle I(X_i)\rangle\right] \tag{2.7}$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}\left[\langle I(X_i)\rangle\right] \tag{2.8}$$

$$= \frac{1}{N}\mathrm{Var}\left[\langle I(X)\rangle\right] \tag{2.9}$$

This says as we have more and more samples, the estimator will converge to the accurate value of the integral. However, the speed of the convergence depends on the sampling distribution of our choice. The more the p resembles f, the smaller $\mathrm{Var}\left[\langle I(X)\rangle\right]$, and the faster the convergence. Ideally if $p(x) = f(x)/I$, $\langle I(x)\rangle \equiv I$, the variance goes to zero.

Now let's apply MC to solve LTE eq. (2.2). This involves choosing a sampling distribution q over the directional domain, from which we have the scattered radiance estimator

$$\langle L_s(\mathbf{x}, \omega_o)\rangle \coloneqq \frac{f(\mathbf{x}, \omega_i, \omega_o)L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})}{q(\omega_i)} \tag{2.10}$$

$$L_s(\mathbf{x}, \omega_o) = \mathbb{E}\left[\langle L_s(\mathbf{x}, \omega_o)\rangle\right] \approx \frac{1}{N}\sum_{j=1}^{N}\frac{f(\mathbf{x}, \omega_i^{(j)}, \omega_o)L_i(\mathbf{x}, \omega_i^{(j)})(\omega_i^{(j)} \cdot \mathbf{n})}{q(\omega_i^{(j)})} \tag{2.11}$$

The key of the above expression is the $L_i(\mathbf{x}, \omega_i^{(j)})$ term, which requires us to compute the incident radiance coming from the sample direction. This can be done by tracing the ray $(\mathbf{x}, \omega_i^{(j)})$ in the scene, finding the intersection $\mathbf{y}$, and applying eqs. (2.2) and (2.11) to compute $L_o(\mathbf{y}, -\omega_i^{(j)})$ recursively. This process of generating ray samples and tracking the radiance until reaching the light source is named *path tracing*.

Path tracing inherits the unbiasedness from MC integration, as long as $q(\omega_i) > 0$ whenever the integrand product $f(\mathbf{x}, \omega_i, \omega_o)L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n}) \neq 0$. However, the intrinsic randomness of MC makes path tracing suffer from variance which appears as noise in the images.

2.3. Importance Sampling

Importance sampling is a technique to reduce the variance of MC integration. The idea is to select a sampling distribution as proportional to the integrand as possible.

For path tracing, the most naive way is to sample a direction uniformly in the sphere or hemisphere, regardless of the product integrated $f(\mathbf{x}, \omega_i, \omega_o)L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})$. A better way is to importance sample the cosine term $\omega_i \cdot \mathbf{n}$, leaving the estimator proportional to $f(\mathbf{x}, \omega_i, \omega_o)L_i(\mathbf{x}, \omega_i)$. Moreover, we can take BSDF into account, i.e. approximately sampling $f(\mathbf{x}, \omega_i, \omega_o)(\omega_i \cdot \mathbf{n})$, so the variance from the material properties is reduced. Despite this, the variance of the scattered radiance estimator eq. (2.10) can still be high if the incident radiance has high-frequency changes.

To address this, an alternative is to sample according to the light distribution. Typically, we can decompose the scattered radiance into the direct and indirect components

$$L_s(\mathbf{x}, \omega_o) = L_s^{\text{dir}}(\mathbf{x}, \omega_o) + L_s^{\text{ind}}(\mathbf{x}, \omega_o) \tag{2.12}$$

where the first term is defined as the contribution of radiance from light sources that are directly visible from $\mathbf{x}$

$$L_s^{\text{dir}}(\mathbf{x}, \omega_o) \coloneqq \int_\Omega f(\mathbf{x}, \omega_i, \omega_o) L_e(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n}) \mathrm{d}\omega_i \tag{2.13}$$

It can be estimated by sampling a point $\mathbf{y}$ on the union of surfaces of the light sources and checking the visibility from $\mathbf{x}$ to $\mathbf{y}$. This technique is called the *next-event estimation* (NEE).

On the other hand, the indirect lighting in eq. (2.12) has to be computed in the path tracing routine recursively.

2.4. Defensive Sampling

On diffuse or rough materials, the change of $f(\mathbf{x}, \omega_i, \omega_o)$ is smooth, so when there is not much occlusion, NEE can effectively sample the direct illumination eq. (2.13). However, when the material is glossy with $f(\mathbf{x}, \omega_i, \omega_o)$ sensitive to incoming directions, sampling merely the light source can still lead to noisy estimates. This is where sampling the BSDF is a safer strategy. But in cases where we don't know whether the lighting or the material is more crucial, how do we decide which one to sample from?

A defensive mixture distribution is a technique to combine the strengths of different sampling strategies and to make the variance of the estimator more robust [Hes95]. It allows us to create a probability density function (PDF) using the weighted sum of each strategy, in the hope that the mixture one can approximate the integrand better. Specifically, if we have a conservative sampling distribution such as the BSDF $p_f(\omega)$, and an aggressive sampling distribution that approximate the direct illumination $p_{\text{dir}}(\omega)$, then the mixture PDF is constructed as

$$q(\omega) = \alpha\, p_f(\omega) + (1 - \alpha)\, p_{\text{dir}}(\omega) \tag{2.14}$$

where $\alpha \in [0, 1]$ controls the rate of sampling with BSDF. To sample from the mixture distribution, we can use the one-sample *multiple importance sampling* (MIS), which draws a uniform 1D sample $\xi \in [0, 1]$ and compares it with α to decide which lobe to sample from. If $\xi < \alpha$, sample p_f, otherwise p_{dir}.

2.5. Path Guiding

The indirect illumination $L_s^{\text{ind}}(\mathbf{x}, \omega_o)$ plays an important role in rendering. Global illumination effects such as color bleeding and caustics are both examples of indirect lighting. Yet, it remains challenging to compute this component within the framework of path tracing as $L_s^{\text{ind}}(\mathbf{x}, \omega_o)$ is unknown and can only be approximated during path sampling.

Advanced Monte-Carlo-based methods have been developed over the years to improve the calculation of indirect illumination, including bidirectional path tracing, Metropolis light transport [Vea98], and photon mapping, etc. The downside of these methods is the requirement of an intricate understanding of the path space, scene-specific parameter tuning, a fundamental change of the rendering pipeline, or introducing bias that appears as visual artifacts in the rendering results. As a result, these advanced methods work less robustly than a forward path tracer in industrial practice.

Path guiding solves the problem in a different fashion. It aims to enhance the sampling process in eq. (2.11) by gaining knowledge about the scene's lighting progressively, as the rendering goes on. The desired goal of path guiding is to find an efficient sampling PDF q at each point and outgoing direction that resembles the product integrand as much as possible

$$L_s(\mathbf{x}, \omega_o) \approx \frac{f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})}{q(\omega_i \mid \mathbf{x}, \omega_o)} \tag{2.15}$$

where q is called the *guiding distribution*, or *guiding cache*. Path guiding has the potential to significantly reduce the rendering variance if q accurately captures the incoming light distribution at every point and outgoing direction in the scene.

In the ideal case where $q(\omega_i \mid \mathbf{x}, \omega_o) \propto f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})$, the estimator eq. (2.15) gets zero variance. However, this would bring us the complexity of representing a 7D field (3 spatial dimensions + 2 incoming direction dimensions + 2 outgoing direction dimensions). In practice, current path-guiding works try to cache the distribution that is proportional to the incident radiance $L_i(\mathbf{x}, \omega_i)$ instead, which boils down to representing a 5D function $q(\omega_i \mid \mathbf{x})$, a PDF of the incoming direction domain, conditioned on the spatial location of the query.

For the learning of the radiance field $L_i(\mathbf{x}, \omega_i)$ or its distribution, path-guiding methods either rely on the MC samples generated at each vertex along path tracing or undergo a separate training phase where probing samples are emitted from the camera or the light sources.

In practice, people often build a mixture PDF for the actual sampling out of the BSDF sampling strategy p_f and the guiding distribution [VHH+19, RHL20], similar to eq. (2.14)

$$q_s(\omega_i \mid \mathbf{x}) = (1 - \alpha)\, q(\omega_i \mid \mathbf{x}) + \alpha\, p_f(\omega_i \mid \mathbf{x}, \omega_o) \tag{2.16}$$

This defensive handling is helpful in the initial iterations of rendering since the guiding distribution is not mature yet.

Compared to advanced rendering algorithms, path guiding enjoys the modularity of being able to integrate with both forward and bidirectional rendering methods while introducing relatively small changes to the original pipeline. The cost is a modest overhead of sample storage and the training of guiding caches. The simplicity and maintainability make path guiding increasingly suitable for production renderers [VHH+19].

2.6. Spatial Subdivision

To achieve the representation of the 5D guiding field $q(\omega \mid \mathbf{x})$, path-guiding works usually decompose the spatial domain into subspaces and then treat the conditional PDF in each one as a 2D PDF that is purely a function of incoming direction.

Assume X is the full space of the scene. A *spatial subdivision* is defined as a set of disjoint subspaces $\{X_i\}_{i=1}^N$, whose union is the full space:

$$X = \cup_{i=1}^N X_i, \quad X_i \cap X_j = \emptyset, \forall i \neq j \tag{2.17}$$

Obviously, for every point in X, there exists exactly one subspace X_i that contains it. So the 5D guiding distribution simplifies to a somewhat piecewise constant function of $\mathbf{x}$

$$q(\omega \mid \mathbf{x}) = q(\omega \mid X_i), \quad \mathbf{x} \in X_i \tag{2.18}$$

where $q(\omega \mid X_i)$ is the guiding cache for subspace X_i. For brevity, we refer to each subspace after the spatial subdivision as a *region*.

In this section, we will review a few commonly used spatial structures in the world of computer graphics.

2.6.1. Uniform Grid

A *uniform grid* trivially decomposes the space into uniformly sized regular *voxels*. Each voxel is an axis-aligned rectangular cuboid. A uniform grid is parameterized by the resolution or the size of the voxel along each dimension.

In CG applications, we are often interested in the scene parts where geometries exist. Particularly, in a surface rendering application, geometries concentrate on 2D manifolds, a small subset of the 3D space. Nevertheless, applying a uniform grid in such cases will produce a vast number of empty voxels, wasting memory. Moreover, regions with fine geometries have the same size as the empty ones, bearing a low-resolution limit.

2.6.2. Octree

To make grids more adaptive, *octrees* are proposed. An octree partitions the space by constructing a tree-like data structure, where each internal node in the tree corresponds to a (axis-aligned cubic) subspace. An internal node is split into eight descendants by cutting at the middle of the node along the three coordinate planes simultaneously. The octree can rely on some heuristics to trigger the split dynamically across the scene. Once the tree is built, the leaf nodes naturally represent a valid subdivision of the space.

A common heuristic used in ray tracing acceleration is the *count of geometries*, which means a node is split when it contains more geometries (triangles, points, etc) than a user-defined threshold. This way, the spatial subdivision adapts to the density of geometries in the scene. Scene parts containing more geometries will get finer subdivisions. As a result, all regions will have more or less the same number of geometries.

Compared to uniform grids, octrees offer greater adaptivity and require less memory, albeit with the drawback of additional overhead from maintaining the tree data structure.

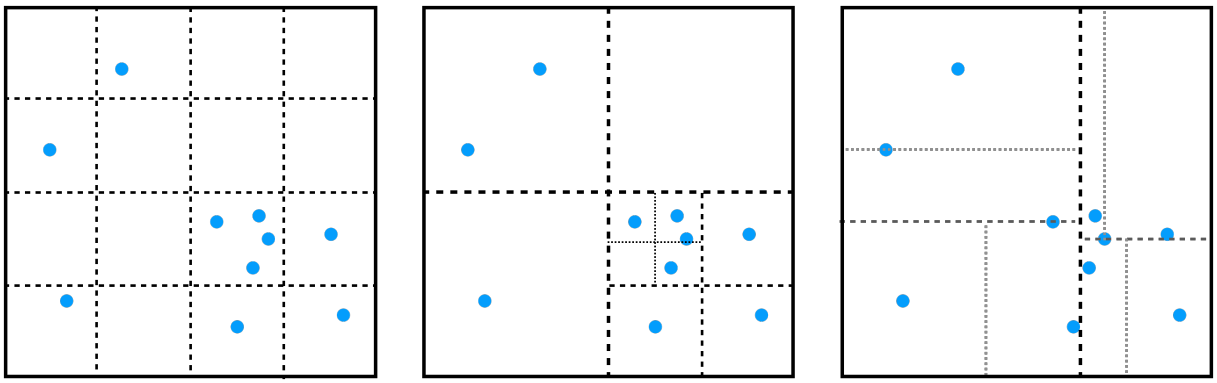


Figure 2.2.: *Illustration of a uniform grid (left), a quadtree (middle), and a k-d tree (right) constructed on top of the same points. The quadtree is the 2D equivalence of an octree. The k-d tree on the right splits at the median point and the longest axis.*

2.6.3. K-d Tree

A *k-dimensional tree* is similar to an octree, except that it splits an internal node into only two descendants by cutting it at some offset along one of the coordinate planes in the k-d space. In this introduction, we refer to the special case where $k = 3$.

Again, k-d trees require heuristics to know when to split and when to stop subdivision such as the count of geometries threshold. On top of that, heuristics are needed to determine which of the coordinate planes to split along (the xy-plane - called *z-cut*, the yz-plane - called *x-cut*, or the zx-plane - called *y-cut*), and where to split. Formally, the algorithm has to determine the axis d and position t of the splitting plane:

$$\mathbf{x}_d = t, \quad d \in \{1, 2, 3\} \tag{2.19}$$

Once a tree is constructed, we define the terminologies in table 2.1 for convenience in future discussions.

Term	**Explanation**
Root node	The top-level tree node.
Internal nodes	Tree nodes with descendants. In particular, a root node is also an internal node.
Leaf nodes	The bottom-most tree nodes, or nodes without descendants.
Depth of a node	Distance from the current node to the root plus one. Specifically, the root node has a depth of 1.
Tree height	The maximum depth of nodes in the tree.

Region	The subspace that the node contains. Apparently, the region should be an axis-aligned rectangular cuboid $X_i = [x_{\min}, x_{\max}) \times [y_{\min}, y_{\max}) \times [z_{\min}, z_{\max})$. Particularly, the union of the leaf node regions should form a valid spatial subdivision of the root region X, which is the bounding box of the whole scene.
Split dimension	The axis d that is perpendicular to the split plane of an internal node.
Spilt position (pivot)	the split position $t \in [(\mathbf{x}_{\min})_d, (\mathbf{x}_{\max})_d]$ of an internal node.
Left child	The left descendant of an internal node i. The left child has the region $X_l = \{\mathbf{x} \in X_i \mid \mathbf{x}_d < t\}$.
Right child	The right descendant of an internal node i. The right child has the region $X_r = \{\mathbf{x} \in X_i \mid \mathbf{x}_d \geqslant t\}$.
(Guiding) cache	The guiding information associated with this tree node.

Table 2.1.: *Terminologies in a k-d tree*

Compared to octrees, k-d trees can produce more balanced data structures due to their binary nature. Moreover, they offer more flexibility in splitting criteria in the sense that they allow us to control both the dimension and location to split at the same time. K-d trees are also simpler to implement and allow faster retrieval of the data with proper optimizations such as treelet reconstruction [KA13].

2.6.4. Bounding Volume Hierarchy

The tree methods mentioned above sections 2.6.2 and 2.6.3 work in a *top-down* fashion, meaning that the subdivision is found by constructing the tree from the root, where during each step, a node is split into its children. Top-down construction may cause issues during initial splits for complex scenes, especially when the geometries are arranged in such a way that any possible axis-aligned cut can intersect some geometry, dividing it into two parts. (For instance, the middle vertical cut in fig. 3.8 intersects with two cuboids.)

Bounding volume hierarchies (BVHs) are a family of spatial data structures typically used in ray tracing acceleration in rendering and collision detection in physics simulation. In our discussion, we **specifically** refer BVH to the ones that construct the tree in a bottom-up (agglomerative) fashion, such as the one described by Gu et al. [GHFB13].

Compared to top-down trees, a BVH works in a converse fashion. It first creates a leaf node (also called a cluster, e.g. an axis-aligned bounding box) that bounds each geometry instance in the scene. Then it iteratively selects the closest pair of clusters, and merges them into a new one that bounds the pair, until all the clusters have been merged into a root node. Such an algorithm

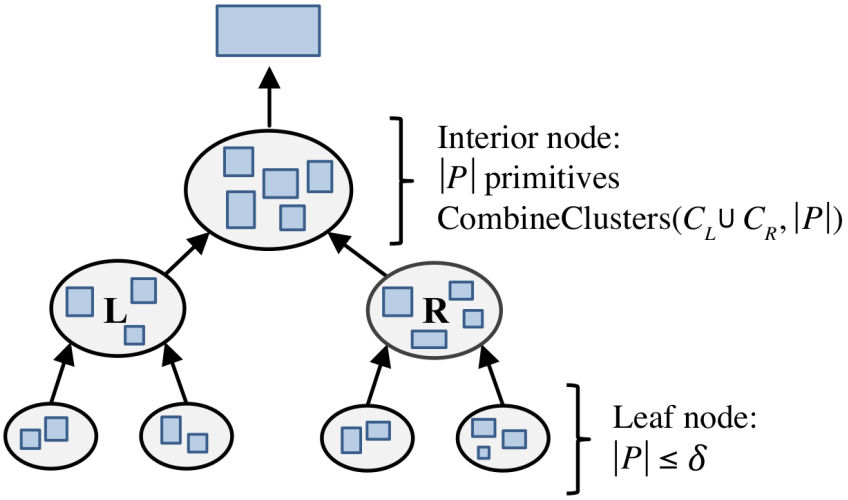


Figure 2.3.: *Illustration of a bottom-up BVH construction algorithm. Source: [GHFB13]*

is computationally expensive (with an $O(N^3)$ time complexity for N geometries). Gu et al. proposed a way to accelerate it by ensuring the leaf node contains at least δ geometries, and limiting the closest-pair-search to local scopes defined by a constraint tree, shown in fig. 2.3. This offers a speed up to $O(N)$ construction, at a cost of a reduced BVH quality. The nearest search and grouping process of BVH naturally avoids the initial split challenge for complex scenes.

Importantly, the result of such a selecting-grouping process is a hierarchy of bounding volumes where each node in the hierarchy descends to a pair of clusters that were used to form the node. The leaf nodes are the bounding volumes of each geometry, whose union does not always cover the whole space (Note there could be gaps or overlaps between nodes). Therefore, strictly speaking, a BVH is not a spatial subdivision eq. (2.17). This issue is discussed in our chapter 7.

2.7. Decision Tree

A *decision tree* is a machine learning method that can predict outcomes given a set of observations, typically used in classification and regression tasks. A decision tree is trained with multi-dimensional discrete or continuous data (features) with ground truth labels (the classification or regression targets). Among all supervised learning methods, decision trees have the advantage of being explainable and simple. They work by building a top-down binary tree where a decision about some dimension is made at each internal node, directing the control flow to a child node. Finally, a leaf node outputs a class label or a constant function value.

We are interested in decision trees because when the input is 3D sample positions, the dimension-selection and decision-making steps coincide with k-d trees, where we also have to choose a split axis and position. But different from k-d trees, decision trees have well-formed criteria to guide the choice of dimension and the position to split based on concepts from information theory. In other words, decision trees are intelligent k-d trees when applied to the spatial subdivision!

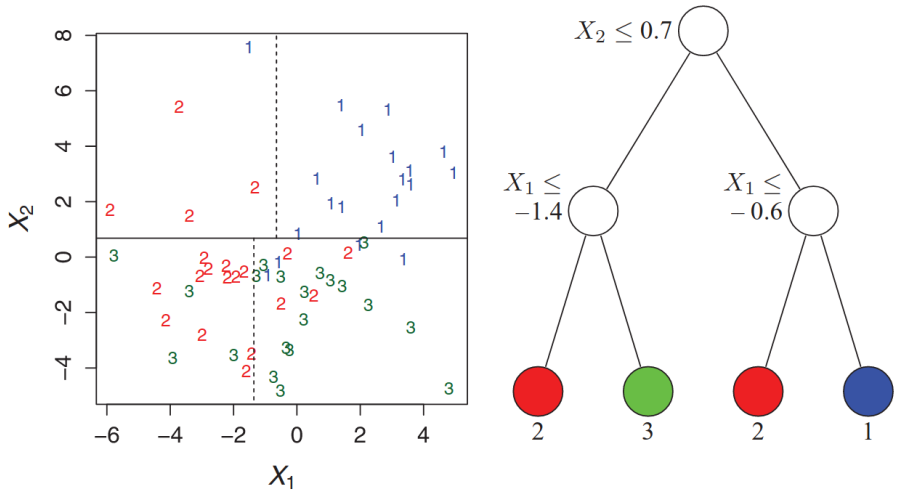


Figure 2.4.: *Illustration of a decision tree trained on 2D position data with three classes. Left: sample position, class, and partitions. Right: the constructed decision tree. Source: [Loh11]*

The next sections recap some widely used decision tree metrics for measuring the "goodness of splitting".

We denote by $S = \{(\mathbf{x}_i, c_i)\}_{i=1}^{N}$ the sample positions and their ground truth labels at an internal node with region X, where $\mathbf{x}_i = (x_i, y_i, z_i) \in X,\ c_i \in \{1, \ldots, K\}$.

We represent by (d, t) the proposal split dimension and position, $d \in \{1, 2, 3\},\ t \in [(\mathbf{x}_{\max})_d, (\mathbf{x}_{\min})_d]$. The split results in the left and right child regions $X_l = \{\mathbf{x} \in X \mid \mathbf{x}_d < t\},\ X_r = \{\mathbf{x} \in X \mid \mathbf{x}_d \geqslant t\}$, and corresponding samples $S_l = \{(\mathbf{x}, c) \in S \mid \mathbf{x}_d < t\},\ S_r = \{(\mathbf{x}, c) \in S \mid \mathbf{x}_d \geqslant t\}$.

The metrics below are all linked with the *homogeneity* of samples.

2.7.1. Entropy

Shannon entropy quantifies the average level of impurity or uncertainty about a set of samples.

The entropy of a node is defined as

$$H(S) = H(p_1, \ldots, p_K) \coloneqq -\sum_{i=1}^{K} p_i \log_2 p_i \tag{2.20}$$

where $p_i = \dfrac{|\{(\mathbf{x}, c) \in S \mid c = i\}|}{|S|}$ is the probability of finding class i in the samples within the node.

When all the K probabilities equal to $\dfrac{1}{K}$, the entropy reaches its maximum at $\log_2 K$. When the probability vector $(p_1, \ldots, p_K)$ is one-hot, the entropy is minimized at 0.

2.7.2. Information Gain

Intuitively, the more reduction in "uncertainty" or "lack of information", the better a split is. The *information gain* is defined as the difference of the entropy when going from one node to its descendant a ($a \in \{l, r\}$)

$$\mathrm{I_g}(S, a) \coloneqq H(S) - H(S_a) \tag{2.21}$$

We are interested in the expected information gain (i.e. *mutual information*), which is the average of $\mathrm{I_g}$ over left and right children

$$\mathbb{E}_a\left[\mathrm{I_g}(S, a)\right] = H(S) - (p_l H(S_l) + p_r H(S_r)) \tag{2.22}$$

$$=: \mathrm{IG}(S, d, t) \tag{2.23}$$

where $p_l = \dfrac{|S_l|}{|S|}$, $p_r = \dfrac{|S_r|}{|S|}$ are the fraction of the left and right samples. S_l, S_r, p_l, p_r implicitly depends on d, t. That's why the mutual information IG is a function of d, t.

The best split is defined as the one that maximizes the mutual information.

$$(d, t)^* = \arg\min_{d,t} \mathrm{IG}(S, d, t) \tag{2.24}$$

2.7.3. Gini Impurity

Gini impurity is another decision tree metric. It is associated with the "lack of information" in physics, defined as

$$\mathrm{Gini}(S) = \mathrm{Gini}(p_1, \ldots, p_K) \coloneqq \sum_{i=1}^{K} p_i(1 - p_i) = 1 - \sum_{i=1}^{K} p_i^2 \tag{2.25}$$

It's maximized at $1 - \dfrac{1}{K}$ when all probabilities equal $\dfrac{1}{K}$, and minimized at 0 for one-hot cases.

Gini impurity can replace the entropy terms in eqs. (2.21) and (2.23) to serve as an alternative splitting criterion.

2.7.4. Variance Reduction

When the target of the input is continuous rather than discrete class labels (called *regression trees*), variance is often used to quantify the dispersion of the data. Let $w_i \in \mathbb{R}$ be the target data of sample i.

The empirical target variance of $S = \{(\mathbf{x}_i, w_i)\}_{i=1}^N$ is defined as

$$V(S) \coloneqq \frac{1}{N} \sum_{i=1}^{N} (w_i - \bar{w})^2 \tag{2.26}$$

$$\bar{w} = \frac{1}{N} \sum_{i=1}^{N} w_i \tag{2.27}$$

The variance reduction of a node led by the split is

$$\mathrm{I_v}(S, d, t) \coloneqq V(S) - (p_l V(S_l) + p_r V(S_r)) \tag{2.28}$$

which implies the best split

$$(d, t)^* = \arg\min_{d,t} \mathrm{I_v}(S, d, t) \tag{2.29}$$

Our proposed spatial subdivision algorithms take a lot of inspiration from the concepts introduced above. In chapters 5 and 6, we will revisit some of these points and make adaptations for path guiding.

3

Related Work

In this chapter, we first review a few of the most related path-guiding works, with a focus on spatial subdivisions and how they represent the guiding distribution. We will trace the advancements of spatial subdivision, showing how each work builds upon precedent ones.

We then identify several failure cases that persist in state-of-the-art approaches, highlighting problems that need to be solved, which contextualizes and motivates the development of our proposed algorithms in chapter 5.

3.1. Path Guiding without Spatial Subdivision

Earlier works [Jen95, SL06, BAJ08] proposed a two-pass algorithm, similar to photon mapping. In the first pass, photon samples are sent to the scene to probe the incident radiance at discrete points and directions $\Delta\Phi(\mathbf{x}^{(j)}, \omega_i^{(j)})$. In the second pass, paths are traced from the camera and importance-sampled using the photon information from the first pass. Specifically, k nearby photons around a query point $\mathbf{x}$ are retrieved to form a directional histogram or mixture model $q(\omega_i^{(j)} \mid \mathbf{x})$ for sampling.

Vorba et al. [VKŠ+14] proposed fitting Gaussian mixtures from photon and importon maps for guiding. They reduce memory usage and computation by caching and reusing existing distributions if one is available near the query location.

Dahm et al. [DK17] noticed the structural similarity between LTE and the equations from reinforcement learning, and thus applied Q-learning to importance sample the light transport. To approximate the Q function, they discretize it with hemispheres storing at surface points exhibiting a 2D low-discrepancy sequence.

These methods involve no spatial subdivision. A core problem is the creation, caching, and fast

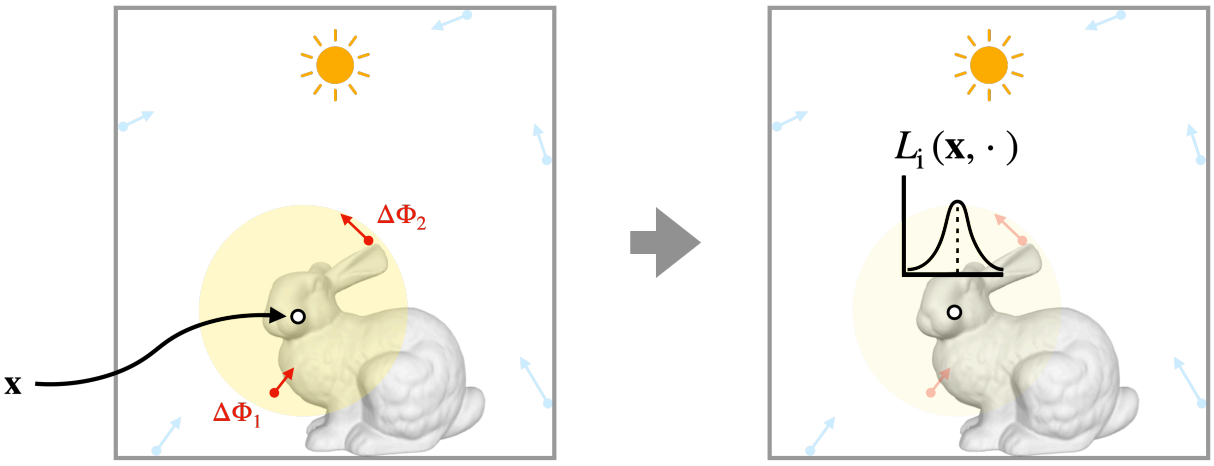


Figure 3.1.: *Path Guiding without Spatial Subdivision. To sample the incident direction at position* $\mathbf{x}$*, neighboring photons are retrieved to construct a representation of the directional radiance distribution* $L_i(\mathbf{x}, \cdot)$

retrieval of the guiding distributions. fig. 3.1 provides a general illustration of the cache query workflow.

3.2. Practical Path Guiding

Muller et al. [MGN17] presented a simple yet practical path-guiding algorithm (PPG). The high-level workflow is an iterative routine that combines rendering and training of the cache into one. They collect guiding samples $\{s_i\}$ (positions, incident directions, and incident radiance estimates: $s_i = (\mathbf{x}_i, \omega_i, \langle L_i(\mathbf{x}_i, \omega_i)\rangle)$) at all path vertices. Then, they use $\{s_i\}$ to update the guiding cache $q(\omega \mid \mathbf{x})$, represented as a 5D spatial-directional tree (*SD-tree*), as shown in fig. 3.2.

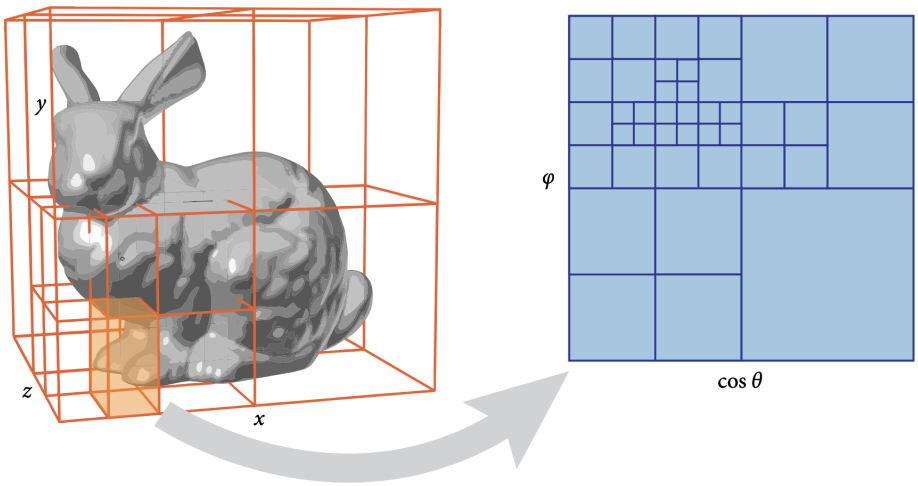


Figure 3.2.: *PPG's subdivision scheme. Source: [VHH+19]*

The spatial subdivision is done with a k-d tree ($k = 3$). The split dimension alternates among x, y, z in a **round-robin** manner (i.e. $x, y, z, x, \dots$), while the split location is set to the **middle point** of the current node. Whether to split is driven by the **count of samples** accumulated in the node.

At a spatial leaf node (a region), a quadtree is created to represent the directional distribution of incident radiance marginalized in that region. The quadtree is built over the world-space cylindrical coordinate domain $(\phi, \cos\theta)$.

The middle-split-heuristics can generate relatively balanced spatial trees for the case when guiding samples distribute uniformly on scene geometries. But when there is a discrepancy in the spatial distribution of the samples, which is usually encountered in practice, splitting at the middle point of a node can cause fluctuation in the number of samples between the child nodes. In extreme cases, one child can have zero samples whereas the other gets all of them (such as the bottom left and the highlighted node in fig. 3.2).

The round-robin split dimension heuristics exhibit a problem when the scene bounding box is not cubic. In a 2D example fig. 3.3, after a few splits, the leaf nodes of the k-d tree have the same aspect ratio as the scene bounding box. Their elongated boundaries make it hard to adapt to the shape of the geometries. PPG remedies this issue by enlarging the scene bounding box to a cube so that after multiples-of-three splits, leaf regions remain cubic and can adapt better to the scene geometries. The downside is that it produces more empty regions.

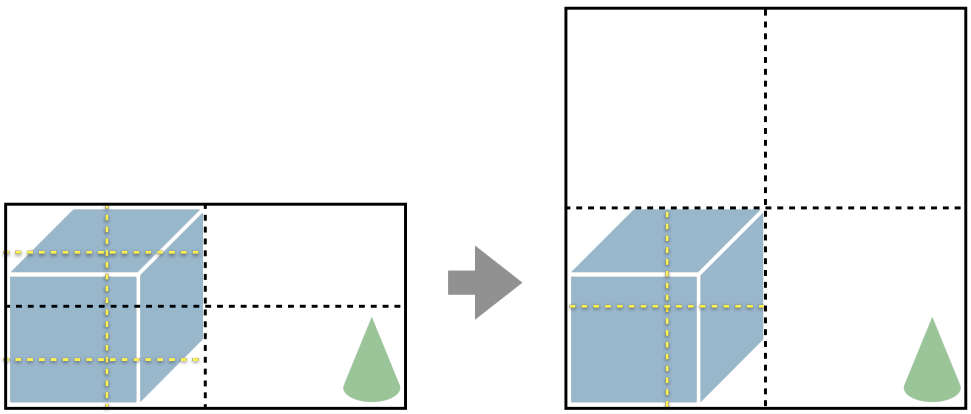


Figure 3.3.: *Illustration of PPG's bounding box extension handling. Left: the middle-round-robin split heuristics will result in rectangular regions when the scene bounding box is not perfectly cubic, which does not adapt to the shape of the geometries well. Right: by extending the bounding box to a cube, the partition adapts better to the geometries.*

3.3. Robust Fitting of Parallax-aware Mixtures for Path Guiding

Ruppert et al. [RHL20] presented one of the leading works in path guiding. Their main improvement over PPG is the replacement of the directional quadtree at a spatial leaf node with a parametric mixture model, along with various optimizations for its fitting.

VMM. They employ the *von Mises-Fisher Model* (VMM) to represent the guiding distribution in a region, based on the von Mises-Fisher distribution function (vMF).

A vMF is a 3D Gaussian distribution with mean μ and isotropic covariance $\kappa^{-1}I$, conditioned on the 2D unit sphere

$$v(\omega \mid \mu, \kappa) \coloneqq \frac{\kappa}{2\pi(e^{\kappa} - e^{-\kappa})} \exp\left(\kappa \mu^T \omega\right) \tag{3.1}$$

The VMM is defined as a weighted sum mixture distribution of vMFs

$$\mathcal{V}(\omega \mid \Theta) \coloneqq \sum_{k=1}^{K} \pi_k v\left(\omega \mid \Theta_k\right) \tag{3.2}$$

where $\Theta_k = \{\mu_k, \kappa_k\}$ are the parameters for each component. Weights π_k are positive and sum up to one. Ruppert et al. argue that VMMs have the potential to represent varying numbers of lighting components. Each vMF can represent the incoming radiance distribution of one (virtual) light source. vMFs also offer the flexibility of approximately integrating with the cosine and BSDF term to enable guiding the full product of LTE eq. (2.2).

Fitting. To learn the guiding distribution, they present a weighted EM algorithm to fit VMM parameters. Their maximum a posteriori probability (MAP) estimate allows for incremental updates with a small batch of samples. This enables the proposal of an interleaving rendering and training process. During the rendering phase (several pixel samples), at vertices of the MC random walk, samples $\{s_i\} = \{(\mathbf{x}_i, \omega_i, p(\omega_i \mid \mathbf{x}_i), \langle L_i(\mathbf{x}_i, \omega_i)\rangle)\}$ are generated and collected into a buffer. During the training phase, the guider traverses the spatial k-d tree with the sample buffer and fits or updates the VMM stored in each region. The faster turn-around between rendering and guiding offers a smaller memory footprint.

Variance-aware split and merge. They further improve the accuracy and robustness of fitting with an adaptive split and merge approach. This allows them to select the suitable number of vMF components for different local regions in the scene with different lighting complexity. The split and merge criteria are based on the *Pearson χ^2-divergence*, which reflects the *normalized estimator variance* (NEV) of sampling with PDF $q(\omega)$ instead of the ground truth $p(\omega) \propto L_i(\mathbf{x}, \omega)$.

$$D_{\chi}^2(p \parallel q) \coloneqq \int \frac{(p(\omega) - q(\omega))^2}{q(\omega)} \mathrm{d}\omega = \mathrm{Var}\left[\frac{\langle \Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\right] \tag{3.3}$$

where the fluence estimator $\langle \Phi(\mathbf{x})\rangle = \langle L_i(\mathbf{x}, \omega)\rangle / p(\omega \mid \mathbf{x})$ is used as the sample weight for *expectation-maximization* (EM) fitting. eq. (3.3) can be estimated using MC integration with sample data.

They detect the χ^2-divergence of every component. A split is triggered if the divergence exceeds a user-defined threshold, which hopefully increases the expressiveness of the model. Merging, on the other hand, is designed to avoid overfitting. They compute pair-wise merging cost based on χ^2-divergence of the original vMF pair and merged vMF. They merge the pair with the lowest cost if it goes below the threshold.

We take some inspiration from their adaptive split and merge in the directional domain while developing our spatial subdivision approaches.

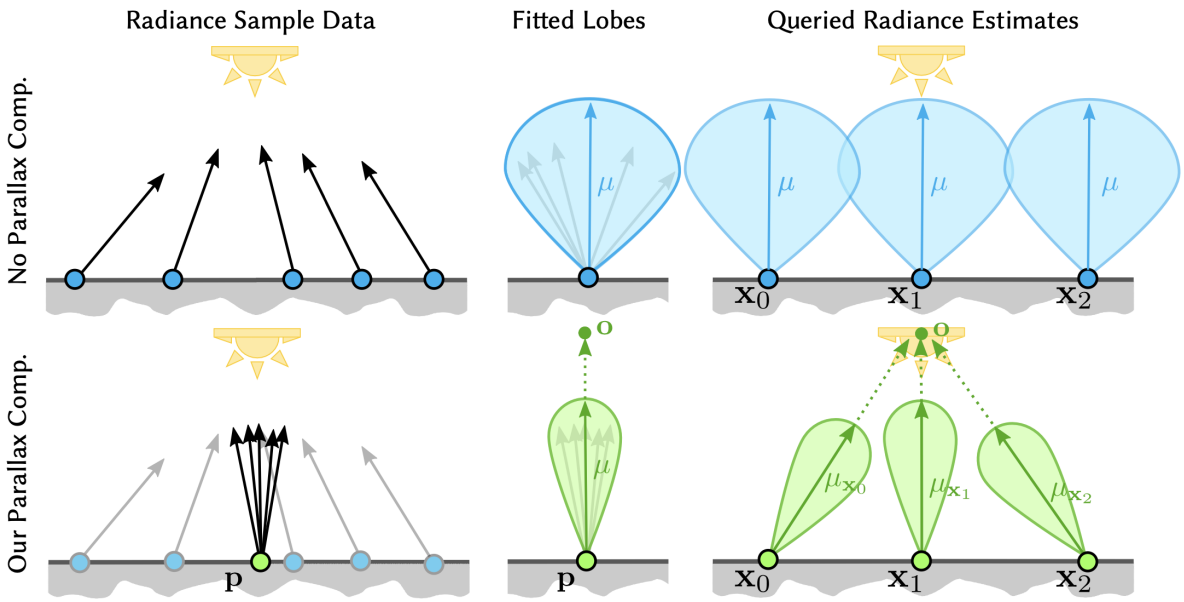


Figure 3.4.: *Parallax effect is compensated by re-adjusting the vMF towards its light source. Source: [RHL20]*

Parallax compensation. They notice that within a region, assuming a single point light source, the guiding samples' directions exhibit shifts in response to their position change, which introduces an extra dispersion in the learned signals. By re-projecting the samples to the region center during training, and re-adjusting the direction of the vMF lobe towards the light source, the distributions can better represent the actual lighting, as shown in fig. 3.4.

Spatial subdivision. Similar to PPG, they leverage a k-d tree to subdivide the space into axis-aligned rectangular regions. Instead of splitting at the middle point with a round-robin dimension selection, they collect the mean and variance statistics of the sample positions $\mathbf{x}_i$, split at the **mean position**, and choose the **axis with the largest variance**. To ensure the child regions after the split get enough valid samples, they only collect the statistics of non-zero-weight samples (sample weight $\langle \Phi(\mathbf{x}_i) \rangle = \langle L_i(\mathbf{x}_i, \omega_i) \rangle / p(\omega_i \mid \mathbf{x}_i)$).

Ideally, splitting at the median point of the samples will ensure a perfectly balanced tree. However, due to the fact that there is no cheap way of incremental median update, they resort to using the mean, which in practice results in fairly balanced trees, and avoids PPG's issue of having no samples for one of the children.

Splitting at the axis with the largest variance, on the other hand, maintains a high cubicality of the resulting region boundaries. Intuitively, the axis with the largest variance has the longest span of valid samples.

3.4. Neural Path Guiding

With the surge in artificial intelligence, more and more methods based on neural networks and deep learning have come up in the field of path guiding.

Muller et al. [MMR+19] and Zheng et al. [ZZ19] opened the avenue of applying deep learning methods to improve MC integration in rendering. Muller et al. presented a method to guide path tracing by sampling a neural network that encodes the incident radiance distribution of the whole scene. They train the network by minimizing *Kullback–Leibler divergence* (KL divergence) or χ^2-divergence between the sampling distribution and the ground truth lighting distribution. Zheng et al. represent and learn the density of MC samples in the *primary sample space* of the rendering by a neural network and use it to effectively guide the sampling.

Notably, these works do not require spatial subdivision at all since a single network is trained for the whole scene.

Zhu et al. [ZXS+21] proposed a photon-driven guiding method analogous to that of Jensen et al. [Jen95]. They perform spatial subdivisions to allow the reuse of incident radiance histograms created from local photons in each region. Furthermore, they feed the histograms through an offline-trained neural network to decode a sampling distribution for better quality guiding.

Remarkably, they subdivide the scene using a *hierarchical grid*. The grid is initialized as a uniform one as introduced in section 2.6.1. As photons come in, valid voxels in the grid (those with camera path vertices) are iteratively subdivided into local binary trees. For the split handling, they propose a hybrid heuristics that involves splitting at either the **median or middle** point, while choosing the dimension in the **round-robin** manner.

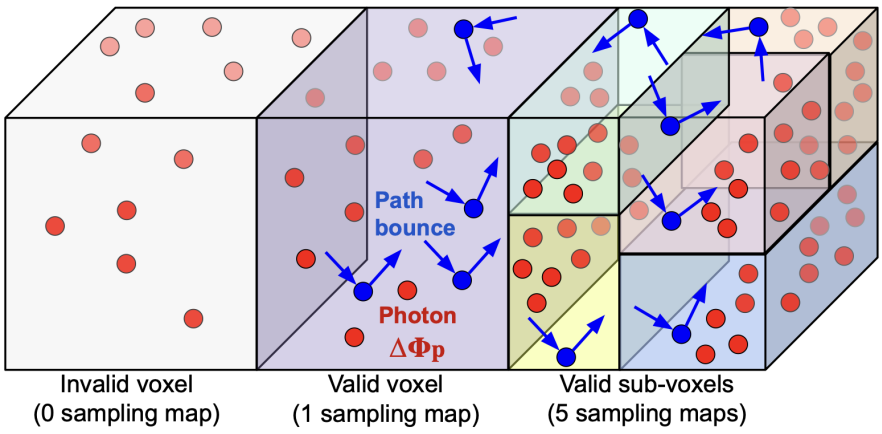


Figure 3.5.: *The hierarchical grid spatial structure to cache sampling maps. Valid voxels are subdivided into a binary tree based on local photon statistics. Source: [ZXS+21]*

Instant Neural Graphics Primitives (Instant NGP) [MESK22] presents a **multi-resolution hash grid** data structure (fig. 3.6) for a diverse set of neural graphics applications. The core is to distribute parametric encodings of the neural network into spatial hash tables with different scales. They argue that coarser scales can store more global information while finer ones store more local ones. The advantages of a multi-resolution hash grid include its simplicity, high performance, and GPU-friendliness. For hash conflicts, they rely on the neural network for

resolution.

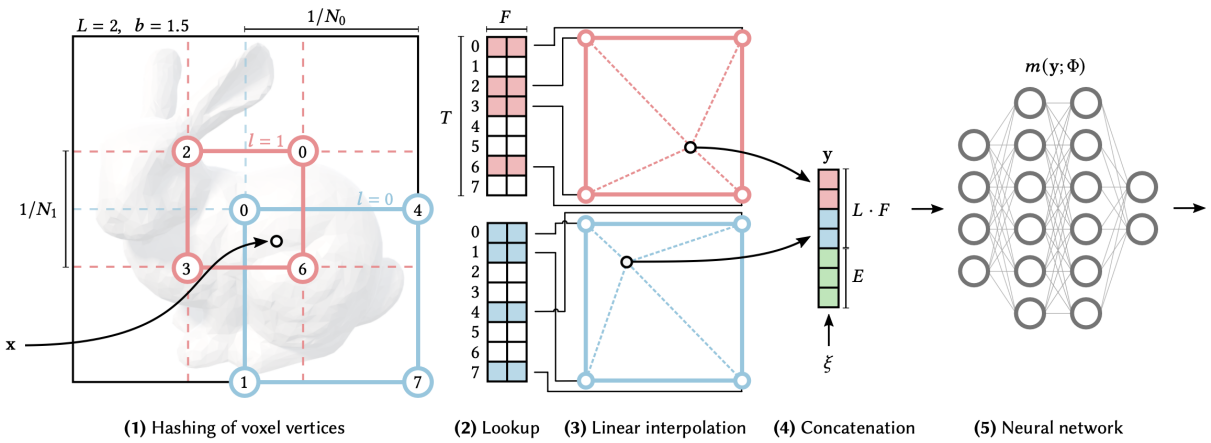


Figure 3.6.: *Multi-resolution hash grids for neural graphics primitives. Source: [MESK22]*

Neural Parametric Mixtures for Path Guiding (NPM) [DWL23] presents a path-guiding approach that combines the vMF guiding section 3.3 and neural networks. They implicitly store the global and local guiding cache parameters in a multi-resolution hash grid. Then they concatenate and decode the spatial encodings through a multi-layer perceptron (MLP) to obtain VMM parameters for path guiding. Similar to [MMR+19], they train the network on the fly by flowing the gradient of KL divergence estimates.

Conceptually, this work subdivides the scene using a set of uniform grids with different scales, where each region indirectly stores the parameters of a directional model at different frequencies.

In this thesis, we focus on non-neural guiding methods. Despite the simplicity of hash grids, or uniform grids, they necessitate setting the grid resolution parameter in advance, which requires scene-specific tuning and cannot be refined during rendering. Uniform-sized voxels also make adapting to local lighting contexts challenging. Employing a multi-scale uniform grid is non-trivial because the number of scales adds one more parameter for tuning. Moreover, currently, there is no available method to distribute VMM parameters across regions with different scales without the help of a neural network.

Therefore, our primary approach will be based on a k-d tree spatial subdivision with directional VMMs. We believe in k-d trees' potential because of their adaptivity to valid sample distributions. They offer flexibility in dimension selection, split position control, and incremental refinement. Additionally, a k-d-tree-like subdivision is highly compatible with a cutting-edge path-guiding implementation (see section 4.1).

3.5. Remaining Problems with Spatial Subdivision

Albeit the improvements in the state-of-the-art spatial subdivision made by Ruppert et al. (section 3.3), problematic cases remain. We demonstrate some typical ones in figs. 3.7 to 3.10 with a summary of observed issues into four categories. This helps inspire the considerations and development of our proposed methods.

Not lighting-aware. The spatial change of actual incident radiance is not taken into account when proposing the splitting position. For instance, the red squares in fig. 3.7 and fig. 3.9 highlight regions where there is a transition from shadow to bright areas. Using only one guiding distribution will mess up the bright part, producing extra rendering errors.

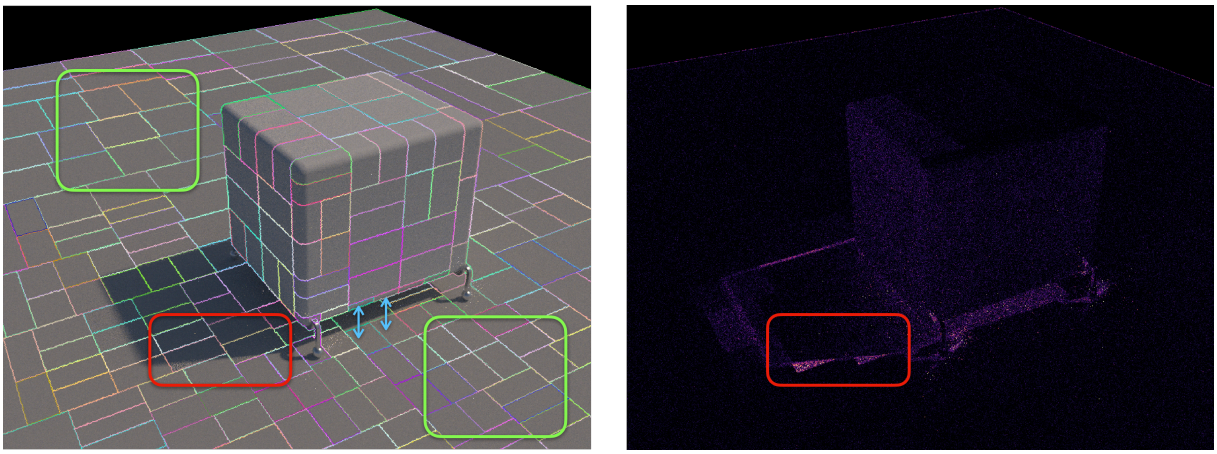


Figure 3.7.: *Example problems of a spatial subdivision in the Cube scene. Left: rendering image overlaid with guiding region boundaries. Right: error map.*

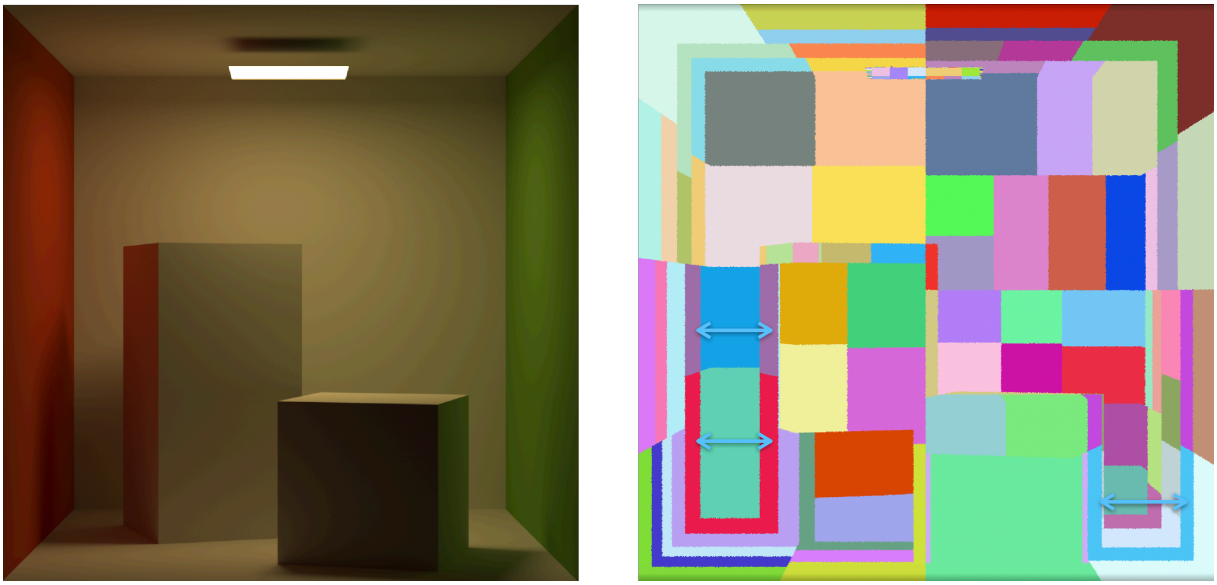


Figure 3.8.: *Example of a spatial subdivision not being fully aware of the underlying geometry (Cornell Box). Left: reference image. Right: visualization of guiding region ID.*

Not geometry-aware. In general, the incident radiance varies significantly at opposing geometric objects because they are either far apart or have different hemispheres[1]. Existing methods tend to use separate guiding caches for distant points, but they do not consider the geometries on which the points lie. In figs. 3.7 to 3.9, the surface pairs highlighted with blue arrows have very different lighting distributions, but are wrongly grouped into one guiding region.

Over-refinement. A high-resolution guiding cache is used for uniformly-lit areas, reducing the number of training samples each region can access, and incurring more memory overhead. Typical cases are highlighted with green squares in fig. 3.7 and fig. 3.9, where the optimal way would be to use one guiding region for the whole bright part of the plane since it is uniformly lit by the sky dome.

Insufficient refinement. A low-resolution guiding cache is used for areas with drastic light changes, limiting the expressiveness of the model. A good case in point is the *Water Caustics* scene fig. 3.10, which exhibits drastically changing lighting brought by the caustic effects. A very coarse subdivision such as the image on the left, resulting from a high sample count threshold, will mix signals from different parts of the caustics together, making it challenging for the guider to summarize useful information. A fix would be to increase the resolution such as the image on the right.

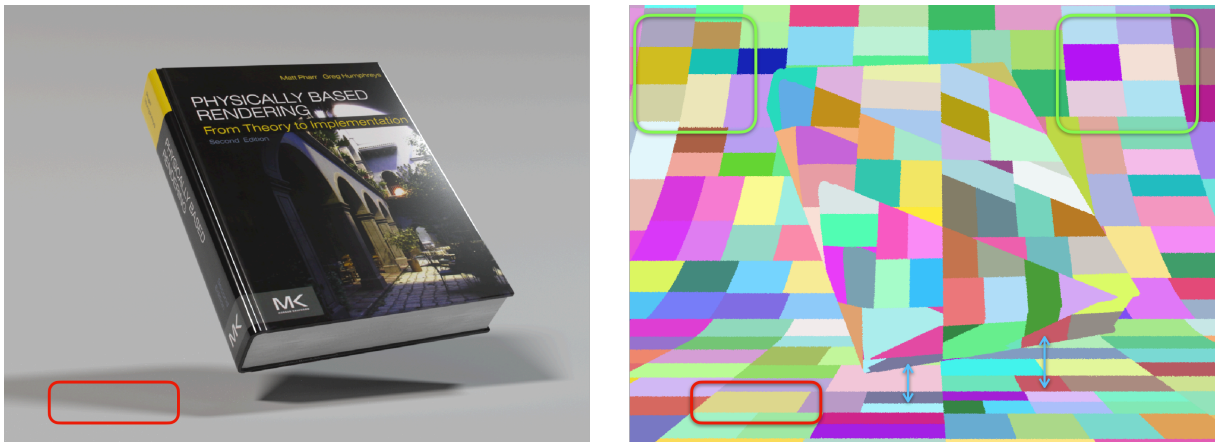


Figure 3.9.: *Example problems of a spatial subdivision in the PBRT Book scene. Left: reference image. Right: guiding region ID.*

[1]For detailed discussions, see section 5.2.1

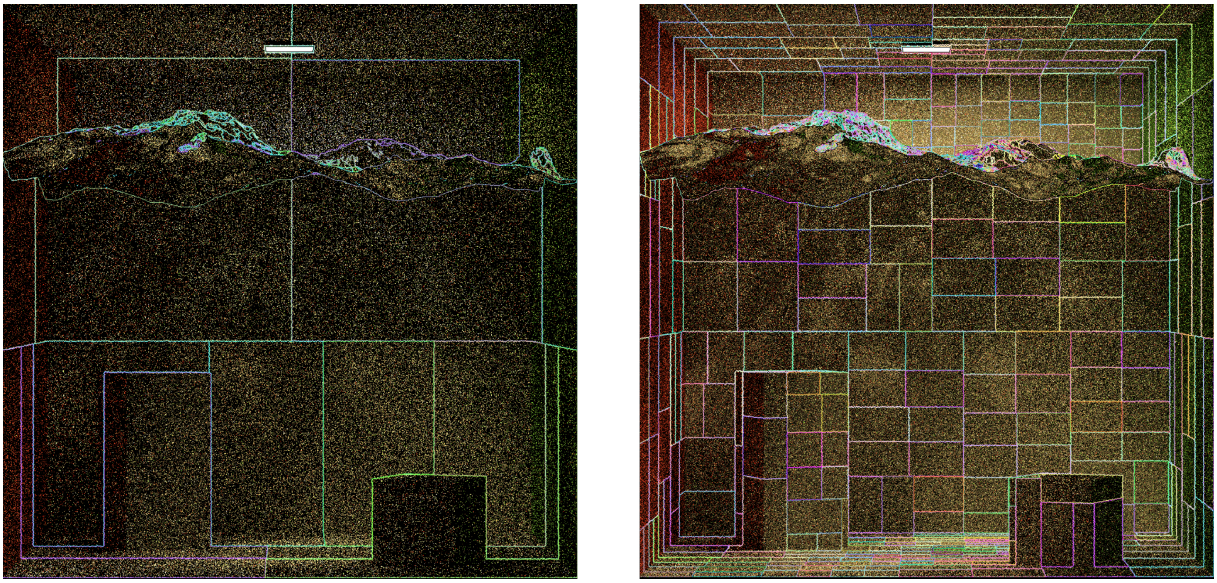


Figure 3.10.: *An example of insufficient refinement on the Water Caustics scene (32 samples per pixel). Left: guided with a lower-resolution cache. Right: guided with a higher-resolution cache.*

4

Context and Motivation

To begin with, we provide context to clarify the aspect of path guiding that our research focuses on.

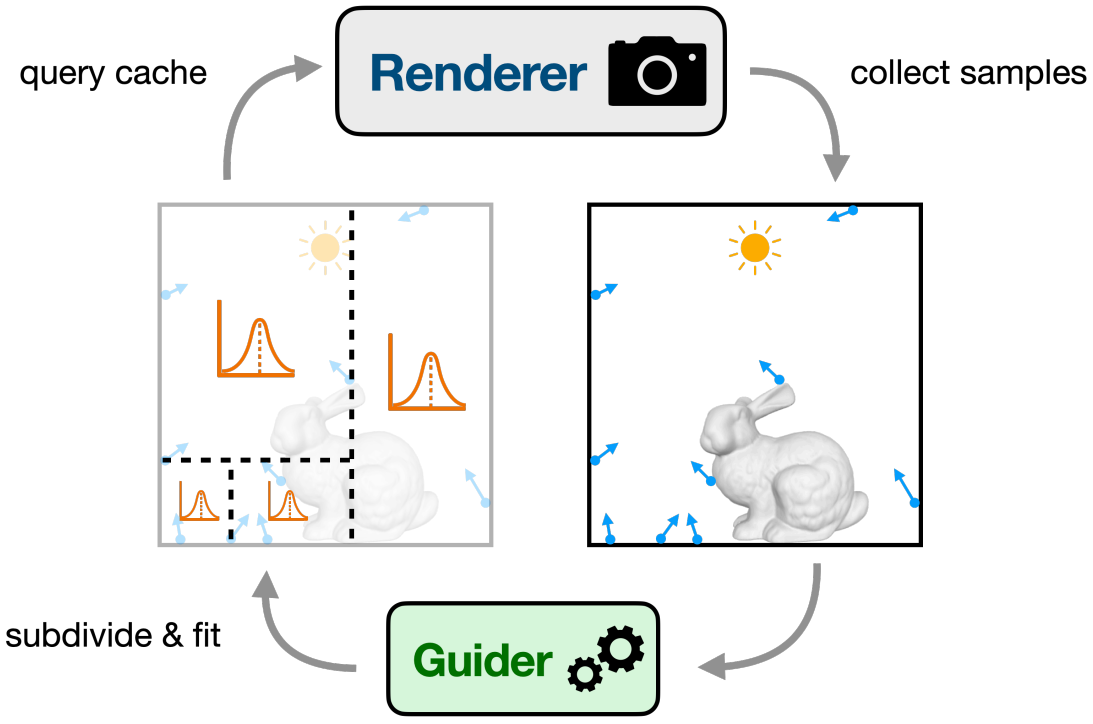


Figure 4.1.: *The collaboration between the renderer and the guider. During each rendering phase, the renderer queries the guider to sample ray directions and compute PDFs. In the training phase, the newly collected samples are sent to the guider to store or update the guiding cache.*

4.1. Open PGL

Our central work is based on *Intel Open Path Guiding Library* (Open PGL) [HD22]. It implements a wide set of state-of-the-art path-guiding algorithms [MGN17, RHL20].

The spatial subdivision part of Open PGL adopts the k-d tree described in section 3.3, which is recapped in section 4.2.

In our discussion, we assume the high-level rendering pipeline to be the way fig. 4.1 illustrates, which aligns with Ruppert et al.'s work described in section 3.3.

During the *rendering phase*, guiding samples generated at every path vertex are collected into a list

$$S = \{s_i\}_{i=1}^{N} \tag{4.1}$$

$$s_i = (\mathbf{x}_i, \omega_i, q_s(\omega_i \mid \mathbf{x}_i), \langle L_i(\mathbf{x}_i, \omega_i)\rangle) \tag{4.2}$$

$q_s(\omega_i \mid \mathbf{x}_i)$ is the sampling PDF of the incoming direction ω_i at location $\mathbf{x}_i$. $\langle L_i(\mathbf{x}_i, \omega_i)\rangle$ is the noisy incident radiance obtained from path tracing. The Monte Carlo weight is a derived quantity

$$\langle \Phi(\mathbf{x_i})\rangle \coloneqq \frac{\langle L_i(\mathbf{x}_i, \omega_i)\rangle}{q_s(\omega_i \mid \mathbf{x}_i)} \tag{4.3}$$

It is also known as the *fluence estimator*, because it serves as an MC estimate for the *fluence*

$$\Phi(\mathbf{x}) \coloneqq \int L_i(\mathbf{x}, \omega)\mathrm{d}\omega \tag{4.4}$$

Entering the *training phase*, Open PGL first subdivides (refines) the scene into local guiding regions. Then it fits (updates) the the directional cache with incoming samples S per region.

We mainly focus on the subdivision step of the training phase.

4.2. Problem Definition

A spatial subdivision algorithm takes a list of sample data (positions, directions, weights, etc. eq. (4.2)) as input, and outputs a subdivision of 3D space, whose formal definition is given in section 2.6.

As we have shown in chapter 3, the state-of-the-art path guiders employ k-d trees to partition the space due to its simplicity, lookup efficiency, and freedom in the split criteria. For these reasons, we base our proposed algorithms in this chapter and chapter 5 on k-d trees.

The behavior of a k-d tree is governed by three *sub-decisions* at a leaf node:

1. Should this node be further split to refine the partition?
2. If yes, which dimension should be chosen?
3. Where should the splitting occur?

PPG's strategy. For instance, the spatial binary tree from the work of Muller et al. (section 3.2) is essentially a k-d tree with the split heuristics below:

1. Split when this node has accumulated more than a certain amount of samples. This fixed upper bound is called the *maximum sample count threshold*.
2. Through the dimensions for splitting in a *round-robin* fashion. I.e. $x, y, z, x, \ldots$.
3. Split at the middle point of the region bounds.

Open PGL's strategy. The k-d tree implementation by Ruppert et al. (section 3.3) and Open PGL adopts the following strategies:

1. Split with the maximum sample count threshold, except that they only count in *valid* (non-zero-weight) samples[1].
2. Choose the dimension with the largest variance in the valid sample positions.
3. Split at the mean position of valid samples along the chosen dimension.

Both strategies can yield suboptimal guiding results, which we already showed in sections 3.2 and 3.5. This drives us to consider what makes a "good" spatial subdivision.

4.3. Motivational Examples

To familiarize ourselves with the problem, let's discuss the optimal split strategy in a low dimensional setup. We have the following ideal assumptions:

1. The incident radiance field $L(x, \omega) : D \to \mathbb{R}_{\geqslant 0}$ is a 2D function over the product of a 1D spatial domain and a 1D angular domain $D = X \times \Omega = [x_{\min}, x_{\max}] \times [\omega_{\min}, \omega_{\max}]$. Hence, there is no need to select the split dimension.
2. The rendering is noise-free. i.e. we can directly evaluate $L(x, \omega)$ with no noise.
3. The guiding distribution q on a tree node is constant to x. i.e. $q(\omega \mid x \in X) = q(\omega)$.
4. The subdivision is greedy, meaning that we do not consider future splits in the children (and in the grandchildren etc.) for optimality.

The goal of rendering is to estimate the fluence at each point in space by MC integration with a guiding distribution

$$\Phi(x) = \int_{\Omega} L(x, \omega) \mathrm{d}\omega \tag{4.5}$$

$$\approx \langle \Phi(x) \rangle = \frac{L(x, \omega)}{q(\omega \mid x)} \tag{4.6}$$

[1]The reason for storing only valid samples is explained in section 3.3

Consider a radiance field shown in fig. 4.2, where

$$L(x, \omega) = 1 + \cos(\varphi(2\omega + \varphi(20x))) \tag{4.7}$$

$$\varphi(t) = \begin{cases} -\pi & t < -\pi \\ \pi & t > \pi \\ t & \text{otherwise} \end{cases} \tag{4.8}$$

$$x \in [-1, 1],\ \omega \in [-\pi, \pi]$$

It describes a spatially shifting single-cosine-lobe illumination. In particular, the peak direction

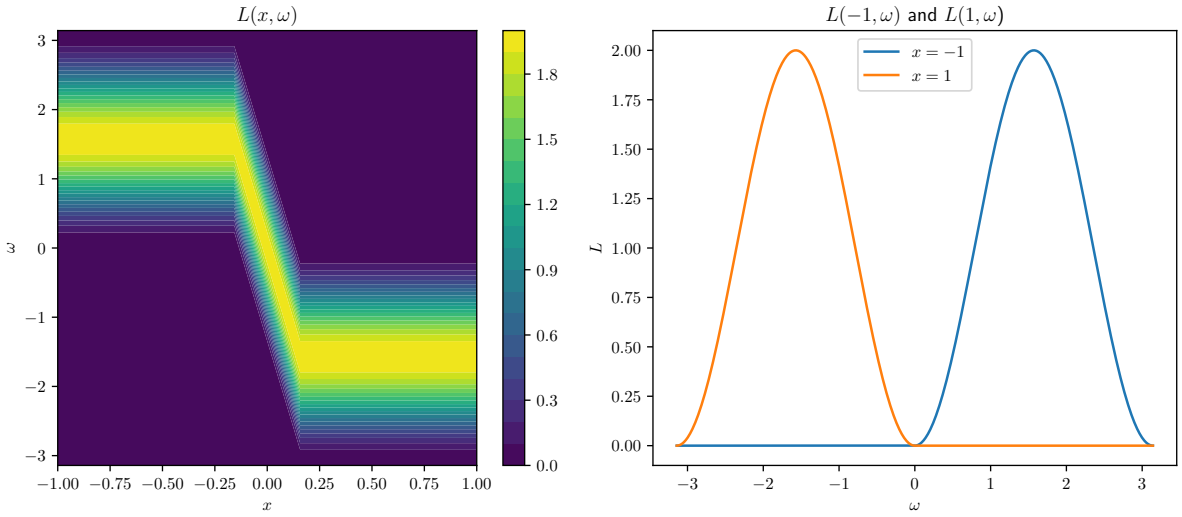


Figure 4.2.: *A 2D radiance field setup (1). Left: contour plot of the radiance field (spatial $\times$ directional, not to be confused with an $x \times y$ spatial plot). Right: 1D directional radiance at two spatial locations.*

shifts from $\pi/2$ to $-\pi/2$, as x changes from $-\pi/20$ to $\pi/20$.

The ground truth fluence is $\Phi(x) = \int_{-\pi}^{\pi} L(x, \omega) \mathrm{d}\omega = \pi$, from which we can find the *ground truth distribution*

$$p(\omega \mid x) \coloneqq \frac{L(x, \omega)}{\Phi(x)} \tag{4.9}$$

Theoretically, if we can set the guiding distribution to p, the rendering variance $\mathrm{Var}[\langle \Phi(x) \rangle]$ is immediately 0. However, due to the third assumption, we are hindered by the spatial constancy of q and the rendering variance is always positive in this case. This encourages us to split the node into two, allowing each child region to have a separate guiding distribution. It is hoped that the combined distributions of the children can provide a better approximation of p.

Concerning "where to split" optimally, one might propose the middle point $x = 0$, as this placement minimizes the *total spatial variation* of the signal in the left and right half-spaces. This reduction in variation facilitates the training of the guiding distributions for the child regions holistically.

Nevertheless, how to quantify "spatial variation" and "ease of training"? How can we demonstrate that splitting at the midpoint is indeed the best?

4.3.1. Quality of a Guiding Distribution

Since our ultimate goal of subdivision and training is to achieve low rendering error, we use the *normalized estimator variance* (NEV)[2] to measure the performance of a guiding distribution $q(\omega)$, which is defined as the spatially marginalized relative variance of the fluence estimator:

$$\mathrm{NEV}(q) \coloneqq \mathbb{E}_{x\sim\hat{p}_X}\mathrm{Var}_{\omega\sim q}\left[\frac{\langle\Phi(x)\rangle}{\Phi(x)}\right] \tag{4.10}$$

$$= \mathbb{E}_x\mathrm{Var}_\omega\left[\frac{L(x,\omega)}{\Phi(x)q(\omega)}\right] = \mathbb{E}_x\mathrm{Var}_\omega\left[\frac{p(\omega \mid x)}{q(\omega)}\right] \tag{4.11}$$

where $x \sim \hat{p}_X$ is the sampling distribution of position, which depends on the density of particles traced from the camera. $\hat{p}_X$ can be different from the ground truth marginal distribution $p_X(x) = \int_\Omega p(x,\omega)\mathrm{d}\omega = \int_\Omega \frac{L(x,\omega)}{\int_X \Phi(x)\mathrm{d}x}\mathrm{d}\omega$. In this toy setup, we assume a uniform spatial sampling distribution $\hat{p}_X(x) = \frac{1}{x_{\max} - x_{\min}}$.

Notably, it can be shown that the variance term in eq. (4.11) is the χ^2-divergence between p and q [MMR+19].

4.3.2. Optimal Guiding Distribution

It is hard to evaluate the NEV of an arbitrary guiding distribution (e.g. trained with noisy samples using an iterative EM algorithm), so we search for the lower bound of the error.

To find the optimal guiding PDF that minimizes $\mathrm{NEV}(q)$, we apply calculus of variation, inspired by how Rath et al. [RGH+20] find the best PDF for variance-aware guiding. Detailed derivation can be found in appendix A.1. The optimal PDF is given by

$$q^*(\omega) = \arg\min_q \mathrm{NEV}(q) = \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}/C_1 \tag{4.12}$$

$$C_1 = \int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}\mathrm{d}\omega \tag{4.13}$$

with which the **NEV lower bound** is found

$$\mathrm{NEV}\left(q^*\right) = \min_q \mathrm{NEV}(q) = \left(\int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}\mathrm{d}\omega\right)^2 - 1 \tag{4.14}$$

It is worth noting that the optimal PDF differs from trivially marginalizing the ground truth PDF

[2] This corresponds to the mean relative squared error of rendering the fluence with the guiding distribution.

with $\hat{p}_X$, which gives

$$q_{\text{trivial}}(\omega) = \int_X p(\omega \mid x)\, \hat{p}(x)\mathrm{d}x / C_2 = \mathbb{E}_x[p(\omega \mid x)]/C_2 \tag{4.15}$$

$$C_2 = \int_\Omega \mathbb{E}_x[p(\omega \mid x)]\mathrm{d}\omega \tag{4.16}$$

Fig. 4.3 shows the difference of q^* and q_{trivial} under our setting eq. (4.7).

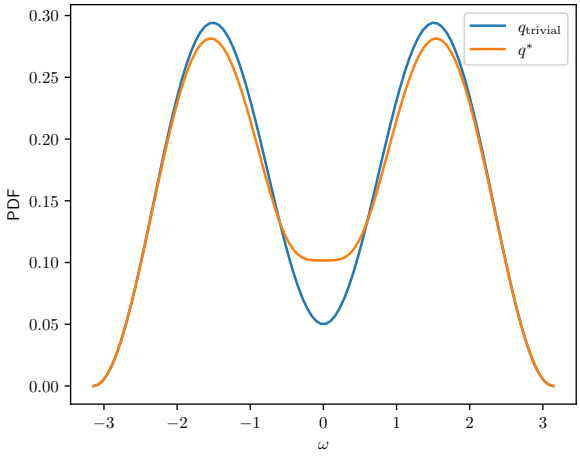


Figure 4.3.: *Marginalized distribution v.s. optimal distribution in the demonstration setup (1)*

In the special case when $p(\omega \mid x)$ does not explicitly depend on x, we have $q^*(\omega) = p(\omega \mid x) = p(\omega)$, and the lower bound error becomes $\mathrm{NEV}(q^*) = \left(\int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega)^2\right]}\mathrm{d}\omega\right)^2 - 1 = \left(\int_\Omega p(\omega)\mathrm{d}\omega\right)^2 - 1 = 0$. This aligns with the zero-variance guiding theory.

4.3.3. Optimal Split Decision

With the assistance of the error lower bound formula eq. (4.14), we can derive the optimal split decision.

Let's denote the parent node distribution by $q_p(\omega \mid x) = q_p(\omega)$ and the child combined model as q_c

$$q_c(\omega \mid x) = \begin{cases} q_l(\omega \mid x) = q_l(\omega) & x < t \\ q_r(\omega \mid x) = q_r(\omega) & x \geqslant t \end{cases} \tag{4.17}$$

where $t \in X$ is the split pivot.

If we assume the guiding distribution is optimally trained for the children, the cost of a split at

t can be defined as NEV lower bound

$$\text{cost}^*(t) \coloneqq \text{NEV}\left(q_c^*\right) = \mathbb{P}(x < t)\ \text{NEV}\left(q_l^*\right) + \mathbb{P}(x \geqslant t)\ \text{NEV}\left(q_r^*\right) \tag{4.18}$$

$$= \mathbb{P}(x < t) \left(\int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]} \mathrm{d}\omega \right)^2 + \mathbb{P}(x \geqslant t) \left(\int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]} \mathrm{d}\omega \right)^2 - 1 \tag{4.19}$$

where $\mathbb{P}(x < t) = \int_{x_{\min}}^{t} \hat{p}(x)\mathrm{d}x,\ \mathbb{P}(x \geqslant t) = \int_{t}^{x_{\max}} \hat{p}(x)\mathrm{d}x$ are the fraction of samples on the left and right.

The best pivot is then the one that results in the lowest cost, or the most reduction in the NEV lower bound. We trigger the split if the lowest child cost is significantly lower than the parent NEV lower bound.

Altogether, the *optimal split criteria* are written as

$$\text{Split if} \quad \text{NEV}\left(q_p^*\right) - \min_{t \in X} \text{cost}^*(t) > C_{\text{threshold}} \tag{4.20}$$

$$\text{at pivot} \quad t^* = \arg\min_{t \in X} \text{cost}^*(t) \tag{4.21}$$

To generalize it to the 3D spatial domain, the cost function is redefined as

$$\text{cost}^*(d, t) \coloneqq \mathbb{P}(\mathbf{x}_d < t)\ \text{NEV}\left(q_l^*\right) + \mathbb{P}(\mathbf{x}_d \geqslant t)\ \text{NEV}\left(q_r^*\right), \quad d \in \{1, 2, 3\} \tag{4.22}$$

The split criteria

$$\text{Split if} \quad \text{NEV}\left(q_p^*\right) - \min_{d,t} \text{cost}^*(d, t) > C_{\text{threshold}} \tag{4.23}$$

$$\text{at dimension and pivot} \quad (d^*, t^*) = \arg\min_{d,t} \text{cost}^*(d, t) \tag{4.24}$$

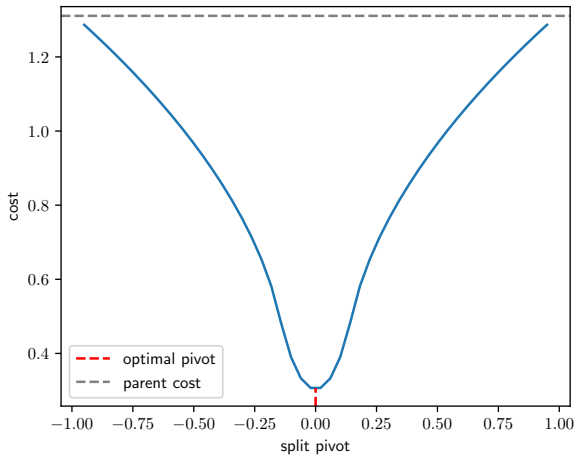


Figure 4.4.: *The cost curve of the demonstration setup (1). The cost function reaches its minimum of 0.307 at $t = 0$. The reduction rate over the parent error is 76.6%*

Unfortunately, eqs. (4.14) and (4.19) involve highly intractable double integrals, which have no analytic forms for most of the cases. Section 4.4 will provide practical solutions to strive for robust subdivisions while circumventing the computation of optimal split costs. For the demonstrative example eq. (4.7), we calculate the error lower bounds and cost function by quadrature, with results shown in fig. 4.4.

The cost curve reaches its minimum at $t = 0$. This coincides with our intuition of splitting at the middle.

4.3.4. Experiments

We experiment with our optimal split strategy eqs. (4.20) and (4.21) in more 2D radiance fields, with results and analysis presented below.

Setup 2

$$L(x, \omega) = 1 + \cos(\omega) \tag{4.25}$$
$$x \in [-1, 1],\ \omega \in [-\pi, \pi]$$

This describes a radiance field that is constant to x. In the angular domain, it is a cut-off cosine-shaped function.

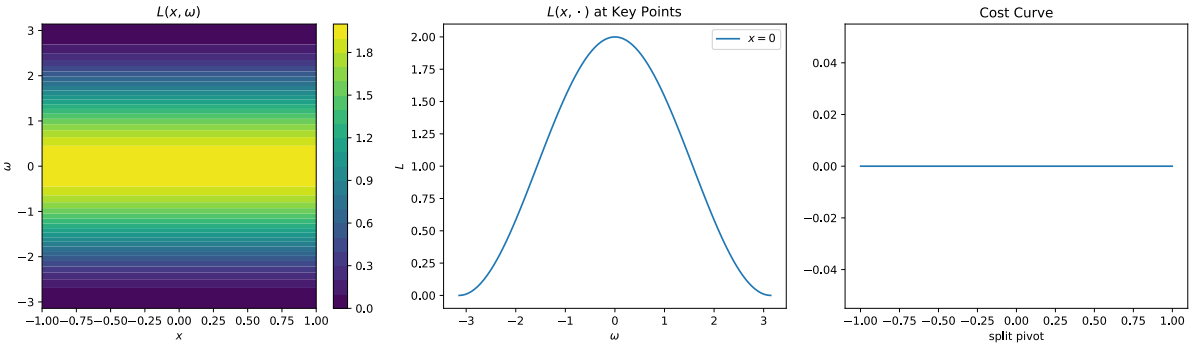


Figure 4.5.: *Optimal subdivision experiment (2).*

Since in this case, the lower bound MRSE $L(q^*) = 0$ within any spatial range (see section 4.3.2), the cost curve remains zero. There is no need for subdivision.

Setup 3.1

$$L(x, \omega) = 1 + \cos(\varphi(2\omega + 2x)) \tag{4.26}$$
$$\varphi \sim (4.8)$$
$$x \in [-1, 1],\ \omega \in [-\pi, \pi]$$

This is a simplified case of eq. (4.7), where the illumination changes uniformly with respect to x in the whole domain. The cost is minimized at $t = 0$.

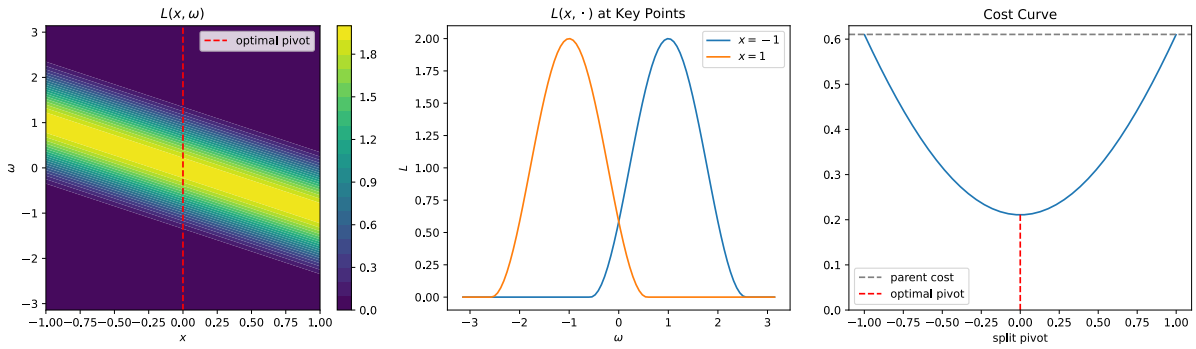


Figure 4.6.: *Optimal subdivision experiment (3.1).*

Setup 3.2

$$L(x, \omega) = 1 + \cos(\varphi(2\omega + x^2 + 2x - 1)) \tag{4.27}$$
$$\varphi \sim (4.8)$$
$$x \in [-1, 1],\ \omega \in [-\pi, \pi]$$

This is a variant of eq. (4.26) whose change to x is nonlinear. The optimal split leans slightly towards the right at 0.23 to balance the "average" signal change on the left and right.

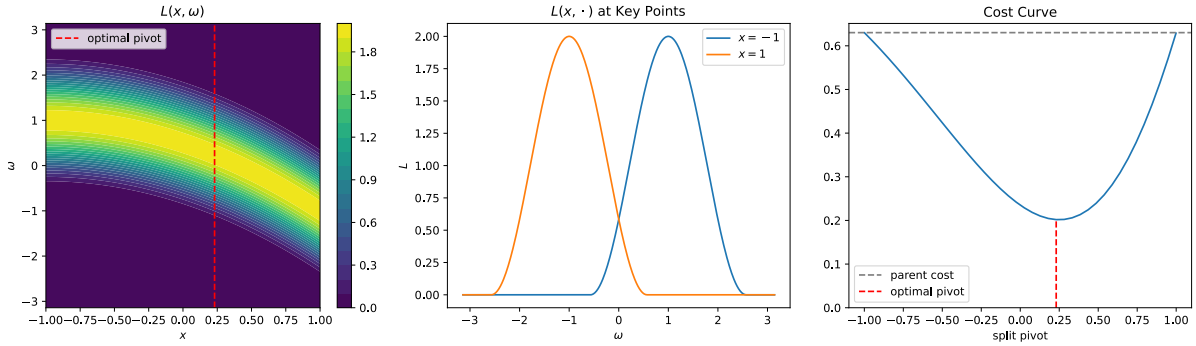


Figure 4.7.: *Optimal subdivision experiment (3.2).*

Setup 3.3

$$L(x, \omega) = 1 + \cos(\varphi(2\omega + 2x^2 - 1)) \tag{4.28}$$
$$\varphi \sim (4.8)$$
$$x \in [-1, 1],\ \omega \in [-\pi, \pi]$$

This is a more elusive example. One may argue that the optimal pivot happens at $t = 0$ because the signal is symmetric about the center. However, upon closer inspection, splitting at the middle will leave us the left and right fields with the exact same configuration as the full field, just half as small. The fact $\text{cost}(0) =$ parent cost proves this.

In fact, two equally optimal pivots are found, symmetrically located about the center.

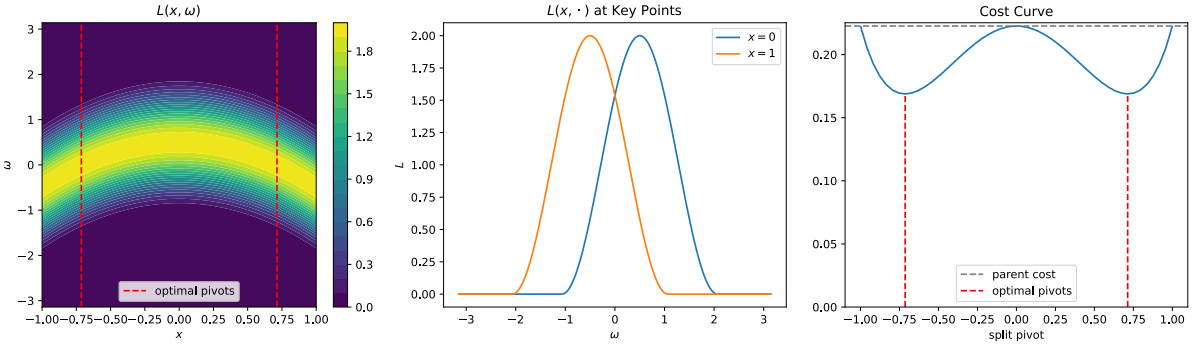


Figure 4.8.: *Optimal subdivision experiment (3.3).*

Setup 4

$$L(x, \omega) = \underbrace{10\eta\left(2x\right)\mathcal{N}(\omega \mid 0, 0.1^2)}_{\text{sunlight}} + \underbrace{0.1}_{\text{ambient light}} \tag{4.29}$$

$$\eta(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 1 \\ t & \text{otherwise} \end{cases} \tag{4.30}$$

$$x \in [-1, 1],\ \omega \in [-5, 5]$$

This mimics a narrow-band sunlight superposed on an ambient light. $\eta\left(2x\right)$ attenuates the sunlight from completely shadowed to completely bright, where the transition is like a "penumbra".

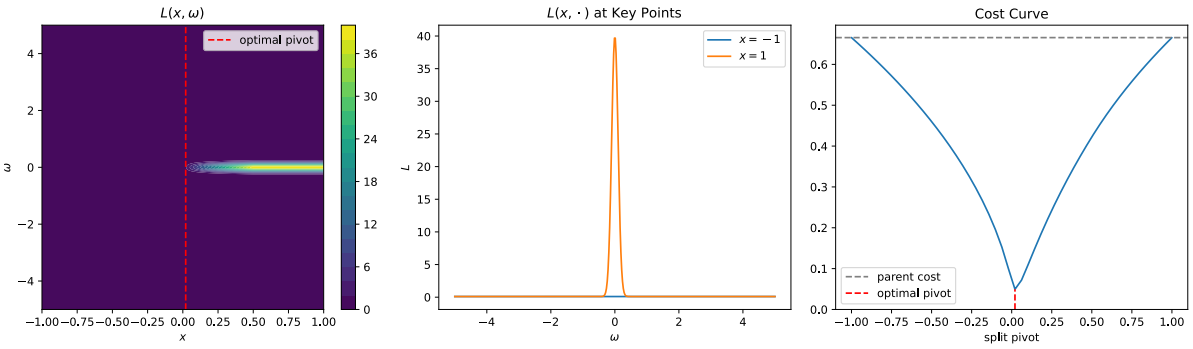


Figure 4.9.: *Optimal subdivision experiment (4).*

The optimal split is found near the left edge of the penumbra.

Setup 5

$$L(x,\omega) = 10(\text{lerp}(\text{lerp}(\underbrace{\mathcal{N}(\omega \mid 0, 0.1^2)}_{\text{left sunlight}}, \underbrace{\mathcal{N}(\omega \mid 1, 0.1^2)}_{\text{middle sunlight}}, g(x)), \underbrace{\mathcal{N}(\omega \mid -1, 0.1^2)}_{\text{right sunlight}}, h(x)) + \underbrace{0.1}_{\text{ambient}} \tag{4.31}$$

$$\text{lerp}(a, b, t) = (1-t)a + tb$$
$$g(x) = \eta(5x + 0.5)$$
$$h(x) = \eta(5x - 2)$$
$$\eta \sim (4.30)$$
$$x \in [-1, 1],\ \omega \in [-5, 5]$$

In this setting, sunlights in three different directions are linearly transitioned via $g(x)$ and $h(x)$. The cost curve gives two local optima (the red and the yellow dashed lines). They both occur near the transition region.

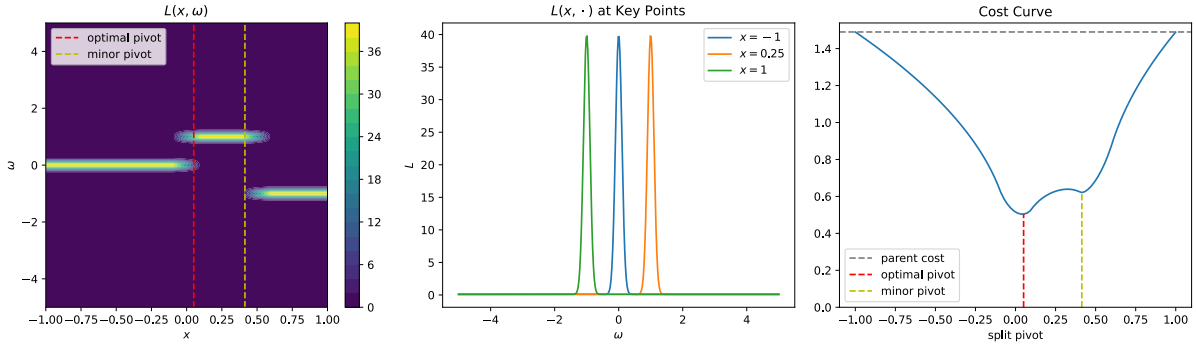


Figure 4.10.: *Optimal subdivision experiment (5).*

4.3.5. Conclusion

From the above experiments, we have the following findings:

1. The cost is usually minimized around the turning points of radiance, i.e. $\{x \mid \frac{\partial L(x,\omega)}{\partial x} \neq 0\}$. Multiple local minima can occur if the turning points consist of multiple intervals.
2. The cost responds rapidly to the pivot near turning points.
3. The more spatially constant the radiance is, the less we gain from subdivision.

4.4. Practical Requirements

Though the optimal cost function eq. (4.19) can tell us accurate information about the spatial domain, it requires solving intractable integrals and the knowledge of the ground truth PDF that we have no access to. Moreover, our ideal assumptions in section 4.3 are not met in reality:

1. There is nested variance in the incident radiance estimate.
2. The guiding distribution can be inter-correlated with the spatial location due to the parallax compensation.
3. The subdivision is a recursive routine, which means the suboptimality in earlier splits can be resolved with recursive splits in the grandchildren nodes.
4. The guiding distribution is trained with finite noisy MC samples, which results in fluctuation in the fitting, leading to higher NEV than our lower bound in an unpredictable manner.

These concerns drive us to develop more practical subdivision strategies. We outline several requirements to achieve robust subdivision based on observations in remaining problems (section 3.5) and the findings in 2D experiments (section 4.3.5), if not optimal.

1. Minimum samples per split. The split decisions (when and where) depend on the samples arriving at a node. For instance, Open PGL's strategy computes the statistics of sample positions, which are noisy estimates. Imposing a lower limit of accumulated valid samples before a split can reduce the noise in the split decision.

2. Minimum samples per node. To ensure stability and low variance in the directional fitting, a minimum number of valid samples (e.g. 1k) is required for each of the child nodes after the split. This requirement leads us to be somewhat defensive about refinements.

3. Adaptive to light change. As shown in figs. 1.1 and 3.7, a low-resolution grid around the regions with rapid light change can incur more noise in the rendering, while fig. 3.9 says a high-resolution grid for uniformly-lit regions wastes memory and lookup time. Insights from ideal setups (section 4.3.5) also tell us the benefit of splitting in response to light change. Thus, an adaptive solution is necessary.

4. Geometry-aware. As observed from the blue arrows in figs. 1.1 and 3.7 to 3.9, incident radiance at opposing geometry surfaces tend to be drastically different. A geometry-aware subdivision should try to **separate different geometries into distinct regions**. We will discuss this in more detail in section 5.2.1. Notably, this requirement can overlap with requirement 3.

5. Explorative in the beginning. As shown in fig. 3.10, having a defensive split strategy does not always do us good. When a node contains a large region with intricate lighting, it is challenging to see the benefit of a split immediately, and the algorithm can get stuck in *local optima*. This problem usually arises in the first few splits. To avoid it, we need to encourage the guider to subdivide more aggressively in early iterations.

6. Isotropic. K-d trees offer fast construction and query only when it is close to balanced ones. We find in practice if an algorithm tends to split at the very ends of a region (called *thin*

splits), though still meeting the minimum samples per node constraint, it is prone to construct ill-formed tall trees with a lot of consecutive thin splits (see the upper left of fig. 4.11), which may compromise the efficiency. Therefore, we impose some penalties on thin splits in the algorithm design. In other words, we often prefer **middle splits** and **cubic** (isotropic) regions to elongated rectangular cuboids. Nevertheless, this requirement can contradict requirement 3 in some cases. When the lighting only varies along a specific axis (e.g. the middle column of figs. 6.43 and 6.44), anisotropic splits can be more effective and should be tolerated.

7. Robust to degeneration. In special cases where samples lie on a plane or even a line parallel to one of the coordinate planes, the algorithm **shouldn't select the degenerate dimensions**, otherwise it can ambiguate the region boundaries due to the rounding in the floating point representation. For instance, if the samples are parallel to the xz-plane (see the lower right of fig. 4.11), splitting at Y is dangerous because it is very likely that $|y_{\min} - y_{\max}| < \epsilon_{\text{machine}}$, causing the split pivot t to be rounded away.

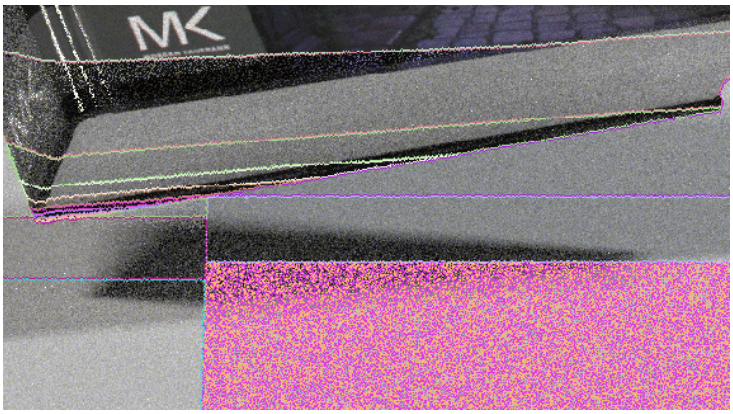


Figure 4.11.: *Problems of not correctly handling thin splits and degenerate samples (rendering overlaid with region boundaries in the PBRT Book scene). The upper left book part is split by consecutive thin y-cuts. The lower right plane is filled with flickering region boundaries because of an unexpected y-cut within the samples parallel to the xz-plane.*

It is worth mentioning that some requirements may overlap or conflict with each other, so it is crucial to find a balance.

Open PGL's solution meets requirements 1 and 2 by only triggering the split at a maximum sample count threshold (e.g. 32k) and placing the pivot at the mean, which is close to the median, ensuring that the child nodes receive around half of the threshold (e.g. 16k). Also, thanks to the mean split, isotropy (requirement 6) is maintained. Selecting the axis with the highest variance will ensure the robustness to degeneration. Nevertheless, other requirements are not satisfied.

In chapter 5, we will present several proposed methods that address one or multiple of these requirements individually. Finally, we will introduce our full models that integrate these components, with an aim of achieving all of the goals.

5

Methodologies

We answer the sub-decisions of a split (section 4.2) from two aspects. Section 5.1 responds to the first sub-decision, whereas section 5.2 addresses the second and third sub-decisions. Our complete spatial subdivision models are introduced in section 5.3.

5.1. When to Split

In this section, we mainly focus on improving k-d trees' adaptivity to light change by controlling the timing to trigger a split. Specifically, we answer the first sub-decision:

> Should this node be further split to refine the partition?

To simplify the discussion and to modularize our findings, we leverage Open PGL's split pivot and dimension selection (mean and longest axis of valid samples):

$$\text{Split at dimension} \quad d = \arg\max_d V(S_{\text{valid}})_d \tag{5.1}$$

$$\text{and pivot} \quad t = (\bar{\mathbf{x}}_{\text{valid}})_d \tag{5.2}$$

where

$$S_{\text{valid}} \coloneqq \{s_i \in S : \langle \Phi(\mathbf{x}_i) \rangle > 0\} \tag{5.3}$$

$$N_{\text{valid}} \coloneqq |S_{\text{valid}}| \tag{5.4}$$

$$\bar{\mathbf{x}}_{\text{valid}} = \frac{1}{|S_{\text{valid}}|} \sum_{s_i \in S_{\text{valid}}} \mathbf{x}_i \tag{5.5}$$

and the 3D sample variance

$$V(S)_d \coloneqq \frac{1}{N} \sum_{i=1}^{N} ((\mathbf{x}_i)_d - \bar{\mathbf{x}}_d)^2, \quad d \in 1, 2, 3 \tag{5.6}$$

Starting from the root node, how do we know the right time to split?

First of all, practical requirement 2 says we cannot have too few samples during a split. We find 1k samples a suitable safeguard for the child nodes' fitting. So the parent node should have at least 2k valid samples before a split. We also find the mean and variance estimates of 2k sample positions quite stable, so the minimum samples per split constraint is automatically met.

Therefore, we do not proceed with a split if there are fewer than two thousand samples. Beyond this constraint, what additional considerations must we address?

5.1.1. Lookahead Guiding Regions

If we always split with more than 2k samples, we might create over-refinements that violate practical requirement 3. Moreover, we have to bear in mind that the child regions have access to fewer samples. Although it sounds reasonable to split at the places with light changes, the increased noise of fitting in a smaller region can make the child distributions worse than the parent's for guiding.

A safe choice is to create a "hypothetical" but not real split, resulting in two *candidate* child regions. The candidate regions are independently trained with samples arriving in themselves. We do not realize the split until the candidate regions' guiding caches are actually better than the parent one.

Interchangeably, we refer to hypothetical splits as *candidate splits*. We term candidate regions as *lookahead guiding regions* because they grant our k-d tree the ability to foresee the horizon by one level down.

The lookahead guiding regions are demonstrated in fig. 5.1. Note that they are only created when the leaf node has accumulated more than 2k samples.

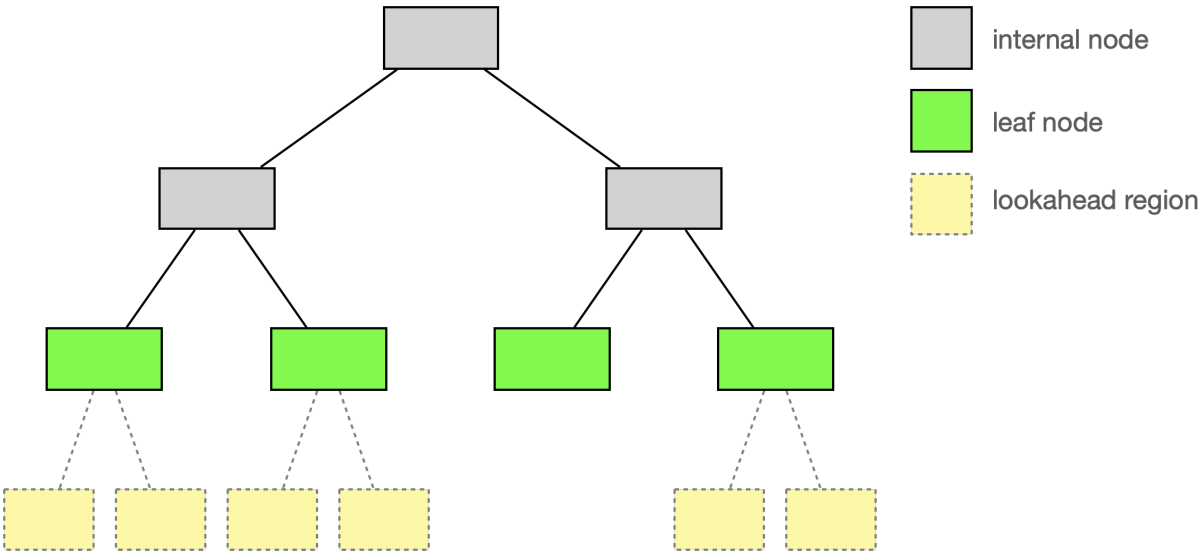


Figure 5.1.: *Illustration of lookahead guiding regions. Note that some leaf nodes might not have lookahead regions because they haven't accumulated enough samples for a candidate split. The lookahead regions are not attached to the k-d tree yet, so they are counted as leaves.*

Now, we are left with one last question: how to compare the guiding cache of the parent (a leaf node) with the candidate children's?

For clarity and consistency, let's first recap a few terminologies and notations in table 5.1 that will be used throughout the section.

Notation	Explanation
X_p	Region of the parent (a leaf node).
X_l, X_r	Region of the left or right candidate child. $X_l \cup X_r = X_p \wedge X_l \cap X_r = \emptyset$.
$L(\mathbf{x}, \omega)$	Incident radiance at the location $\mathbf{x}$ towards direction ω.
$\Phi(\mathbf{x}) = \int L(\mathbf{x}, \omega) \mathrm{d}\omega$	Fluence at $\mathbf{x}$.
$p(\omega \mid \mathbf{x}) = \frac{L(\mathbf{x}, \omega)}{\Phi(\mathbf{x})}$	Ground truth lighting distribution at $\mathbf{x}$. It is the zero-variance PDF for the MC integration of fluence.
$\hat{p}_X(\mathbf{x})$	Distribution of sample position, or the density of particles traced from the camera, conditioned on region X.
$q(\omega \mid \mathbf{x})$	A guiding distribution to check.
$q_p(\omega \mid \mathbf{x})$	Guiding distribution of the parent. Without loss of generalization, we include the variable $\mathbf{x}$ to indicate the possible use of parallax compensation.
$q_c(\omega \mid \mathbf{x}) \sim$ (5.7)	Guiding distribution of the candidate children. Note that the term "child model" refers to the combined distribution that shares the same spatial domain X_p as the parent.
$q_s(\omega \mid \mathbf{x})$	Actual sampling distribution to generate training samples eq. (4.2). It can be the mixture distribution between the guiding distribution and the BSDF sampling distribution eq. (2.16).
$\langle L(\mathbf{x}, \omega) \rangle$	Noisy observation of incident radiance.
$\langle \Phi(\mathbf{x}) \rangle = \frac{\langle L(\mathbf{x}, \omega) \rangle}{q_s(\omega \mid \mathbf{x})}$	the MC estimator of fluence, or the weight of training samples eq. (4.3).

Table 5.1.: *Notations for adaptive splitting approaches*

The child model q_c is defined as

$$q_c(\omega \mid \mathbf{x}) \coloneqq \begin{cases} q_l(\omega \mid \mathbf{x}) & \mathbf{x} \in X_l \\ q_r(\omega \mid \mathbf{x}) & \mathbf{x} \in X_r \end{cases} \tag{5.7}$$

In the following, we will introduce two solutions (sections 5.1.2 and 5.1.3) that achieve the comparison between the guiding cache of the leaf regions and that of the lookahead regions.

5.1.2. Adaptive Splitting Based on Divergence Estimates

Inspired by Ruppert et al.'s [RHL20] variance-aware split and merge of the VMM, we attempt to evaluate the cache using the divergence between a guiding distribution q of interest and the ground truth distribution p.

Marginalized KL Divergence

We start with Kullback-Leibler divergence, a widely used metric in machine learning. It measures the dissimilarity between two PDFs. The KL divergence between any distribution $p(\omega)$ and $q(\omega)$ is given by

$$D_{\mathrm{KL}}(p \parallel q) \coloneqq \int_{\Omega} p(\omega) \log \frac{p(\omega)}{q(\omega)} \mathrm{d}\omega \tag{5.8}$$

where the support of q should cover that of p: $\forall \omega,\ p(\omega) > 0 \rightarrow q(\omega) > 0$

According to Jensen's inequality, the KL divergence is always non-negative. It equals zero if and only if $p(\omega) = q(\omega)$ almost everywhere.

The KL divergence can boil down to the *cross-entropy* between p, q minus the *entropy* of p.

$$D_{\mathrm{KL}}(p \parallel q) = H(p, q) - H(p) \tag{5.9}$$

$$H(p, q) \coloneqq -\int_{\Omega} p(\omega) \log q(\omega) \mathrm{d}\omega \tag{5.10}$$

$$H(p) \coloneqq -\int_{\Omega} p(\omega) \log p(\omega) \mathrm{d}\omega \tag{5.11}$$

We now define the *marginalized KL divergence* between a guiding distribution to be checked (defined on a region X) and the ground truth distribution as the spatial expectation of the directional-domain KL divergence:

$$\tilde{D}_{\mathrm{KL}}(q) \coloneqq \mathbb{E}_{\mathbf{x} \sim \hat{p}_X}[D_{\mathrm{KL}}(p_{\omega|\mathbf{x}} \parallel q_{\omega|\mathbf{x}})] = \mathbb{E}_{\mathbf{x} \sim \hat{p}_X}\left[\int p(\omega \mid \mathbf{x}) \log \frac{p(\omega \mid \mathbf{x})}{q(\omega \mid \mathbf{x})} \mathrm{d}\omega\right] \tag{5.12}$$

Expanding it using eq. (5.9), we rewrite it into

$$\tilde{D}_{\mathrm{KL}}(q) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_X}[H(p_{\omega|\mathbf{x}}, q_{\omega|\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \hat{p}_X}[H(p_{\omega|\mathbf{x}})] \tag{5.13}$$

The second term does not depend on q, so we are interested in the first *marginalized cross-entropy* term

$$\tilde{H}(q) \coloneqq \mathbb{E}_{\mathbf{x} \sim \hat{p}_X}[H(p_{\omega|\mathbf{x}}, q_{\omega|\mathbf{x}})] \tag{5.14}$$

We developed an unbiased eq. (A.19) and a biased MC estimator eq. (A.33) for it. The derivations can be found in appendix A.2.1. Since the unbiased estimator involves the ground truth

fluence, which we do not have access to, we adopt the biased one to evaluate the guiding cache of the parent and child.

To simplify, let's introduce the following *sufficient statistics* that rely on sample data eq. (4.2) at the child half-spaces X_l, X_r and guiding PDFs:

$$a_{p,l} = \sum_{i:\ \mathbf{x}_i \in X_l} -\langle \Phi(\mathbf{x}_i) \rangle \log q_p(\omega_i \mid \mathbf{x}_i),\ a_{p,r} = \sum_{i:\ \mathbf{x}_i \in X_r} -\langle \Phi(\mathbf{x}_i) \rangle \log q_p(\omega_i \mid \mathbf{x}_i) \tag{5.15}$$

$$a_{c,l} = \sum_{i:\ \mathbf{x}_i \in X_l} -\langle \Phi(\mathbf{x}_i) \rangle \log q_l(\omega_i \mid \mathbf{x}_i),\ a_{c,r} = \sum_{i:\ \mathbf{x}_i \in X_r} -\langle \Phi(\mathbf{x}_i) \rangle \log q_r(\omega_i \mid \mathbf{x}_i) \tag{5.16}$$

$$f_l = \sum_{i:\ \mathbf{x}_i \in X_l} \langle \Phi(\mathbf{x}_i) \rangle,\ f_r = \sum_{i:\ \mathbf{x}_i \in X_r} \langle \Phi(\mathbf{x}_i) \rangle \tag{5.17}$$

$$N_l = \mathrm{freq}(\mathbf{x}_i \in X_l),\ N_r = \mathrm{freq}(\mathbf{x}_i \in X_r),\ N = N_l + N_r \tag{5.18}$$

where freq counts the frequency of samples in that region.

Substituting the terms in eqs. (A.34) and (A.35), we arrive at the final estimators

$$\langle \tilde{H}(q_p) \rangle = \frac{N_l}{N} \frac{a_{p,l}}{f_l} + \frac{N_r}{N} \frac{a_{p,r}}{f_r} \tag{5.19}$$

$$\langle \tilde{H}(q_c) \rangle = \frac{N_l}{N} \frac{a_{c,l}}{f_l} + \frac{N_r}{N} \frac{a_{c,r}}{f_r} \tag{5.20}$$

Since the sufficient statistics depend on the split dimension and position, they are collected **per child region** instead of the parent one as samples come in, as is illustrated in fig. 5.2.

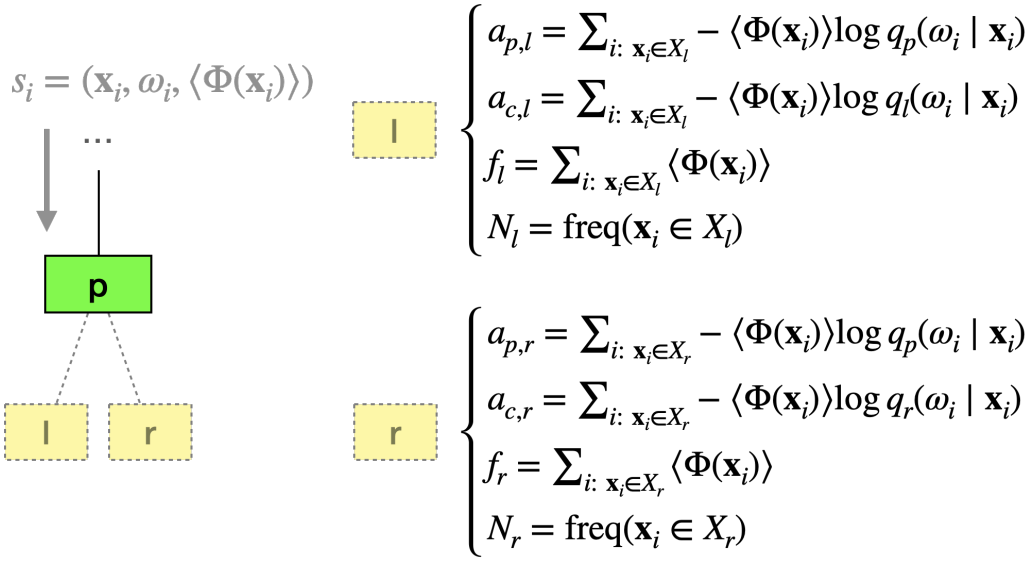


Figure 5.2.: *The storing of sufficient statistics for marginalized cross-entropy*

Importantly, to obtain unbiased estimates for fluence, sufficient statistics need to take all samples into account, including those with zero weights.

Marginalized χ^2-Divergence

Ruppert et al.'s approach (section 3.3) of adaptive merge and split is based on the χ^2-divergence eq. (3.3), which corresponds to NEV [MMR+19] and is more closely related to rendering error.

Similarly, we can define the *marginalized χ^2-divergence* between q and p:

$$\tilde{D}_{\chi^2}(q) := \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}[D_{\chi^2}(p_{\omega|\mathbf{x}} \parallel q_{\omega|\mathbf{x}})] = \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}\left[\int \frac{p(\omega \mid \mathbf{x})^2}{q(\omega|\ \mathbf{x})}\mathrm{d}\omega - 1\right] \tag{5.21}$$

Due to the occurrence of the squared ground truth PDF, we failed to find an unbiased estimator for it. The biased MC estimator is developed in appendix A.2.2.

Similarly, we define the following sufficient statistics for the parent and child regions:

$$b_{p,l} = \sum_{i:\ \mathbf{x}_i\in X_l} \langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega_i \mid \mathbf{x}_i)/q_p(\omega_i \mid \mathbf{x}_i),\ b_{p,r} = \sum_{i:\ \mathbf{x}_i\in X_r} \langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega_i \mid \mathbf{x}_i)/q_p(\omega_i \mid \mathbf{x}_i) \tag{5.22}$$

$$b_{c,l} = \sum_{i:\ \mathbf{x}_i\in X_l} \langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega_i \mid \mathbf{x}_i)/q_l(\omega_i \mid \mathbf{x}_i),\ b_{c,r} = \sum_{i:\ \mathbf{x}_i\in X_r} \langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega_i \mid \mathbf{x}_i)/q_r(\omega_i \mid \mathbf{x}_i) \tag{5.23}$$

$$f_l,\ f_r \sim (5.17)$$
$$N_l,\ N_r,\ N \sim (5.18)$$

Plugging them into eqs. (A.40), (A.46) and (A.47) yields the estimators for the parent and child marginalized χ^2-divergence:

$$\langle\tilde{D}_{\chi^2}(q_p)\rangle = \frac{N_l}{N}\cdot\frac{N_l b_{p,l}}{f_l^2} + \frac{N_r}{N}\cdot\frac{N_r b_{p,r}}{f_r^2} - 1 \tag{5.24}$$

$$\langle\tilde{D}_{\chi^2}(q_c)\rangle = \frac{N_l}{N}\cdot\frac{N_l b_{c,l}}{f_l^2} + \frac{N_r}{N}\cdot\frac{N_r b_{c,r}}{f_r^2} - 1 \tag{5.25}$$

Notably, due to more approximations made in the derivation, our χ^2-divergence estimators exhibit more bias than the the cross-entropy ones eqs. (5.19) and (5.20). The presence of the squared terms in the χ^2-divergence estimators also produce more noise in the evaluation, as evidenced by section 6.1.1 and fig. 6.16.

Split Criteria

The divergence estimators allow us to compare the quality of the parent's and candidate children's guiding caches. The lower the divergence is, the better the guiding cache approximates the true signals. Due to the intrinsic noise from MC integration, we trigger the split of a leaf node only when the divergence of its candidate children is **significantly lower** than that of the parent. Formally, the split criteria are written as

$$\begin{aligned} \text{Split if} \quad & N_{\text{valid}} \geqslant 2N_{\min} \wedge \langle\tilde{D}(q_p)\rangle - \langle\tilde{D}(q_c)\rangle > D_{\text{ths}} \\ \text{at dimension} \quad & \sim (5.1) \\ \text{and pivot} \quad & \sim (5.2) \end{aligned} \tag{5.26}$$

where $\tilde{D}$ can be the marginalized cross-entropy or χ^2-divergence, D_{ths} is a fixed constant threshold. $N_{\min}$ is the minimum number of valid samples per child node after the split, hence the factor 2.

Implementation Details

When to evaluate? In the original pipeline of Open PGL's training phase, the k-d tree is incrementally subdivided, and then guiding caches at the leaf nodes are fit or updated with the incoming samples. Our proposed divergence metrics require evaluating the caches with the samples. This is only possible when the guiding PDF is available. If we naively put the evaluation after the fitting step, the divergence tends to be **underestimated** because we are training and testing with the same dataset[1].

The smaller the dataset, the more likely the underestimation will be. Hence, even if the parent and child regions share the same lighting configuration, the child's divergence is still more likely to be lower than the parent's. We find in practice, this can indeed lead to over-refinements in a super simple uniformly-lit scene (see fig. 6.3).

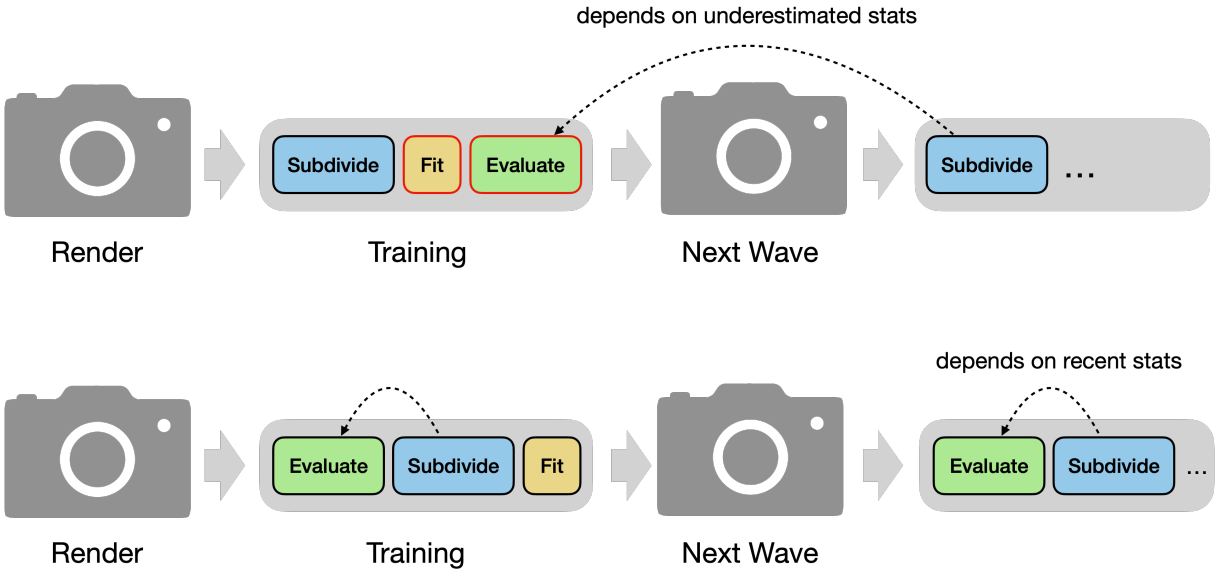


Figure 5.3.: *Adaptive splitting pipeline. Upper: naive order. Split decisions depend on statistics from the previous iteration. Lower: improved order reduces this issue.*

To remedy this, we adjust the order of subdivide-fit-evaluate steps to evaluate-subdivide-fit, as shown in fig. 5.3. Since the fit or updated cache is now evaluated with samples from the next rendering wave, which is much less correlated with the guiding distribution, the underestimation is less severe.

Guiding distribution for queries. During the rendering phase, the renderer queries the guiding PDF at a given scene point $\mathbf{x}$ and samples from it. Since lookahead regions are probably

[1]This is known as "overfitting" in machine learning

premature, we exclude them from the k-d tree retrieval. i.e. the retrieval should return a (non-lookahead) leaf node.

Candidate split handling. When a leaf node first exceeds the minimum sample per split threshold, a candidate split should be created. The candidate split dimension and pivot are determined by eqs. (5.1) and (5.2). The sufficient statistics stored at the children are initialized to the **child one** stored at the parent [2]. Similar to how Open PGL inherits and decays the VMM statistics when a child region is created, we also decay our sufficient statistics. Specifically

$$\begin{aligned} a_{p,l} = a_{c,l} = a_{p,r} = a_{c,r} &\leftarrow \lambda a'_c \\ b_{p,l} = b_{c,l} = b_{p,r} = b_{c,r} &\leftarrow \lambda b'_c \\ f_l = f_r &\leftarrow \lambda f' \\ N_l = N_r &\leftarrow \lambda N' \end{aligned} \tag{5.27}$$

where λ is the decay ratio (we use 0.25 in our experiments). The meaning of a, b, f, N is given in eqs. (5.15) to (5.18), (5.22) and (5.23). a'_c, b'_c, f', N' denote the **child** statistics stored at the parent node. This handling implies we start from a "tie"-position between the parent and candidate child cache.

Real split handling. When the split condition eq. (5.26) is met, the candidate split dimension and pivot are adopted. The parent leaf node becomes an internal node, while two new leaf nodes are created, reusing the cache of the former candidate regions.

Fail-to-split handling. Training samples from the early rendering waves are more noisy because of the lack of a guiding cache. This could result in more fluctuation in the divergence estimate. If some unlucky candidate region gets unexpectedly high divergence estimates during early evaluations, there are chances when it will never be adopted in the future, even if its guiding distribution surpasses the parent's over time.

To avoid the curse of "bad history" and to encourage exploration, we propose the following handling when a split test eq. (5.26) fails:

1. We decay the sufficient statistics by λ.
2. The mean and variance statistics, as well as the VMM statistics, are also decayed by λ.
3. The candidate split dimension and pivot are re-proposed with the new mean and variance statistics in the hope for a more balanced valid sample distribution.

5.1.3. Adaptive Splitting Based on Multiple Importance Sampling

In this section, we will present an alternative adaptive subdivision approach based on multiple importance sampling (MIS).

[2] which once was a candidate child to the grandparent

MIS is used for combining multiple sampling strategies when we don't which one is the most suitable. For instance, the actual sampling distribution used by Ruppert et al. [RHL20] and our renderer is the mixture of the guiding distribution and the BSDF sampling distribution eq. (2.16). Vorba et al. [VHH+19] present an online approach to adjust this weight such that the mixture PDF resembles the ground truth PDF. If we look from another view, the optimized weight tells us how much BSDF is preferred to the guiding distribution.

The Mixture Distribution

Inspired by this, we can also build a mixture model with the leaf region cache q_p and the lookahead region cache q_c, when we are unsure which one is better.

$$q_\alpha(\omega \mid \mathbf{x}) := (1 - \alpha(\mathbf{x}))\, q_p(\omega \mid \mathbf{x}) + \alpha(\mathbf{x})\, q_c(\omega \mid \mathbf{x}) \tag{5.28}$$

The MIS weight $\alpha(\mathbf{x}) \in [0, 1]$ is set to a constant value (which will be discussed shortly) in each lookahead region (as shown in fig. 5.4), whose meaning is the preference of the child cache to the parent cache. Assuming we can find the best value of α, it can offer us an alternative way to compare the parent and child distributions.

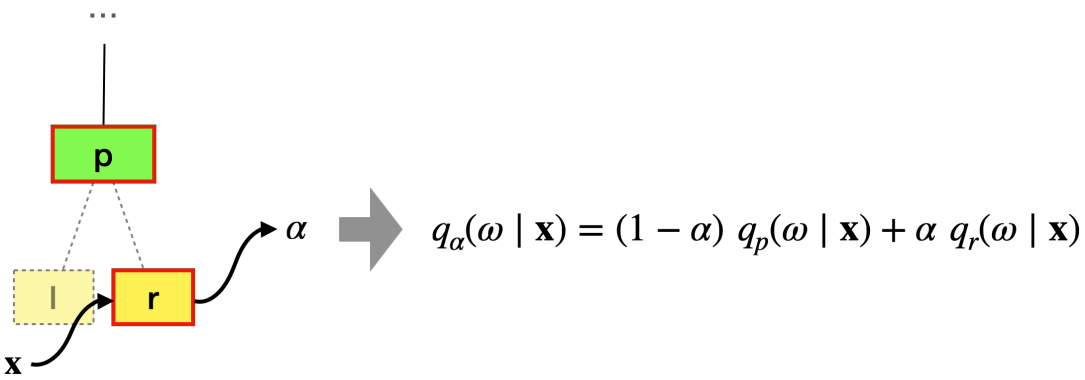


Figure 5.4.: *Mixture between a leaf region's and a lookahead region's cache at* $\mathbf{x}$*. The MIS weight* α *is constant within the hit lookahead region.*

MIS Weight Optimization

Motivated by the previous works [VHH+19, MMR+19], we can optimize the MIS weight per lookahead region, by minimizing the divergence between the mixture distribution q_α and the ground truth distribution p. We choose the marginalized KL divergence (eq. (5.12)), for its simplicity and numeric stability.

$$\alpha^* = \arg\min_{\alpha \in [0,1]} \tilde{D}_{\mathrm{KL}}(q_\alpha) \tag{5.29}$$

Sbert et al. [SSK22] show that eq. (5.29) is a convex optimization problem. They apply the second-order Newton-Raphson method to solve for the optimal α^*.

Vorba et al. [VHH+19] leverage an *adaptive moment estimation* (Adam) optimizer [KB14] to solve for eq. (5.29). To further constraint the weight between 0 and 1, they introduce a hidden variable and a sigmoid activation

$$\alpha(\theta) = \frac{1}{1+\exp(-\theta)}, \quad \theta \in \mathbb{R} \tag{5.30}$$

and optimize θ instead of α.

In our experiments, we find out Vorba et al.'s method more stable than that of Sbert et al. thanks to Adam's adjustment to the learning rate based on momentum estimates of gradients. In theory, the second-order method converges faster than the first-order Adam optimizer with few gradient samples. Still, in our cases, each region contains at least thousands of samples, making the stability issue more pronounced than the convergence speed. Therefore, we adopt the Adam optimization approach.

Moreover, we abandon the use of sigmoid activation eq. (5.30) and directly optimize α to avoid the vanishing gradient problem of the sigmoid function. Mathematically, applying the sigmoid activation also violates the convexity of the optimization problem eq. (5.29).

Now, let's introduce our optimization approach.

The stochastic gradient for α is derived as

$$\begin{aligned}
\nabla_\alpha \tilde{D}_{\mathrm{KL}}(q_\alpha) &= \nabla_\alpha \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}[D_{\mathrm{KL}}(p_{\omega|\mathbf{x}} \parallel q_{\alpha,\omega|\mathbf{x}})] && (5.31)\\
&= \nabla_\alpha \mathbb{E}_{\mathbf{x},\omega\sim q_s(\cdot|\mathbf{x})}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\log q_\alpha(\omega|\,\mathbf{x})\right] \qquad \text{similar to (A.18)} && (5.32)\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\nabla_\alpha \log q_\alpha(\omega|\,\mathbf{x})\right] && (5.33)\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\frac{\nabla_\alpha q_\alpha(\omega|\,\mathbf{x})}{q_\alpha(\omega|\,\mathbf{x})}\right] \qquad \text{chain rule} && (5.34)
\end{aligned}$$

Using the identity $\nabla_\alpha q_\alpha(\omega|\,\mathbf{x}) = q_c(\omega \mid \mathbf{x}) - \ q_p(\omega \mid \mathbf{x})$, we have

$$\nabla_\alpha \tilde{D}_{\mathrm{KL}}(q_\alpha) = \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\frac{q_c(\omega \mid \mathbf{x}) - \ q_p(\omega \mid \mathbf{x})}{q_\alpha(\omega|\,\mathbf{x})}\right] \tag{5.35}$$

Vorba et al. eliminate the normalizer $\Phi(\mathbf{x})$ by assuming it is a constant scaling factor, which can be handled by the Adam optimizer. However, we point out that due to the spatial marginalization $\mathbb{E}_{\mathbf{x}\sim\hat{p}_X}$, the normalizer serves as a location-dependent weighting factor, which should not be ignored. To circumvent this issue, we redefine our optimization target as

$$\begin{aligned}
J(\alpha) &\coloneqq \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}[\Phi(\mathbf{x})D_{\mathrm{KL}}(p_{\omega|\mathbf{x}} \parallel q_{\alpha,\omega|\mathbf{x}})] + \frac{1}{2}\eta(\alpha-\alpha_0)^2 && (5.36)\\
\alpha^* &= \arg\min_{\alpha\in[0,1]} J(\alpha) && (5.37)
\end{aligned}$$

which multiplies the ground truth fluence to the KL divergence before spatial marginalization. Similar to Vorba et al., we add a regularization term to pull the MIS weight towards a constant α_0, which is set to 0.5 to indicate a "tie" between the parent and child cache.

The gradient of the new objective is

$$\nabla_\alpha J(\alpha) = \mathbb{E}_{\mathbf{x},\omega}\left[-\langle\Phi(\mathbf{x})\rangle \frac{q_c(\omega \mid \mathbf{x}) - q_p(\omega \mid \mathbf{x})}{q_\alpha(\omega \mid \mathbf{x})}\right] + \eta(\alpha - \alpha_0) \tag{5.38}$$

We can build an MC estimator (called the *stochastic gradient*) for eq. (5.38), using incoming sample data

$$\langle\nabla_\alpha J(\alpha)\rangle = \frac{1}{M}\sum_{i=1}^{M}\left[-\langle\Phi(\mathbf{x}_i)\rangle \frac{q_c(\omega_i \mid \mathbf{x}_i) - q_p(\omega_i \mid \mathbf{x}_i)}{q_\alpha(\omega_i \mid \mathbf{x}_i)}\right] + \eta(\alpha - \alpha_0) \tag{5.39}$$

where the batch size M is independent of the sample count N.

The incoming samples are divided into batches of size m. Batch gradients eq. (5.39) are iteratively fed to an Adam optimizer to update the MIS weight.

$$\text{state} \leftarrow \text{AdamStep}(\text{state}, \langle\nabla_\alpha J(\alpha)\rangle) \tag{5.40}$$

The state variables of our Adam optimizer include the iteration count i, the first and second momentum estimate m, v, and the parameter α.

$$\text{state} \coloneqq (i, m, v, \alpha) \tag{5.41}$$

Lastly, we clamp the updated α to be in the range of $[0.05, 0.95]$ to reserve minimum probabilities of sampling each of the components.

In theory, the parent and child PDFs are evolving over iterations, which could lead to a shift in the optimal MIS weight. Fortunately, we find in practice, the MIS weight usually converges to a certain value in a stochastic manner.

Split Criteria

If we assume that q_p and q_c try to learn the same signals, then $\alpha < 0.5$ indicates the parent PDF is better, while $\alpha > 0.5$ indicates the preference of the child PDF. But similar to section 5.1.2, noise in the fitting and the optimization leads to oscillation of the MIS weight. To be resistant to this noise and to control the "aggressiveness" of subdivision, we trigger the real split only when α is above some threshold α_{ths} (e.g. 0.6, 0.7, etc.).

Yet, there are two MIS weights at a leaf region-the left and right child's. How do we make a split decision with two numbers? To maintain the balance of the tree and to encourage exploration, we trigger the split if either the left or right weight exceeds the threshold:

$$\begin{aligned} \text{Split if} \quad & N_{\text{valid}} \geqslant 2N_{\min} \wedge (\alpha_l > \alpha_{\text{ths}} \vee \alpha_r > \alpha_{\text{ths}}) \\ \text{at dimension} \quad & \sim (5.1) \\ \text{and pivot} \quad & \sim (5.2) \end{aligned} \tag{5.42}$$

Theoretically, an intrinsic problem with this approach arises from the ambiguity of multi-solutions. When q_p and q_c are equal, any α between 0 and 1 can serve as the solution for eq. (5.29). The optimization process in this case is unstable, triggering the split criteria unpredictably.

Implementation Details

Guiding distribution for queries. Unlike divergence-based solutions, the mixture guiding distribution eq. (5.28) naturally makes a valid PDF that can be sampled with one-sample MIS. It has some slight advantages over the PDF of the leaf region q_p in that, it opts for a better approximation of p with by trading off q_p and the lookahead region PDF q_c. To exploit this fact, during a query, we retrieve both the leaf region PDF and the lookahead region PDF (if it exists), and return the mixture PDF using the optimized MIS weight stored at that lookahead region, as shown in fig. 5.4.

Efficiency-wise, we provide an alternative query handling by only returning the leaf region PDF.

Initialization. The MIS weight of a newly created candidate region is initialized to 0.5, the prior weight implied by the regularization term.

Candidate split handling. Different from section 5.1.2, the lookahead regions do not inherit the Adam state variables eq. (5.41) from the parent region, but instead initialize them to zero (α to 0.5).

Hyperparameters. The hyperparameters of the Adam optimizer along with M and η are set as the following, according to our experiment results

- Learning rate $lr = 10^{-3}$.
- Exponential decay rate for the first moment $\beta_1 = 0.9$.
- Exponential decay rate for the second moment $\beta_2 = 0.999$.
- Epsilon $\epsilon = 10^{-8}$.
- Batch size $M = 16$.
- Regularization strength $\eta = 10^{-3}$.

5.1.4. Defensive Fallback To Escape Local Optima

Our adaptive solutions have an intrinsic limitation-they look ahead by only one level down, assuming the benefit of refinements can be directly seen by the algorithm. In reality, this is not always the case. This greedy approach can get stuck when the cache does not improve until two or more recursive splits.

In a case such as fig. 6.18, the adaptive approach gets stuck at the first split. Ideally, it should have subdivided like Open PGL's approach fig. 6.20. The candidate split is placed where the resulting marginalized ground truth PDFs of the lookahead regions do not change much from the parent one. Consequently, the cross-entropy estimates of the child are not significantly lower than the parent's, misleading the algorithm to stop subdivision at a shallow tree depth.

Completely solving this problem is not trivial. It might require the training and evaluation of lookahead guiding caches at multiple levels, which is implementation- and performance-wise intractable.

We propose a simple fix by adding a maximum sample count threshold to the split criteria eq. (5.26), similar to Open PGL. Specifically, we enforce a real split when the leaf node has accumulated more valid samples than a large value (e.g. 512k).

The final split criterion for the divergence-based adaptive splitting is

$$\text{Split if} \quad \left(N_{\text{valid}} \geqslant 2N_{\min} \wedge \langle \tilde{D}(q_p) \rangle - \langle \tilde{D}(q_c) \rangle > D_{\text{ths}}\right) \vee N_{\text{valid}} \geqslant N_{\max} \tag{5.43}$$

For the MIS-based adaptive splitting

$$\text{Split if} \quad (N_{\text{valid}} \geqslant 2N_{\min} \wedge (\alpha_l > \alpha_{\text{ths}} \vee \alpha_r > \alpha_{\text{ths}})) \vee N_{\text{valid}} \geqslant N_{\max} \tag{5.44}$$

In response to requirement 5 in section 4.4, we set the maximum sample count threshold to a smaller value (e.g. 32k) when the node's depth is no greater than 4.

The experiment results for this section and their analysis can be found in sections 6.1.1 and 6.2.1.

5.2. Where to Split

In this section, we pay attention to the orthogonal problem of choosing the split dimension and pivot. i.e. the second and the third sub-decisions:

> If yes, which dimension should be chosen?
> Where should the splitting occur?

Again, to simplify and modularize the discussion, we adopt Open PGL's strategy of "when to split":

$$\text{Split if} \quad N_{\text{valid}} \geqslant N_{\max} \tag{5.45}$$

Learned from the optimal guiding takeaways section 4.3.5, the best split should happen at the turning points of lighting[3]. Nevertheless, the challenge in solving for the theoretically optimal split position eq. (4.21) and the noise in the observation do not allow us to directly infer the best split. Instead, we strive for heuristics-based solutions that only depend on sample data $S = \{s_i\}_{i=1}^{N}$, with an aim of efficiency and robustness.

5.2.1. Geometry-Aware Splitting

The Proximity Assumption

Open PGL splits at the mean and longest axis of valid samples. As a result, it produces close-to-cubic regions where the average distance from samples to the center of mass is somehow

[3]To recap, turning points are defined as $\{\mathbf{x} \mid \partial L/\partial x \neq 0 \vee \partial L/\partial y \neq 0 \vee \partial L/\partial z \neq 0\}$

minimized. The underlying heuristic of this approach is that "points that are close to each other tend to have similar lighting distribution", which we refer to as the *proximity assumption*.

The proximity assumption holds at uniformly-lit regions with flat geometries such as the diffuse surfaces in figs. 3.7 and 3.8. However, it breaks in either a non-uniform lighting scenario (such as fig. 1.1) or regions with complex geometries.

For instance, in the collage fig. 5.5, opposing close-by surfaces exhibit distinct incident radiance (with a different hemisphere or visibility to light sources). However, the proximity assumption misleads Open PGL's strategy to group these surfaces into one region, causing suboptimal guiding.

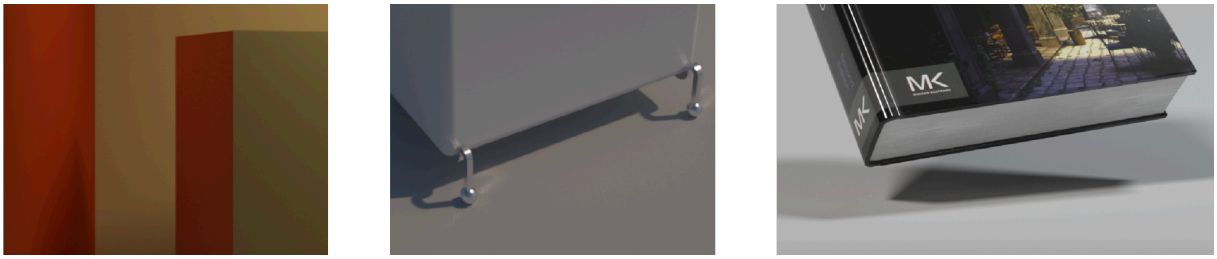


Figure 5.5.: *Failure cases of the proximity assumption (Patches from Cornell Box, Cube, and PBRT-Book).*

The Geometry Assumption

We refine the proximity assumption by proposing the *geometry assumption*, which states that "proximate points that lie on the same geometry tend to have similar lighting distribution", corresponding to requirement 4.

This heuristic fixes the failure cases in fig. 5.5. According to the geometry assumption, the red wall and the cube in fig. 5.5 (left), the cube and ground (middle), as well as the book and the ground (right) should belong to distinct regions because they are from different geometry objects.

Importantly, this is still a very crude heuristic because there is no guarantee that the incident radiance depends on which geometry it hits. Close points on two geometries could have similar lighting distributions, while points on one geometry could have distinct lighting profiles (e.g. a large geometry with self-shadows).

We have developed a family of algorithms driven by the above assumptions that only take sample positions $\{\mathbf{x}_i\}_{i=1}^{N}$ into account.

Variance-Scanning

As a first attempt, we come up with an algorithm that finds the best split by minimizing a cost function. The cost function quantifies the proximity of the points in the left and right half-spaces

if splitting at dimension d and pivot t. It is defined as the sum of within-group variances.

$$\text{cost}_0(d,t) \coloneqq TV(S_l(d,t)) + TV(S_r(d,t)) \tag{5.46}$$

$$(d^*,t^*) = \arg\min_{d,t} \text{cost}_0(d,t),\ \text{s.t.}\ \min(N_l,N_r) \geqslant N_{\min} \tag{5.47}$$

The meaning of each term is given by

$$\text{Total variance} \quad TV(S) \coloneqq \sum_{k=1}^{3} V(S)_k \tag{5.48}$$

$$\text{3D sample variance} \quad V \sim (5.6) \tag{5.49}$$

$$\text{Samples on the left} \quad S_l(d,t) = \{s_i \in S : (\mathbf{x}_i)_d < t\} \tag{5.50}$$

$$\text{Samples on the right} \quad S_r(d,t) = \{s_i \in S : (\mathbf{x}_i)_d \geqslant t\} \tag{5.51}$$

Since the total variance corresponds to the average squared distance from samples to the center of mass, the cost measures the dispersion of samples in the child regions in an average sense.

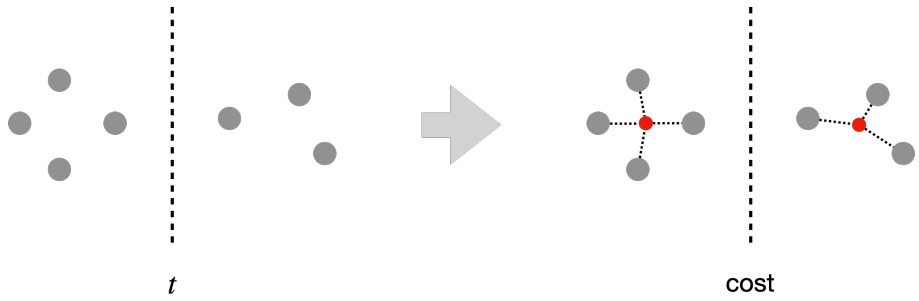


Figure 5.6.: *Variance-scanning with cost eq.* (5.46)*. Left: input samples. Right: the cost is computed as the sum of squared distances to the center of mass.*

The constraint $\min(N_l,N_r) \geqslant N_{\min}$ ensures the minimum number of samples is assigned to each of the children, for a safe variance estimate and directional fitting.

Finding the minimum. To find the minimum eq. (5.47), we can exhaust all possible dimensions $d \in \{1,2,3\}$ and key points $t_i \in \{(\mathbf{x}_i)_d\}$ subject to the constraint, compute the variances and costs at each point, and then take their minimum. This naive algorithm has a runtime of $O(3 \times N \times N) = O\left(N^2\right)$ as each variance-pair takes $O(N)$ to compute. Compared to Open PGL that runs in $O(N)$, the $O\left(N^2\right)$ overhead makes it unattractive.

To improve the performance, we can first sort the samples according to their d-dimension coordinates, and then enumerate t from $(\mathbf{x}_{N_{\min}+1})_d$ to $(\mathbf{x}_{N-N_{\min}+1})_d$. The advantage of sorting is that it allows us to compute the variances $S_l(d,t)$ and $S_r(d,t)$ on the fly as we sweep t from left to right[4]. The improved algorithm runs in $O(3 \times (N \log N + N)) = O(N \log N)$.

Since this algorithm involves a left-to-right scan, we call it *variance-scanning*. Fig. 5.6 illustrates the workflow and intuition behind the cost metric.

[4]Welford algorithm for online variance update [Wel62, Knu97]

Unfortunately, the formulation eq. (5.46) is extremely unstable. A 2D experiment fig. 5.7 (upper) visualizes the cost function of 200 points uniformly sampled from $[0, 1]^2$. Ideally, if the samples are perfectly random, using the variance formula for uniform distribution

$$\mathrm{Var}_{X\sim\mathrm{Unif}[0,l]}[X] = \frac{1}{12}l^2 \tag{5.52}$$

The cost function then forms a parabola

$$\mathrm{cost}_{\mathrm{theory}}(t) = \frac{1}{12}\left(t^2 + 1^2\right) + \frac{1}{12}\left((1-t)^2 + 1\right) \tag{5.53}$$

which is minimized at the middle point $t = 0.5$. However, due to the discrepancy in the random sequence, the actual cost curve is contaminated by noise, which gives a suboptimal split pivot at 0.9. Furthermore, we notice that the cost function in this example is higher than the total variance of the parent $TV(S)$.

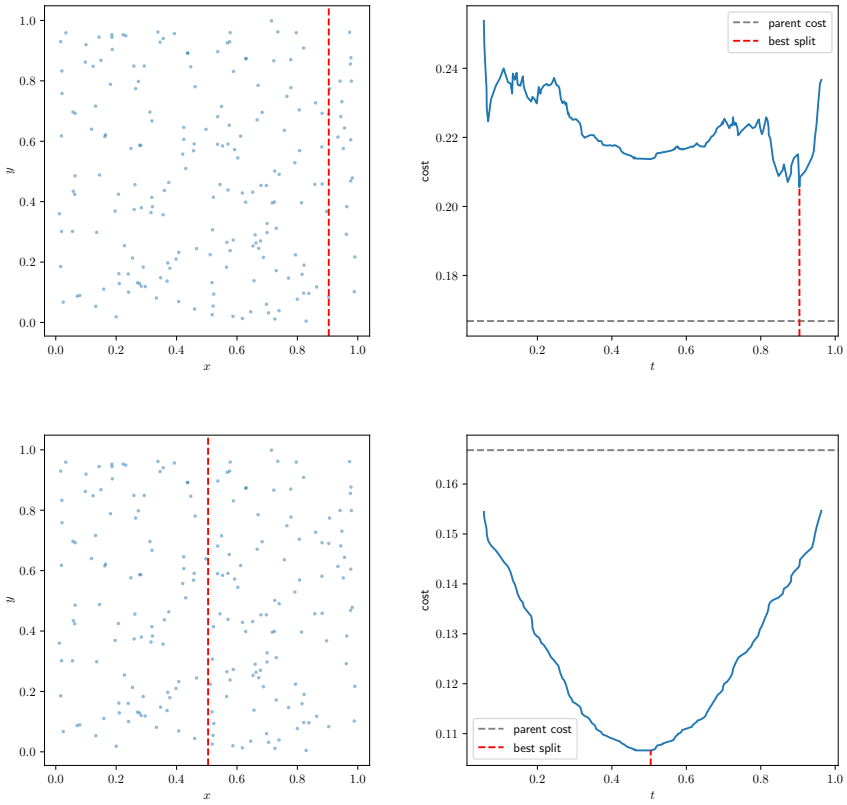


Figure 5.7.: *A 2D experiment of variance-scanning without (upper) v.s. with (lower) weighting terms.*

Weighting terms. To fix this problematic formulation, we change the cost definition to be the **weighted** sum of within-group variances

$$\mathrm{cost}_1(d, t) := \frac{N_l}{N}TV(S_l(d, t)) + \frac{N_r}{N}TV(S_r(d, t)) \tag{5.54}$$

where the weighting terms $\frac{N_l}{N}, \frac{N_r}{N}$ are the fractions of samples on the left or right. They serve as a prior to pull the optimal t^* towards the median of the samples, necessary for the algorithm's robustness.

This weighted version of variance-scanning eq. (5.54) produces the lower plots in fig. 5.7, whose cost curve is much smoother and has a minimum point at around 0.5. Besides, the cost curve is strictly below the parent cost, defined as the total variance of the samples in the parent space $TV(S)$.

In experiments (section 6.2.2), we find the variance-scanning with cost eq. (5.54) behaves very similarly to Open PGL, which tends to split at the samples' median, not really addressing the geometry assumption.

Analysis. To understand this problem, let's zoom in to the middle patch in fig. 5.5 and examine the cost curve of the samples there.

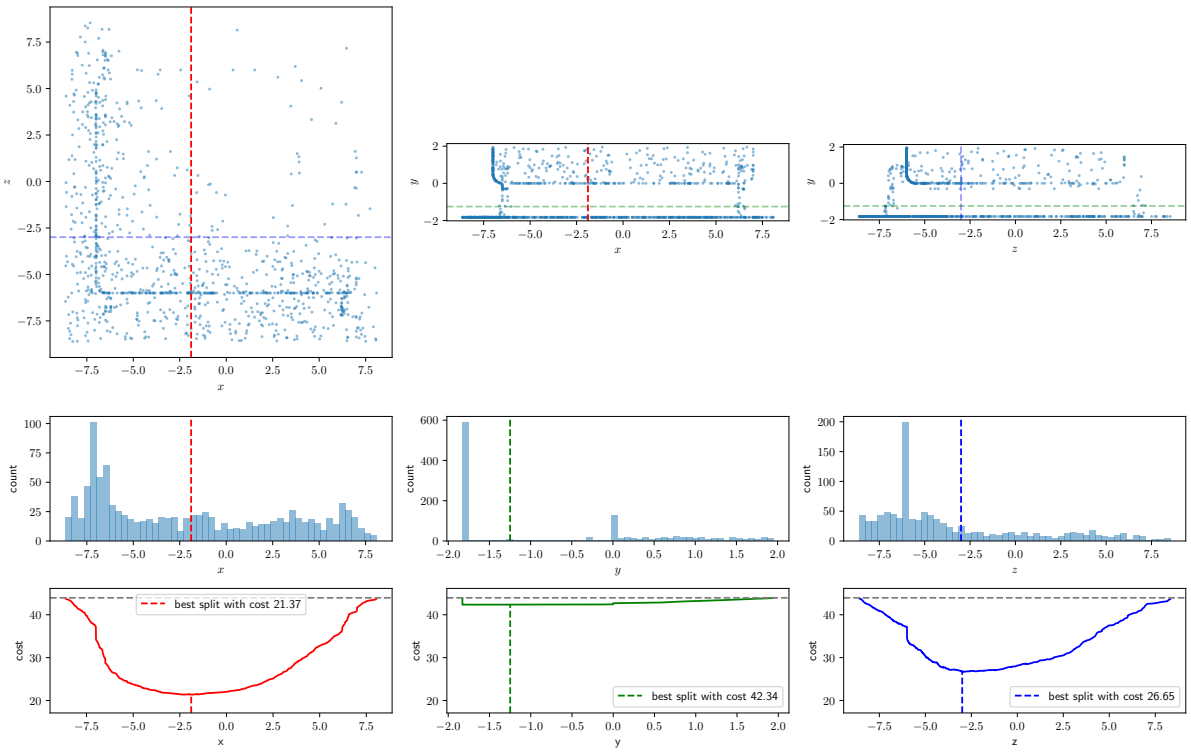


Figure 5.8.: *Variance-scanning analysis (Cube). The first row shows the top, front, and left views of samples on a region containing the cropped cube and the ground. The second row shows the histograms in each dimension. The third row visualizes the cost function eq.* (5.54)*. The dashed gray horizontal line shows the parent cost.*

In the plots fig. 5.8, x-cut (red dashed line) is selected because it achieves the lowest cost. However, one would argue the y-cut (green dashed line) is better as it separates the surface of the upper cube and the ground apart, while the x- and z-cuts only care about the proximity but not the geometry.

By inspecting the cost curves, we immediately know that the selection of the x-cut is driven by a high variance reduction along the x-dimension (same for z). Conversely, the variance curve barely changes in the y-dimension. A closer look at the sample views shows that the dispersion of the samples is mainly due to their spread in the x- and z-dimensions (with a length of around 15 units), while the y-axis only extends 4 units. Our variance-scanning considers the total variance, which is the sum of the x, y, z components eq. (5.48). The larger the extent of the axis, the larger its variance contribution. This explains why our variance-scanning favors choosing the longest axis.

Covariance-Scanning

Inspired by fig. 5.8, to make the y-cut stand out, we can change the cost metric into the **product** of the x, y, z variances so that the variance reduction in each dimension becomes equally important.

$$\text{cost}'_1(d,t) := \frac{N_l}{N} PV(S_l(d,t)) + \frac{N_r}{N} PV(S_r(d,t)) \tag{5.55}$$

where

$$\text{Product variance} \quad PV(S) := \prod_{k=1}^{3} V(S)_k \tag{5.56}$$

Using this metric, the result with the same example is drastically improved. Shown in fig. 5.9, the cost reduces most significantly in the y-dimension, yielding a correct horizontal split.

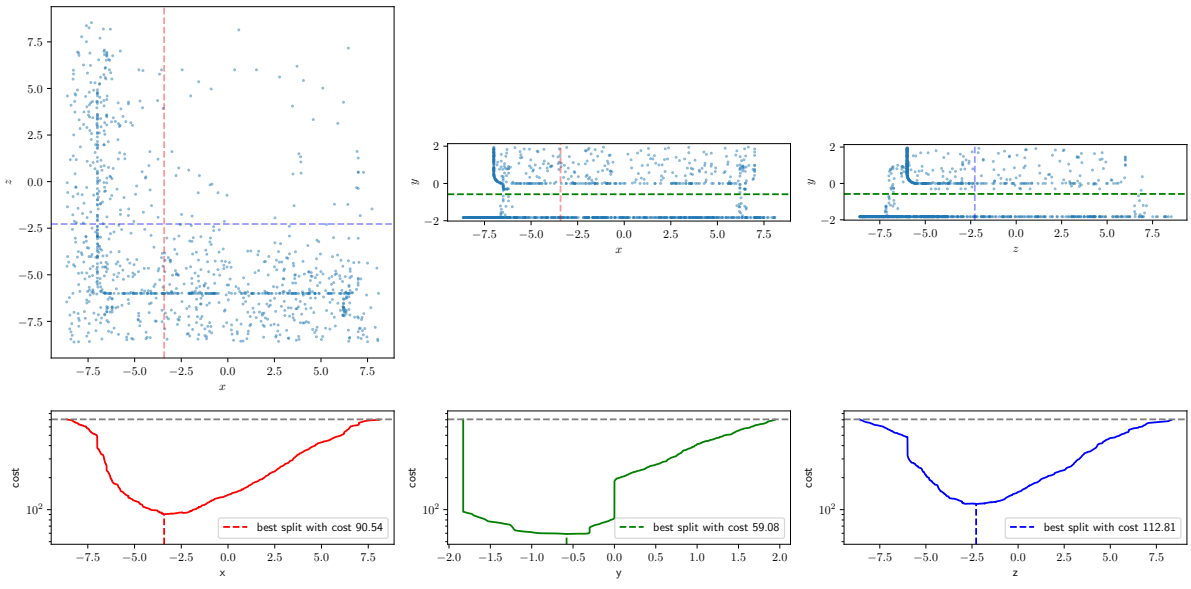


Figure 5.9.: *Improved variance-scanning result with the product metric eq.* (5.55).

This metric has been proven to function as expected in our experiments (sections 6.1.2 and 6.2.2). However, is there any deeper significance to the product variance?

Let's recall a 3D Gaussian distribution with mean μ and covariance Σ

$$\mathcal{N}(\mathbf{x} \mid \mu, \Sigma) \coloneqq \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right) \tag{5.57}$$

The sublevel sets of the distribution are given by ellipsoids centered at μ

$$L_c^-(\mu, \Sigma) = \left\{\mathbf{x} \in \mathbb{R}^3 : (\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu) \leqslant c\right\} \tag{5.58}$$

whose volumes are proportional to the square root of the determinant of Σ

$$\mathrm{Vol}\left(L_c^-(\mu, \Sigma)\right) = \frac{4}{3}\pi\sqrt{\det(c\Sigma)} \propto \sqrt{\det \Sigma} \tag{5.59}$$

In the special case where the covariance matrix is diagonal $\Sigma = \mathrm{diag}\left(\sigma_1^2, \sigma_2^2, \sigma_3^2\right)$, i.e. the sublevel ellipsoids are axis-aligned, the volume boils down to the product of the standard deviations in each dimension

$$\mathrm{Vol}\left(L_c^-(\mu, \Sigma)\right) \propto \sqrt{\sigma_1^2\sigma_2^2\sigma_3^2} = \sigma_1\sigma_2\sigma_3 \tag{5.60}$$

Back to our topic. The product variance metric eq. (5.55) we used is the squared volume of an **axis-aligned** Gaussian fit to the samples.

$$PV(S) = \prod_{k=1}^{3} V(S)_k = \prod_{k=1}^{3} \sigma(S)_k^2 \propto \mathrm{Vol}\left(L_c^-(\mu, \mathrm{diag}(V(S)))\right)^2 \tag{5.61}$$

Therefore, our cost function eq. (5.55) roughly measures the reduction of the squared volume of the samples in the half-spaces. We find in our 2D experiments section 6.1.2. If the region is dominated by multiple geometries, a split that segments geometries apart usually results in the most volume reduction. In other words, this new metric is geometry-aware.

Motivated by eqs. (5.57) to (5.59), a more generalized and robust volume measurement should include the case of **skewed** Gaussians instead of axis-aligned ones, which requires the estimate of the covariance matrix Σ instead of the variances. Moreover, to reveal the reduction of volume instead of squared ones, we should take the square root of $\det \Sigma$. These observations give us the following new cost function

$$\mathrm{cost}_2(d, t) \coloneqq \frac{N_l}{N}\mathrm{Vol}(S_l(d, t)) + \frac{N_r}{N}\mathrm{Vol}(S_r(d, t)) \tag{5.62}$$

where

$$\text{Sample volume estimate} \quad \mathrm{Vol}(S) \coloneqq \sqrt{\det(\mathrm{Cov}(S) + \epsilon\mathrm{I})} \tag{5.63}$$

$$\text{Sample covariance matrix} \quad \mathrm{Cov}(S) \coloneqq \begin{pmatrix} V(S)_1 & \mathrm{Cov}(S_1, S_2) & \mathrm{Cov}(S_1, S_3) \\ \mathrm{Cov}(S_1, S_2) & V(S)_2 & \mathrm{Cov}(S_2, S_3) \\ \mathrm{Cov}(S_1, S_3) & \mathrm{Cov}(S_2, S_3) & V(S)_3 \end{pmatrix} \tag{5.64}$$

$$\text{Component covariance} \quad \mathrm{Cov}(S_a, S_b) = \frac{1}{N}\sum_{i=1}^{N}((\mathbf{x}_i)_a - \overline{\mathbf{x}}_a)((\mathbf{x}_i)_b - \overline{\mathbf{x}}_b) \tag{5.65}$$

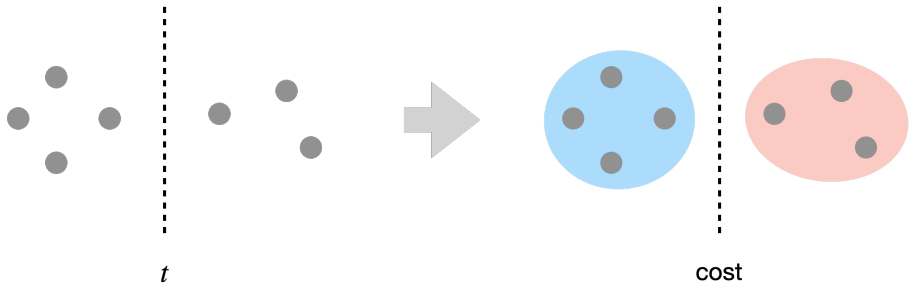


Figure 5.10.: *Covariance-scanning with cost eq. (5.62). Left: input samples. Right: the cost is computed as the volume of a (skewed) Gaussian blob fit to the samples.*

A small $\epsilon = 10^{-6}$ is added to the diagonals of the covariance matrix before taking the determinant eq. (5.63) to avoid degenerate cases. Without this handling, the cost function will always be zero if samples in the whole region are coplanar or colinear.

The advantage of the covariance-based metric eq. (5.62) over the variance-based one eq. (5.55) is demonstrated in an example in fig. 5.11. The best split pivot should occur at the joint where two line segments meet. The variance-based metric (red) fails because it fits an axis-aligned Gaussian, which overestimates the volume of the skewed line on the left, yielding high costs. The covariance-based metric (green) successfully detects the joint location because splitting there will collapse the volumes on both sides to almost zero.

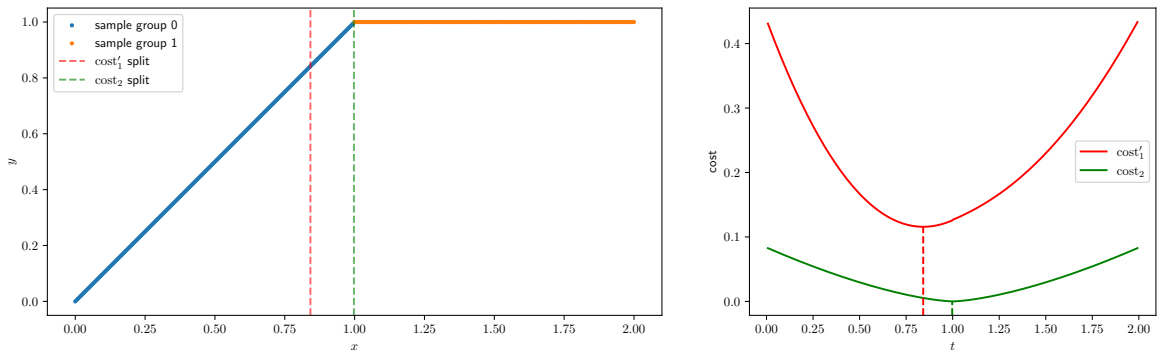


Figure 5.11.: *Variance-scanning v.s. covariance-scanning on a 2D experiment. Left: samples are uniformly placed on two line segments contacting at* $(1, 1)$*. Right: the cost curves of eqs. (5.55) and (5.62).*

The downside is that we need to collect the covariance estimates of samples eq. (5.65), on top of the variances. In total, our scanning algorithm must update six variables online: three variances and three covariances. The new algorithm is thus called *covariance-scanning*, as illustrated by fig. 5.10.

Information-Gain-Scanning

One might still argue that the cost function eq. (5.62) is ill-formed as it is purely an outcome of geometric intuitions. There is no strict proof that a split proposed by eqs. (5.47) and (5.62)

is optimal for guiding. Besides, we did not show the necessity of using volume instead of the squared volume, or the validity of adding two volumes together instead of multiplying them.

On the other hand, we notice three fundamental aspects that make covariance-scanning effective: the weighting terms, the determinant operation, and the covariance matrix.

What other property of the Gaussian distribution involves the determinant of the covariance matrix? Entropy.

$$H(\mathcal{N}(\mu, \Sigma)) = \frac{1}{2} \log \det(2\pi e \Sigma) = \frac{1}{2} \log \det \Sigma + \text{const.} \tag{5.66}$$

Do other algorithms employ weighted entropies? Consider decision trees. Referring back to eq. (2.23), one type of decision tree finds a split by minimizing the mutual information

$$\text{IG}(d, t) = H(S) - \left(\frac{N_l}{N} H(S_l(d, t)) + \frac{N_r}{N} H(S_r(d, t)) \right) \tag{5.67}$$

The original algorithm uses the Shannon entropy of supervised data eq. (2.20). However, in our case, we only have unsupervised position data with a different "entropy" definition eq. (5.66). If we plug that into eq. (5.67), we obtain a novel algorithm

$$\text{IG}(d, t) \coloneqq H(S) - \left(\frac{N_l}{N} H(S_l(d, t)) + \frac{N_r}{N} H(S_r(d, t)) \right) \tag{5.68}$$

$$(d^*, t^*) = \arg\max_{d,t} \text{IG}(d, t), \text{ s.t. } \min(N_l, N_r) \geqslant N_{\min} \tag{5.69}$$

where

$$\text{Sample spatial entropy} \quad H(S) \coloneqq \frac{1}{2} \log \det(\text{Cov}(S) + \epsilon \text{I}) \tag{5.70}$$

Or equivalently, we rewrite it into the following optimization problem

$$\text{cost}_3(d, t) \coloneqq \frac{N_l}{N} H(S_l(d, t)) + \frac{N_r}{N} H(S_r(d, t)) \tag{5.71}$$

$$(d^*, t^*) = \arg\min_{d,t} \text{cost}_3(d, t), \text{ s.t. } \min(N_l, N_r) \geqslant N_{\min} \tag{5.72}$$

Interestingly, eq. (5.70) tells us to take the **log** of the determinant before summing it up into the final cost eq. (5.71). In our experiments sections 6.1.2 and 6.2.2, we will show that the subtle change in the measurement can sometimes improve the subdivision quality compared to the volume-based metric eq. (5.62).

The new gain (cost) function tells us how much information we can gain from making the split, or how much uncertainty the split reduces. Since it is based on concepts from information theory, we name this well-formed new scanning algorithm *information-gain-scanning*, our final geometry-aware splitting solution.

An additional benefit of information-gain-scanning is that it offers a scale-invariant quantity eq. (5.68), whose magnitude is independent of the size of the geometries (see appendix A.3). This property allows us to compare mutual information to some threshold, assisting the split decision in our later discussion.

5.2.2. Lighting-Aware Splitting

So far, we have developed several variance- and covariance-based cost functions that reveal the information of geometries while sweeping left to right. They are parameter-free, simple to implement, and enjoy an efficient run time of $O(N \log N)$.

Motivated by this framework, we aim to expand scanning algorithms to incorporate more information from training samples.

Fluence-Scanning

We are immediately attracted to the sample weights, or the fluence estimates $\langle \Phi(\mathbf{x}_i) \rangle$ that quantify the amount of incoming light at a position.

Suppose we treat the sample position as the input variable and the ground truth fluence as the target variable. In that case, we can directly apply the variance reduction algorithm from regression trees eq. (2.28). This generates the following optimization problem

$$\text{cost}_\Phi(d, t) \coloneqq \frac{N_l}{N} V_\Phi(S_l(d, t)) + \frac{N_r}{N} V_\Phi(S_r(d, t)) \tag{5.73}$$

$$(d^*, t^*) = \arg\min_{d,t} \text{cost}_\Phi(d, t), \text{ s.t. } \min(N_l, N_r) \geqslant N_{\min} \tag{5.74}$$

where

$$\text{Sample fluence variance} \quad V_\Phi(S) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \left(\langle \Phi(\mathbf{x}_i) \rangle - \overline{\langle \Phi(\mathbf{x}) \rangle} \right)^2 \tag{5.75}$$

We can leverage the same sorting-scanning trick to collect the fluence variance on the fly and find the best split in $O(N \log N)$. We refer to such a lighting-aware algorithm as *fluence-scanning*, whose workflow is illustrated in fig. 5.12.

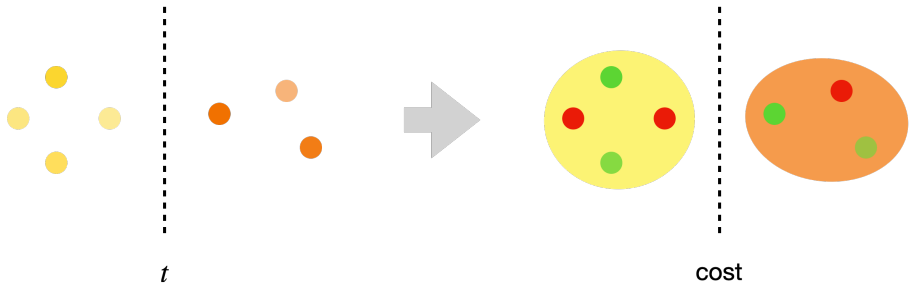


Figure 5.12.: *Fluence-scanning with cost eq.* (5.73)*. Left: input samples annotated with fluence values. Right: the cost is computed as the variance in the fluence estimates. Red and green colors show the fluctuation of fluence estimates.*

Analysis

The cost eq. (5.73) resembles the optimal cost function eq. (4.19) in the weighting terms and error metric.

The optimal cost function entails the marginalized NEV lower bound of the fluence estimator eq. (4.11). The marginalized NEV of a sampling distribution $q_s(\omega \mid \mathbf{x})$ is given by

$$\mathrm{NEV}^*(q_s) = \mathbb{E}_{\mathbf{x}}\mathrm{Var}_{\omega\sim q_s(\cdot|\mathbf{x})}\left[\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\right] = \mathbb{E}_{\mathbf{x}}\mathrm{Var}_{\omega}\left[\frac{L(\mathbf{x},\omega)}{\Phi(\mathbf{x})q_s(\omega \mid \mathbf{x})}\right] \tag{5.76}$$

However, fluence-scanning's error metric eq. (5.75) is actually measuring the *total mean squared error* (MSE) of the fluence estimator

$$\mathbb{E}[V_\Phi(S)] = L_\Phi(q_s) = \mathrm{Var}_{x,\omega\sim q_s(\cdot|\mathbf{x})}[\langle\Phi(\mathbf{x})\rangle] = \mathrm{Var}_{x,\omega}\left[\frac{L(\mathbf{x},\omega)}{q_s(\omega \mid \mathbf{x})}\right] \tag{5.77}$$

The first difference between these two errors is that the extra $\Phi(x)$ term is missing in our eq. (5.77). This may appear as suboptimality in the spatial subdivision regarding relative rendering errors.

Secondly, applying the law of total variance to eq. (5.77), we have

$$\begin{aligned} L_\Phi(q_s) &= \mathrm{Var}_{x,\omega}[\langle\Phi(\mathbf{x})\rangle] && (5.78)\\ &= \mathbb{E}_{\mathbf{x}}\mathrm{Var}_{\omega}[\langle\Phi(\mathbf{x})\rangle] + \mathrm{Var}_{\mathbf{x}}\mathbb{E}_{\omega}[\langle\Phi(\mathbf{x})\rangle] && (5.79)\\ &= \underbrace{\mathbb{E}_{\mathbf{x}}\mathrm{Var}_{\omega}[\langle\Phi(\mathbf{x})\rangle]}_{\text{marginalized MSE}} + \underbrace{\mathrm{Var}_{\mathbf{x}}[\Phi(\mathbf{x})]}_{\text{spatial variance}} && (5.80) \end{aligned}$$

where the first term is the *marginalized MSE* of the fluence estimator. The second term measures the spatial variance of the fluence function. The first term depends on the sampling PDF q_s, usually the guiding or the defensive sampling PDF learned in the previous training iteration, which can introduce inter-correlation into the split decision.

Unfortunately, if we try to remove the marginalized MSE term from the fluence-scanning algorithm, we are left with the intrinsic challenge of estimating the ground truth fluence. Even if we manage to measure only the spatial variance, problems still arise because the fluence function does not tell us enough information about incoming radiance. A failure case is our initial toy setup fig. 4.2 where the incident radiance changes across the space while maintaining a constant fluence $\Phi(\mathbf{x}) = \pi$. A cost function based on the spatial variance of $\Phi(\mathbf{x})$ will be constant with respect to t.

Bridging the gap between the theoretically optimal split cost and fluence-scanning, as well as finding a sensible way to combine the marginalized MSE and spatial variance of fluence will be challenging future avenues.

5.2.3. Degeneration Handling

The ϵ perturbation added to eqs. (5.63) and (5.70) ensures the numeric stability when points are coplanar or colinear. However, if the degeneration happens at one of the coordinate axes, i.e. the position variance along that axis is almost zero, the scanning algorithms will still execute along these axes. This can, in turn, lead to selecting the degenerating dimensions to split, causing pathological and ambiguous subdivisions as illustrated earlier in fig. 4.11.

To avoid this, before scanning along axis d, we check if its variance eq. (5.6) is too small compared to the longest-axis variance. If so, skip this dimension for scanning. Formally

$$\text{Skip dimension } d \text{ if } \quad V(S)_d < k_{\text{skip}} \max_i V(S)_i \tag{5.81}$$

The skip ratio k_{skip} is set to 10^{-4} in our implementation.

5.2.4. Incremental Update

The incremental update means the subdivision algorithm can memorize some information about samples arriving at a k-d tree node if a split decision is negative in the current iteration, and use this information to make better proposals of split pivot and dimension in future iterations.

Open PGL's strategy achieves this goal by storing the mean and variance statistics at each leaf node, spending only a few float variables. Our adaptive splitting solutions achieve the incremental update by storing the sufficient statistics for divergence estimates or the state variables for Adam optimization.

However, it is not trivial to make our scanning algorithms sections 5.2.1 and 5.2.2 incremental. The core of these algorithms is the cost curve, represented as key points with the same number as the sample count. Making this representation incremental requires finding suitable ways to summarize the curves.

A naive way for the incremental update to cache all the sample data arriving at this node. Then we run the scanning algorithm with the union of incoming samples and stored ones. But this soon leads to expensive memory consumption over time.

We finally decided **not** to cache previous samples **nor** to support the incremental update for two reasons:

1. The memory consumption issue explained above.
2. The gain in the split quality by incorporating previous samples is minimal while the increase in the memory cost is substantial.

5.2.5. Defensive Fallback To Maintain Isotropy

None of the algorithms in this section consider requirement 6. In extreme cases, covariance-scanning, information-gain-scanning, and fluence-scanning can propose split pivots very close to the left or right end, or repeating split dimensions that produce thin elongated regions. An example is shown in fig. 5.13 (left). For variance-scanning, however, we didn't observe such issues.

Upon a closer inspection in a 2D experiment fig. 5.14, we ascribe this phenomenon to the "greediness" of the optimization eqs. (5.47), (5.69), (5.72) and (5.74), which strives to select the minimum point of the cost curve, regardless of the magnitude of the cost reduction. To remedy this, let's first introduce a few concepts.

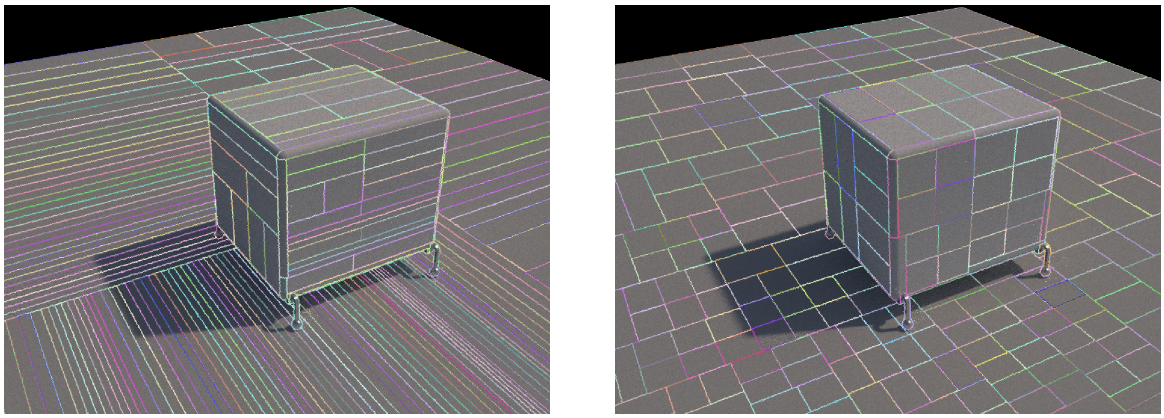


Figure 5.13.: *Information-gain-scanning without (left) vs. with (right) defensive fallback (Cube, 8 SPP)*

Gain of a split. The *parent cost* is defined as the error of samples on the parent region, denoted with $\mathrm{cost}(S)$. Specifically:

$$\mathrm{cost}_1(S) \coloneqq TV(S) \tag{5.82}$$
$$\mathrm{cost}_2(S) \coloneqq \mathrm{Vol}(S) \tag{5.83}$$
$$\mathrm{cost}_3(S) \coloneqq H(S) \tag{5.84}$$
$$\mathrm{cost}_\Phi(S) \coloneqq V_\Phi(S) \tag{5.85}$$

Then, we define the *gain* of a split as the maximum ratio of cost reduction in a certain dimension:

$$\mathrm{gain}(d) \coloneqq \frac{\mathrm{cost}(S) - \min_t \mathrm{cost}(d,t)}{\mathrm{cost}(S)} \tag{5.86}$$

For information-gain-scanning, the definition is a bit different, since the absolute difference of two entropies is more meaningful than the ratio difference, which has a unit of "bit":

$$\mathrm{gain}(d) \coloneqq \mathrm{cost}_3(S) - \min_t \mathrm{cost}_3(d,t) = \max_t \mathrm{IG}(d,t) \tag{5.87}$$

In practice, the gain is always positive. Its magnitude can tell us about the "confidence" of a proposed split. The higher the gain, the more the algorithm believes the split benefits the guiding.

For covariance-scanning, the physical meaning of the gain is the percentage in volume reduction.

Example. To give some intuition about the gain, let's consider a 2D case where sample positions follow a uniform distribution $\mathrm{Unif}([0,a]\times[0,b])$. The covariance $\Sigma = \mathrm{diag}(\frac{1}{12}a^2, \frac{1}{12}b^2)$. Hence, the parent cost is $\sqrt{\det\Sigma} = \frac{1}{12}ab$. Apparently, splitting at the center, either at $x = a/2$ or $y = b/2$, the cost function is minimized at $\frac{1}{2}\cdot\frac{1}{12}\frac{a}{2}b + \frac{1}{2}\cdot\frac{1}{12}a\frac{b}{2} = \frac{1}{24}ab$. So the gains for both dimensions are 0.5. A wise choice would be to select the dimension with the longer extent. However, our way of greedily selecting the dimension with the highest gain will produce an unpredictable dimension.

This example tells us, that for covariance-based methods, when the gains across dimensions are similar, we should use a more conservative dimension selection policy. We resort to Open

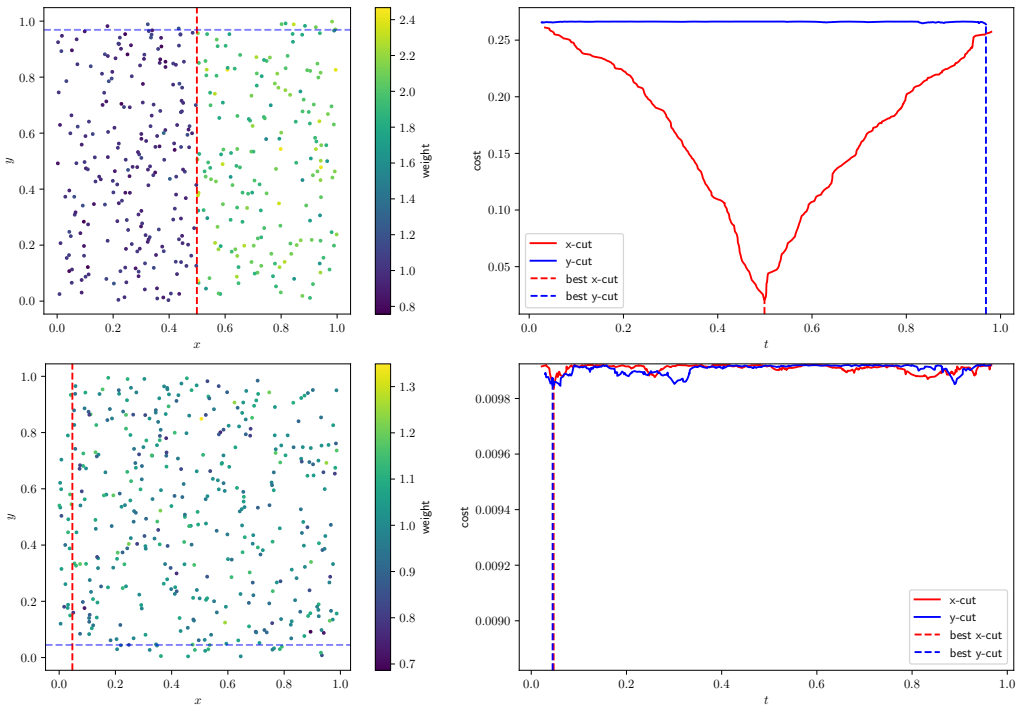


Figure 5.14.: *The stability issue of fluence-scanning in 2D experiments. Samples are uniformly generated in the unit square. Upper row: when sample weights correlate with x, the best pivot significantly reduces the variance. Lower row: when sample weights are independent of the spatial location, the cost doesn't change much. Fluctuation in the cost curve misleads the algorithm in finding the wrong pivot or dimension.*

PGL's dimension strategy eq. (5.1) that at least maintains the isotropy, while we still use the split positions suggested by the scanning algorithms.

Specifically, for covariance- and information-gain-scanning, we switch to the defensive strategy when the gain difference across dimensions is no greater than a threshold:

$$(d^*, t^*) = \begin{cases} \arg\min_{d,t} \operatorname{cost}(d,t) & \Delta\text{gain} > g_{\text{ths}} \\ (d_0, \arg\min_t \operatorname{cost}(d_0, t)) & \text{otherwise} \end{cases} \tag{5.88}$$

where $\Delta\text{gain} = \max_d \text{gain}(d) - \text{secondmax}_d \text{gain}(d)$. The optimization constraint $\min(N_l, N_r) \geqslant N_{\min}$ is omitted. The defensive split dimension d_0 is given by eq. (5.1).

The gain ratio threshold g_{ths} controls the **defensiveness**. The higher, the more likely it falls back to Open PGL's strategy. For covariance-scanning, the gain is in $[0, 1]$, so $g_{\text{ths}} \in [0, 1]$. For information-gain-scanning, however, the gain and the threshold have no upper bound.

For fluence-scanning, the handling is different. The meaning of gain is the reduction ratio of the variance in the fluence. We notice that when the gain is small, the proposed split pivot can be very unstable, as shown in fig. 5.14. So we trigger fallbacks with the value of the gain, and resort to not only use Open PGL's dimension, but also the pivot.

$$(d^*, t^*) = \begin{cases} \arg\min_{d,t} \operatorname{cost}_\Phi(d,t) & \text{gain}(d^*) > g_{\text{ths}} \\ \sim (5.1), (5.2) & \text{otherwise} \end{cases} \tag{5.89}$$

The gain threshold $g_{\text{ths}} \in [0, 1]$.

Based on our experiences, we recommend setting the defensiveness thresholds for covariance-scanning, information-gain-scanning, and fluence-scanning to 0.2, 1.0 bit, and 0.03, respectively. An improved result with the defensive fallback is shown in fig. 5.13 (right).

Results and analysis for "where to split" can be found in sections 6.1.2 and 6.2.2.

5.3. Full Model

So far, we have presented the main components of our spatial subdivision algorithms. To determine "when to split", we have the maximum sample count threshold, divergence reduction threshold, and MIS weight threshold. For "where to split", we utilize variance-scanning, covariance-scanning, information-gain-scanning, and fluence-scanning.

Our complete spatial subdivision schemes combine some of the most performant modules according to our experiment results in section 6.2. We list some of the best practices in the following.

KL divergence + information-gain-scanning with defensive fallback

$$\text{Split if} \quad \left(N_{\text{valid}} \geqslant 2N_{\min} \wedge \langle \tilde{D}_{\text{KL}}(q_p) \rangle - \langle \tilde{D}_{\text{KL}}(q_c) \rangle > D_{\text{ths}}\right) \tag{5.90}$$

$$\vee\, N_{\text{valid}} \geqslant N_{\max} \tag{5.91}$$

$$\text{at } (d^*, t^*) = \begin{cases} \arg\max_{d,t} \text{IG}(d,t) & N_{\text{cur}} \geqslant 2N_{\min} \wedge \Delta\text{gain} > g_{\text{ths}} \\ (d_0, \arg\max_t \text{IG}(d_0, t)) & N_{\text{cur}} \geqslant 2N_{\min} \wedge \Delta\text{gain} \leqslant g_{\text{ths}} \\ (d_0, t_0) & N_{\text{cur}} < 2N_{\min} \end{cases} \tag{5.92}$$

$$d_0 = \arg\max_d V(S_{\text{valid}})_d \tag{5.93}$$

$$t_0 = (\overline{\mathbf{x}}_{\text{valid}})_{d_0} \tag{5.94}$$

where eq. (5.90) is the adaptive threshold based on KL divergence. eq. (5.91) enforces the split to escape the local optima of the adaptive one and also to encourage more frequent refinement at the beginning of the subdivision. As a defensive backup policy, $N_{\max}$ is set to be way larger than Open PGL's maximum sample count threshold.

Eq. (5.92) is the information-gain-scanning algorithm with a gain-based dimension selection strategy. Note that N_{cur} is the number of samples in the current recent training iteration, which is no greater than N_{valid}, the number of valid samples **accumulated** at this node, as we do not store previous samples (see section 5.2.4). This is why there can still be the case of having insufficient input samples to run the scanning algorithm. As a result, we fully fall back to using Open PGL's split strategy.

KL divergence + variance-scanning with defensive fallback

Eq. (5.92) can be replaced by

$$\text{Split at } (d^*, t^*) = \begin{cases} \arg\min_{d,t} \text{cost}_1(d,t) & N_{\text{cur}} \geqslant 2N_{\min} \\ (d_0, t_0) & \text{otherwise} \end{cases} \tag{5.95}$$

KL divergence + fluence-scanning with defensive fallback

Eq. (5.92) can be replaced by

$$\text{Split at } (d^*, t^*) = \begin{cases} \arg\min_{d,t} \text{cost}_\Phi(d,t) & N_{\text{cur}} \geqslant 2N_{\min} \wedge \text{gain} > g_{\text{ths}} \\ (d_0, t_0) & \text{otherwise} \end{cases} \tag{5.96}$$

As explained in section 5.2.5, the $\text{gain} > g_{\text{ths}}$ condition deals with the instability of fluence-scanning. This strategy addresses practical requirement 3 with both when- and where-to-split policies.

With the defensive fallbacks, each of our full models is capable of addressing all of our practical requirements in section 4.4.

Please see section 6.2.3 for a quantitative evaluation of our full models.

6

Experiments and Analysis

This chapter provides a comprehensive evaluation and analysis of the models proposed in chapter 5. We first experiment with each module to understand their behavior in detail and get insights into their strengths and weaknesses. Then we implement these methods in the PBRT renderer with Open PGL, conduct rendering experiments, and demonstrate their performances both qualitatively and quantitatively.

6.1. 2D Experiments

6.1.1. Adaptive Subdivision

We validate the divergence-based adaptive split decision (section 5.1.2) on the same 2D setups as section 4.3. Candidate splits are set near the theoretical optimal locations. We repeat this process: generate MC samples, evaluate the models with our KL or χ^2-divergence estimators, and update the parent and child guiding distributions[1]. We then compare this evaluation result ("split" or "not split" at each iteration) with the theoretical ground truth.

Specifically, for the generation step, 200 MC samples with data $s_i = (x_i, \omega_i, q_s(\omega_i \mid x_i), \langle L(x_i, \omega_i)\rangle)$ are generated. Positions and directions are sampled uniformly. Gamma-distribution (to ensure positiveness) noise is added to the observed radiance values. The gamma distribution parameters are set in such a way that the mean equals to $L(x_i, \omega_i)$ and the variance equals to 10^{-2}.

We use weighted EM and two-lobe Gaussian mixtures to fit the guiding distributions with the samples $(\omega_i, \langle \Phi(x_i)\rangle = \langle L(x_i, \omega_i)\rangle / q_s(\omega_i \mid x_i))$. In particular, the parent model q_p has access to all training samples. The left and right children q_l, q_r see only samples in their half space. The "child model" q_c is defined in eq. (5.7).

[1] Why evaluating before fitting? See the **Implementation Details** part in section 5.1.2

For the evaluation of the parent and child models in each iteration, we compute their divergence estimates using MC samples with eqs. (5.19), (5.20), (5.24) and (5.25) by accumulating their sufficient statistics from the first iteration, with an exponential decay ratio of 0.5. To validate this result, we compute the ground truth cross-entropy and χ^2-divergence by quadrature using eqs. (5.14) and (5.21), and compare the estimated values with the ground truths.

In total, 20 iterations of evaluation, generation, and training are launched.

Setup 1

The radiance field setup, trained parent and child models, and their divergence curves are shown in fig. 6.1. The "gain" is defined as the reduction of divergence by adopting the child model. It is the parent divergence subtracted by the corresponding child one. A gain above the threshold 0 means the tendency to adopt the candidate split, otherwise not.

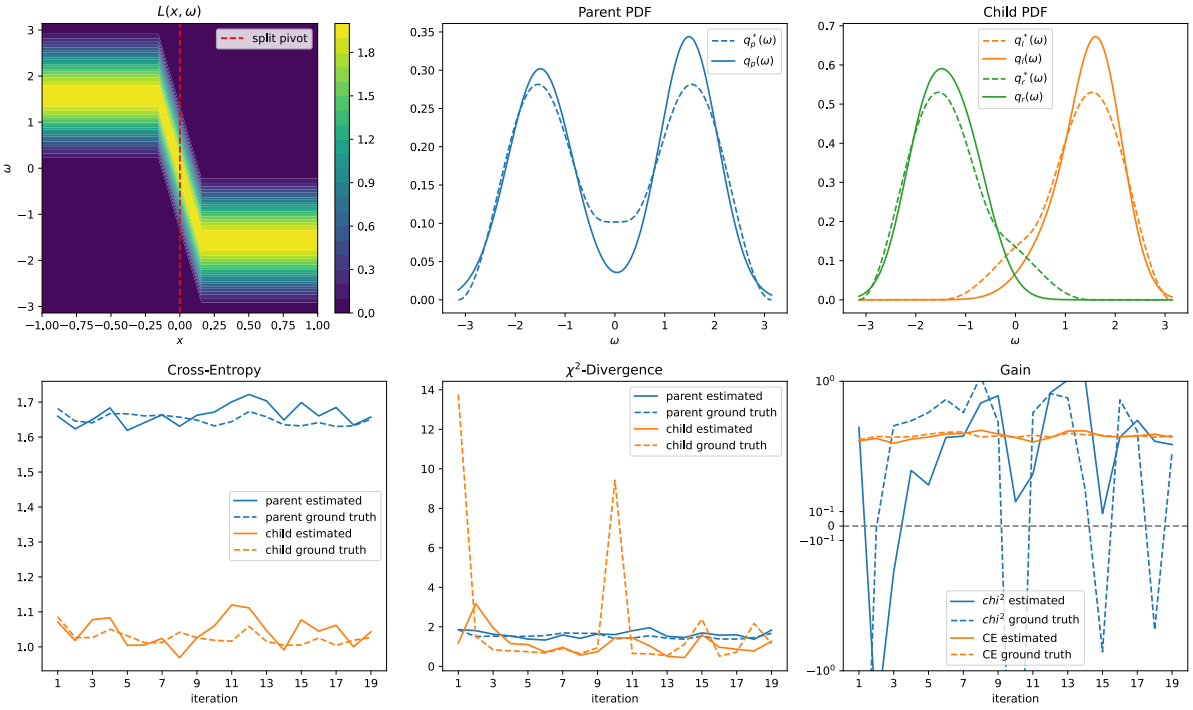


Figure 6.1.: *Divergence metric experiment (1). First row: the radiance field, the parent model, and the child model after 10 iterations. Second row: the cross-entropy, χ^2-divergence, and the gain over iterations.*

In this case, the gains are most positive all the time, indicating strong confidence in the split, which agrees with our intuition that cutting in the middle can reduce the signal complexity.

By comparing the ground truth curves with the estimated ones, we can conclude that both metrics make a fair approximation. The cross-entropy suffers from less noise than the χ^2-divergence. Interestingly, the ground truth χ^2-divergence also exhibits large fluctuations, revealing its sensitivity to the distribution change over iterations.

Setup 2

In this case, the signals are constant to x. Splitting in the middle will not improve the representation of the guiding field. Even worse, because child models have access to fewer samples, the noise in the fitting is more than the parent's. This fact is perfectly captured in the result fig. 6.2, where the gain curves are almost always negative. By examining the parent and child PDFs, we can see the child model approximates the optimal distribution slightly worse than the parent.

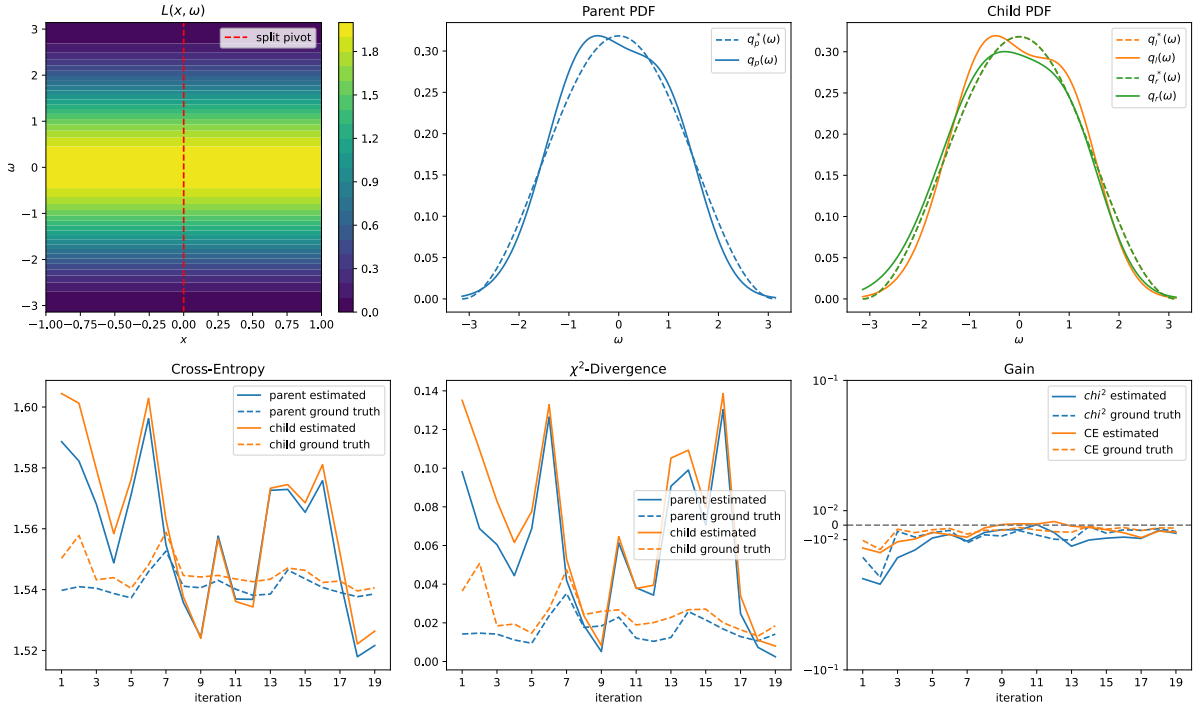


Figure 6.2.: *Divergence metric experiment (2)*

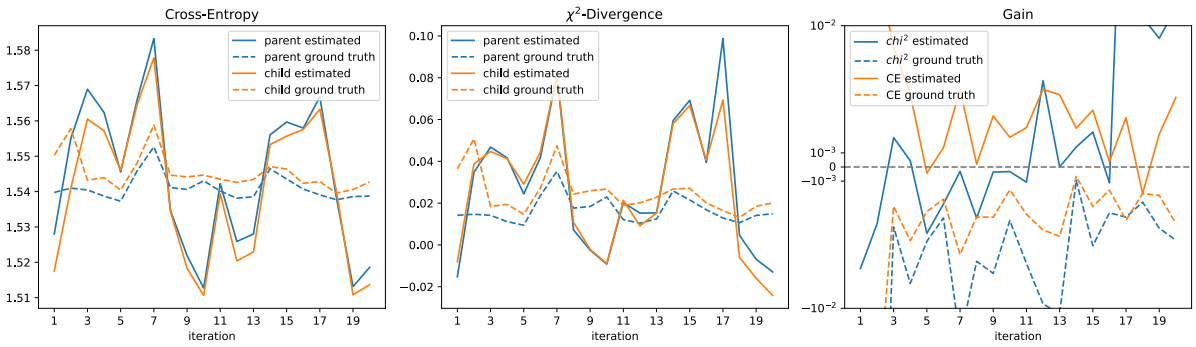


Figure 6.3.: *Problems with evaluating after the training (Experiment 2).*

We also experimented with swapping the order of "evaluation, generation, training" to "generation, training, evaluation". In the results fig. 6.3, the child divergences tend to be more underestimated than the parent's. In the gain curve plot, the estimated gains tend to take the positive side, contrary to the negative ground truths.

This finding supports our proposal of delaying the evaluation to the beginning of the next iteration fig. 5.3.

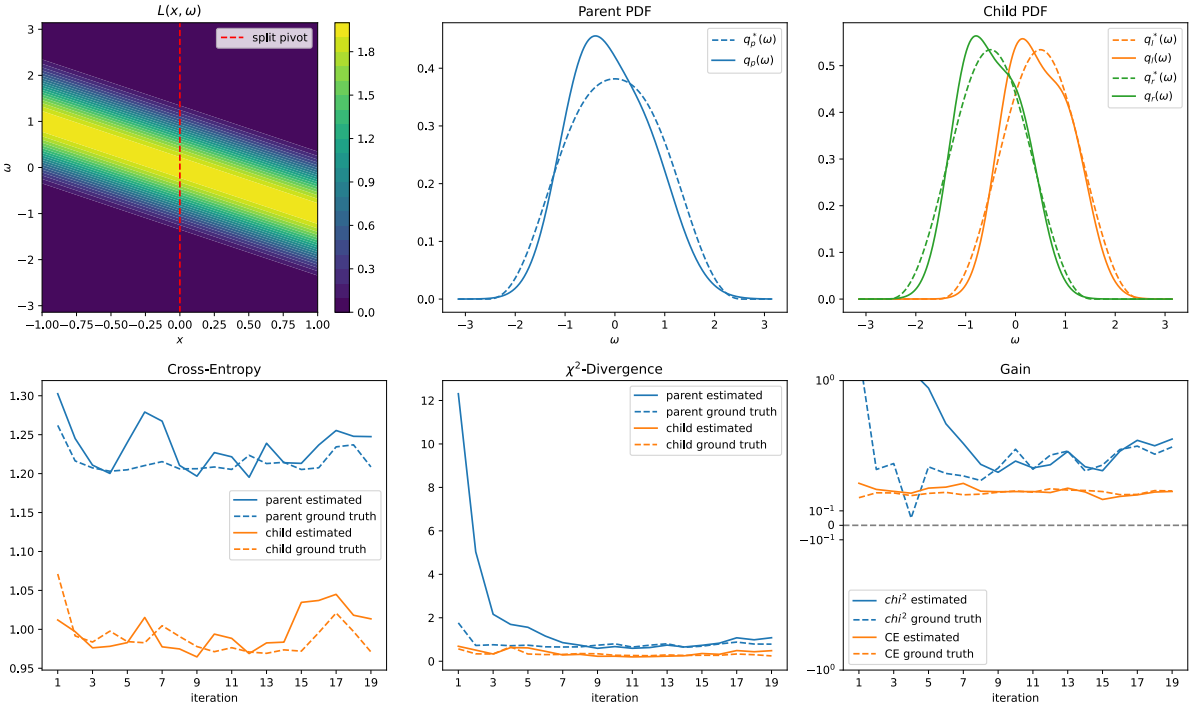


Figure 6.4.: *Divergence metric experiment (3.1)*

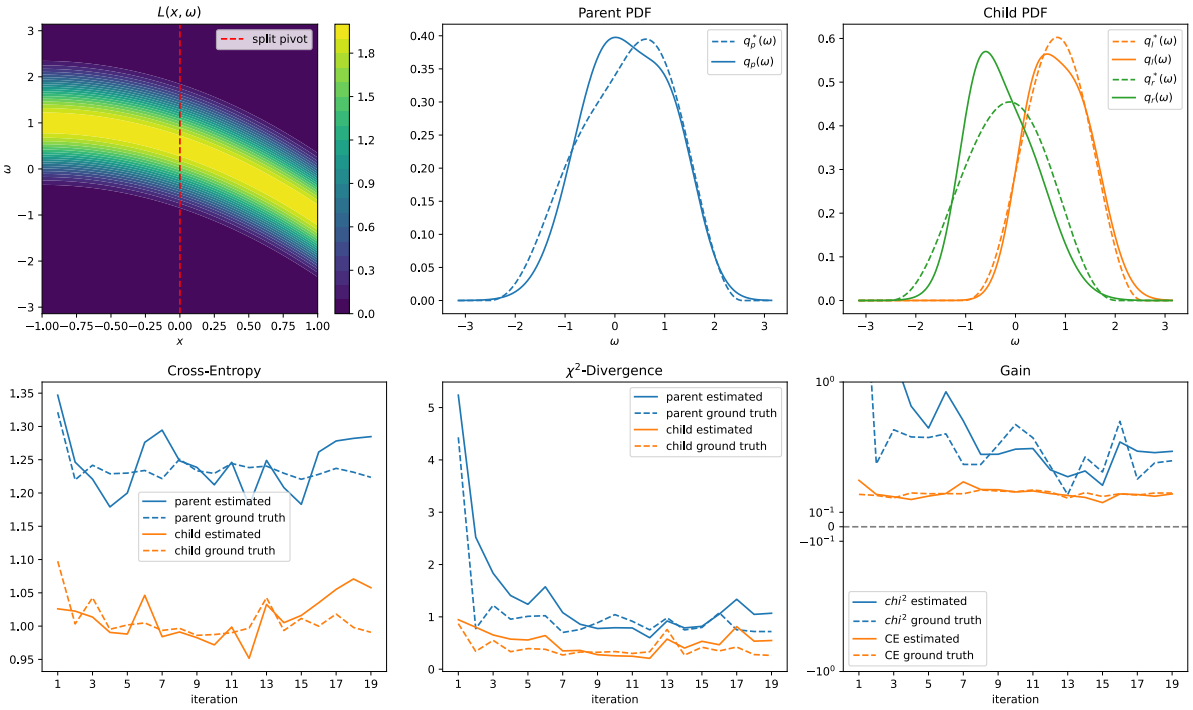


Figure 6.5.: *Divergence metric experiment (3.2)*

Setup 3.1

The spatial dependency of the radiance field makes the child model stand out in both of our metrics, yielding a consistently positive gain, shown in fig. 6.4.

Setup 3.2

When the spatial dependency is nonlinear, the finding is the same, shown in fig. 6.5.

Setup 3.3

This case is similar to the second setup, where the child performs worse due to fewer available samples.

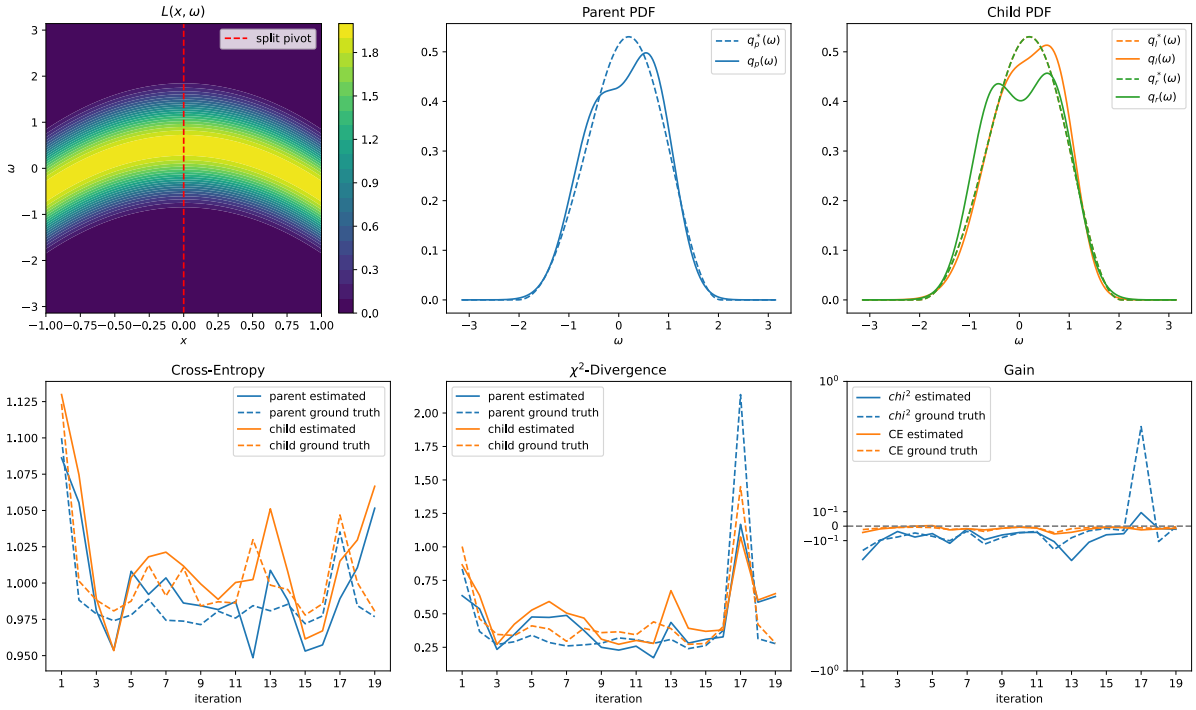


Figure 6.6.: *Divergence metric experiment (3.3)*

Setup 4

Thanks to the split, the child can represent the sunlight and ambient separately. The cross-entropy curve captures this advantage. However, the χ^2-divergence fails due to its increased sensitivity and noise.

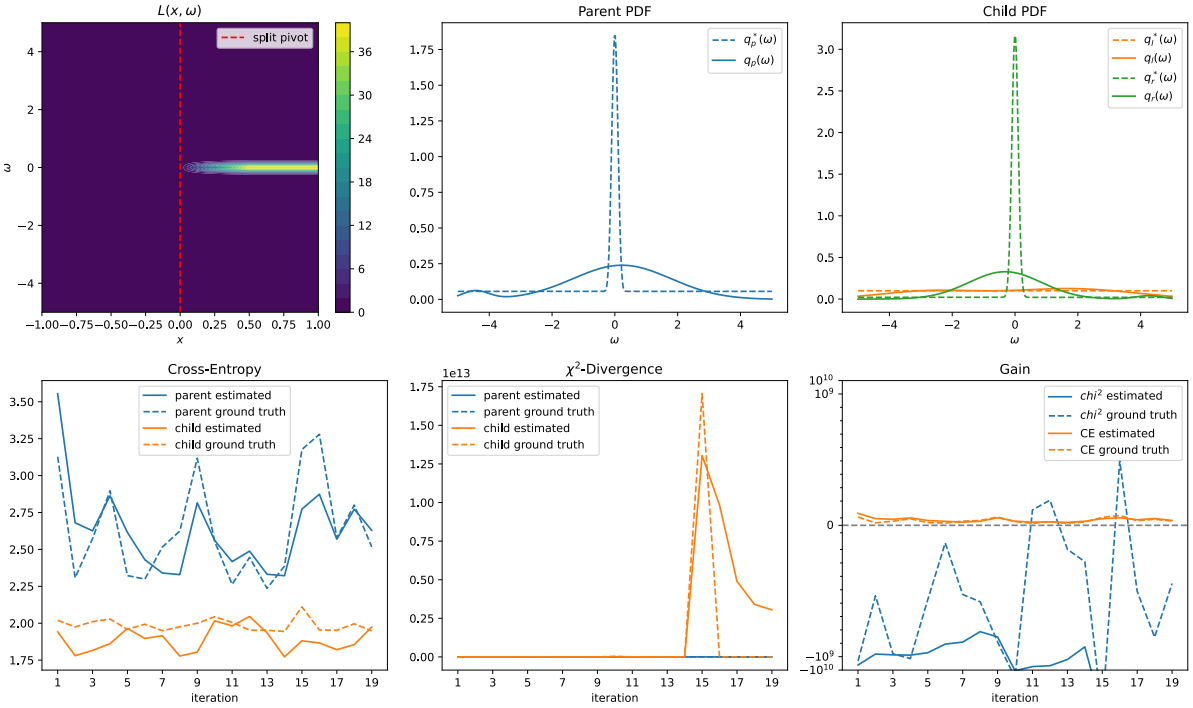


Figure 6.7.: *Divergence metric experiment (4)*

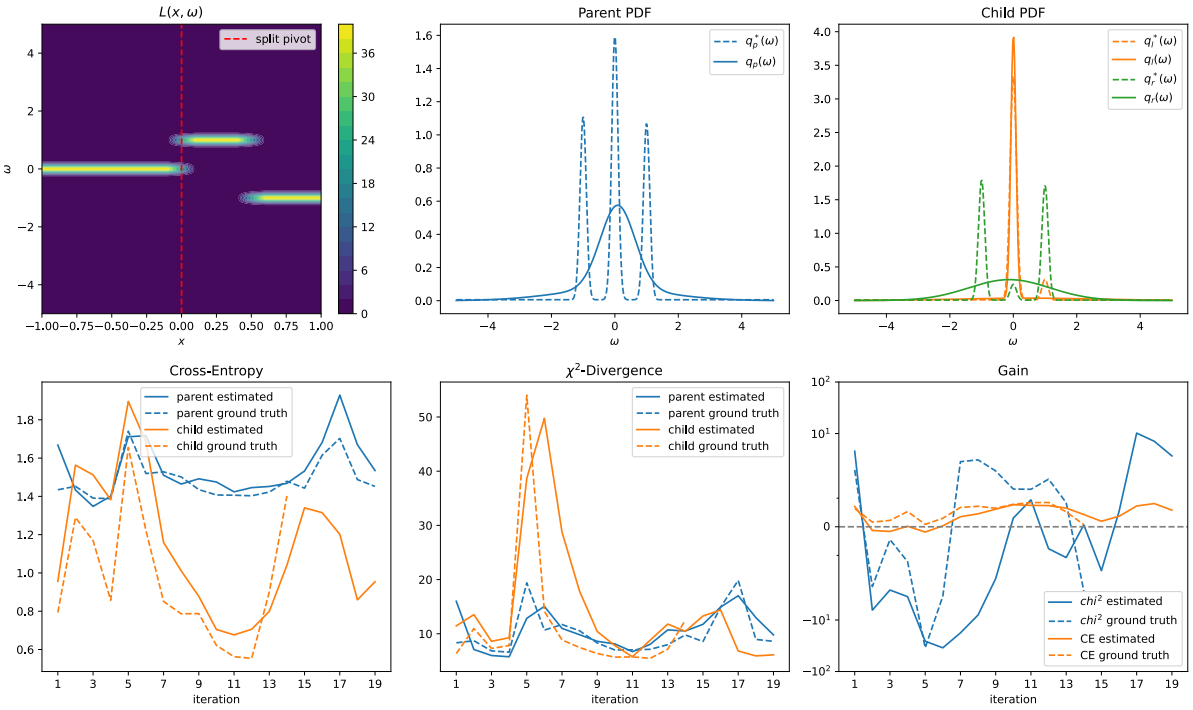


Figure 6.8.: *Divergence metric experiment (5)*

Setup 5

In this complex lighting case, the child model still approximates the signals better than the parent, if not perfectly. Again, χ^2-divergence suffers from fast oscillation. Fortunately, the

cross-entropy curves are stable enough to highlight the gain of the split.

Conclusion. From all of these experiments, by inspecting the gain curves with a focus on the signs, we conclude that our estimators make split decisions that are consistent with the ground truths, although they exhibit higher levels of noise.

We find out that KL divergence, or cross-entropy, is a more stable metric than χ^2-divergence. Our cross-entropy estimator also manifests less bias from the ground truth compared to χ^2-divergence. This, again, aligns with our discussion in section 5.1.2.

6.1.2. 2D Geometry-Aware Subdivision

In this section, we test our scanning algorithms that tackle the "where to split" problem with some fabricated 2D scenes. The input in these experiments is 500 position-only samples generated from mixtures of Gaussian or uniform distributions with diverse shapes. We hope these 2D examples can emulate some typical local subdivision scenarios in real rendering scenes.

We will examine the cost curves eqs. (5.54), (5.62) and (5.71) obtained from applying variance-scanning, covariance-scanning, and information-gain-scanning to the input samples in each dimension, to understand their behaviors in detail.

The results are displayed in figs. 6.9 and 6.10. In each row, we visualize the sample distribution and the best splits proposed by each algorithm in the left image, and show the cost curves along x- and y-axis in the right three images. The best split of each algorithm is defined as the pivot and dimension with the lowest cost eq. (5.47).

From these results, we can see all scanning algorithms aim to reduce the average amount of dispersion of samples with an axis-aligned cut. All algorithms tend to split in the center, if samples are somewhat uniformly distributed (see *Uniform* and *Uniform Triangle*).

If there are multiple "objects" in the scenes, in cases where their extents in each dimension are similar (*2 Gaussians* and *5 Gaussians*), all algorithms propose a split near the gaps between these objects, where each gap appears as a local minimum on the cost curve.

If there is a significant discrepancy in their extents, we find that variance-scanning and covariance-scanning sometimes fail to identify the objects and propose a cut in the longest dimension (*Thin Gaussians* and *3 Gaussians*). The reason for this is explained in fig. 5.8. Information-gain-scanning and covariance-scanning avoid this issue in *Thin Gaussians* by measuring the gain resulting from volume reduction. However, in *3 Gaussians*, covariance-scanning still proposes an x-cut (green) because of a slightly higher volume reduction compared to the y-cut. Only information-gain-scanning selects the expected y-cut. We attribute this to its entropy-based metric eq. (5.71), which makes more sense with the weighted-sum operation than the volume eq. (5.62).

Segments and *T* emulate scenes with closely-contacted geometries. Under our geometry assumption introduced in section 5.2.1, an accurate split at the contact point would best separate the geometries for guiding. The results magnify the difference between variance-scanning and the two covariance-based methods. Variance-scanning's distance-based metric makes it

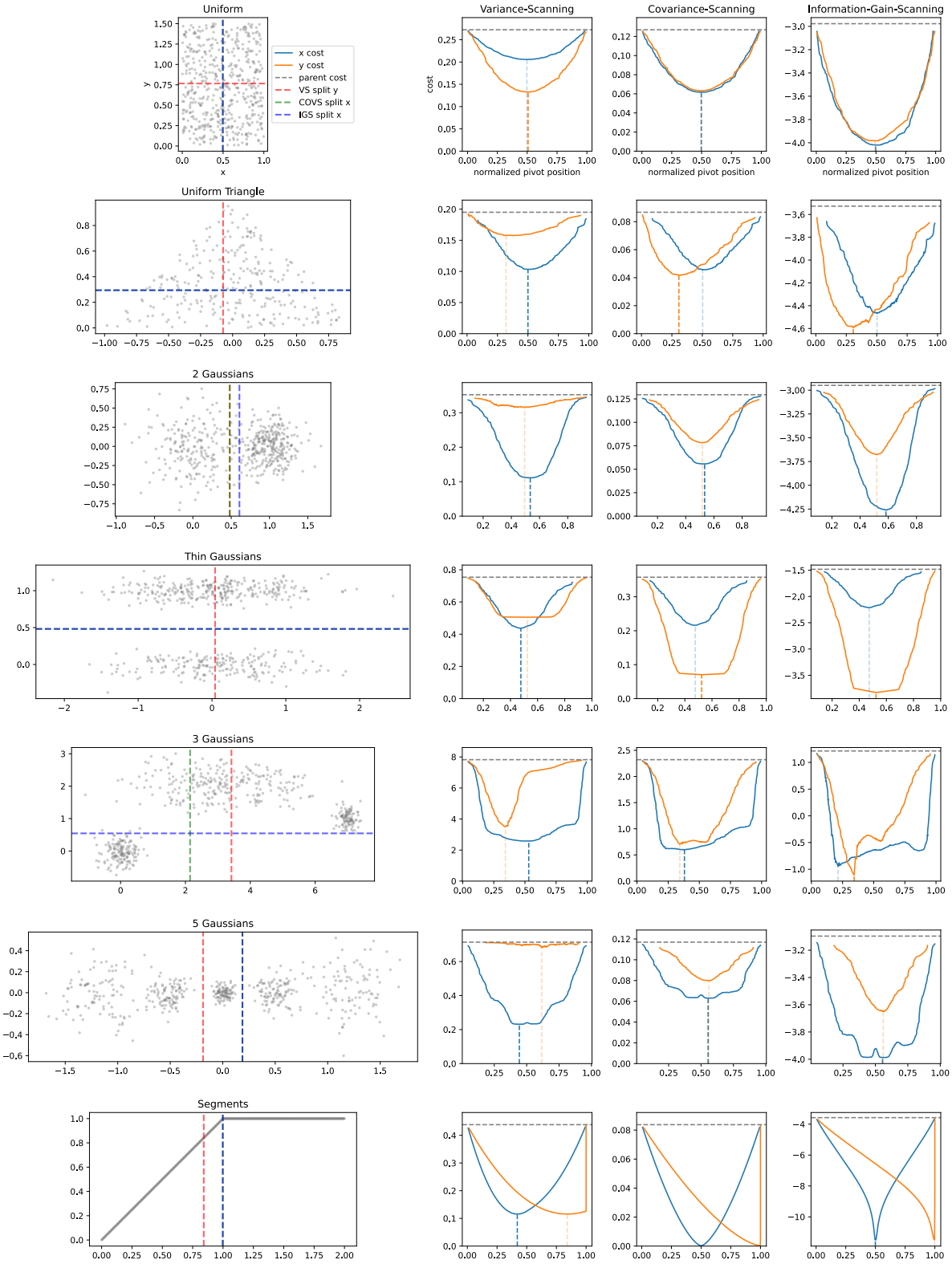


Figure 6.9.: *2D experiments with our scanning algorithms (a)*

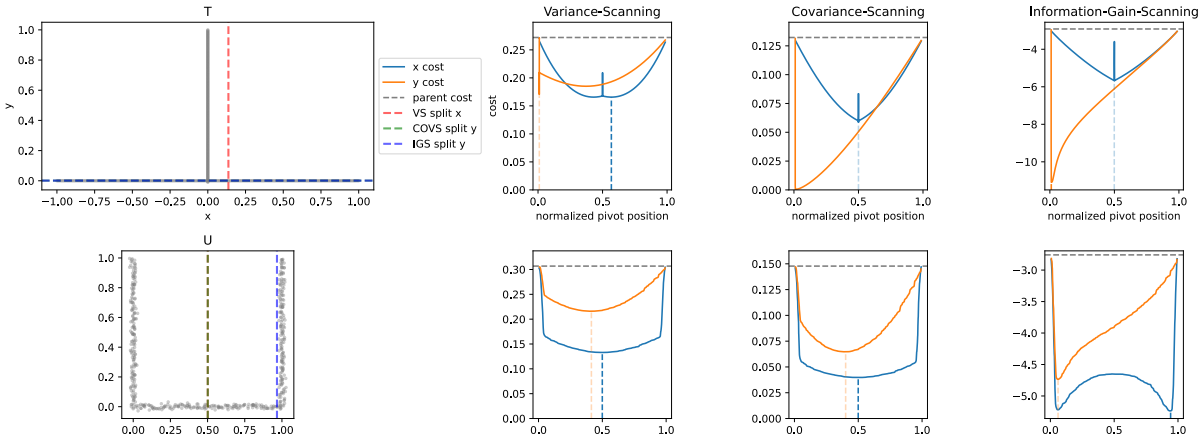


Figure 6.10.: *2D experiments with our scanning algorithms (b)*

favor the proximity assumption more, producing splits not accurately occurring at the joints. Covariance- and information-gain-scanning, on the other hand, perfectly detect the joint location that results in the most volume reduction (to almost zero). In the final *U* scene, both variance- and covariance-scanning prefer an x-cut at the center. Only information-gain-scanning selects the pivot near the left or right vertical lines. One would imagine if the subdivision is continued recursively, information-gain-scanning can sharply distinguish all the lines and keep them intact while the other two chop off the middle line in the beginning. Again, we ascribe information-gain-scanning's aggressive feature to its metric formation.

However, two failure cases (*Uniform* and *Uniform Triangle*) are found for covariance- and information-gain-scanning where they are prone to select a shorter axis, compromising the isotropy requirement. Through a closer look at the cost curves, we immediately know the reason: the cost reductions (gains) for both axes are identical, making them indistinguishable! This issue arises when the points are up to a scale-transform when rotating 90 degrees. The good news is we can easily contour it by enforcing the selection of the longest axis when the gains along different axes are similar, which indicates the algorithm's indecision about the dimension. This aligns with our fallback handling in section 5.2.5.

We conclude that the proposed scanning algorithms can, to some extent, infer underlying geometries from sample points in an unsupervised manner. Information-gain-scanning generates splits most aggressively for the goal of separating geometries, while variance-scanning is the most conservative one. There needs to be careful handling of their robustness and stability when applying these algorithms to rendering applications.

6.2. Rendering Experiments

Our implementation of the proposed subdivision schemes is done on PBRT-v4 [PJH23] and Open PGL [HD22]. The test environment is on Ubuntu 20.04, with an eight-core CPU and 32GB memory.

We select a diverse set of scenes, categorized into three groups: planar, small, and large.

In *planar scenes* fig. 6.11, the camera is looking at a plane illuminated by a sky dome. Floating geometries drop various shapes of shadows on the plane. Indirect illumination is turned off so that the guiding samples concentrate only on the plane. This makes these scenes suitable for testing spatial subdivisions' adaptivity to lighting.

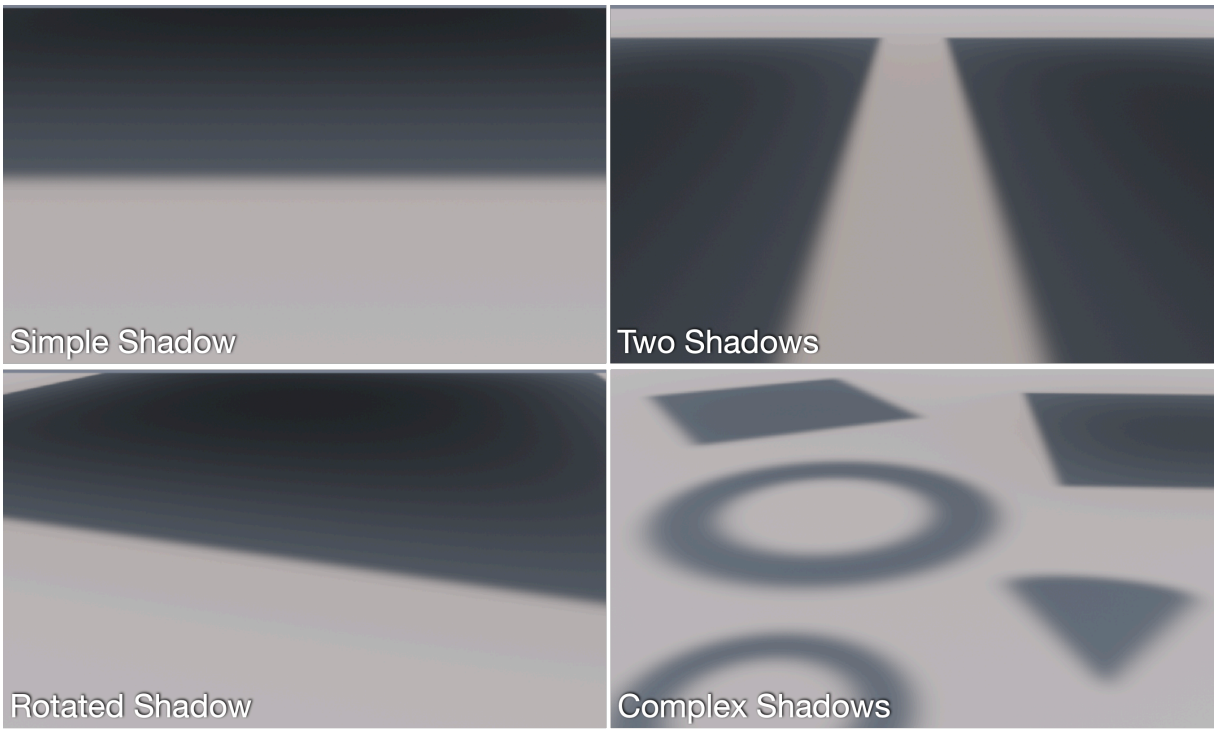


Figure 6.11.: *Planar scenes*

Small scenes (fig. 6.12) consist of simple geometries (cubes, planes, etc.), containing some shadow and caustics effects. These scenes are designed to facilitate the understanding and analysis of spatial subdivision.

Large scenes fig. 6.13 are mainly selected from commonly used benchmarks for testing path guiders' performance. *Pool*, *Villa*, *Living Room*, and *Kitchen* contain intricate indirect illumination. Specifically, we select *Classroom* and *San Miguel* scene variants, which feature complex geometries, to evaluate our geometry-aware solutions.

To emphasize the guider's performance, **we disable next-event estimation**. We enable defensive sampling eq. (2.16) with a BSDF sampling rate of 0.5. We turn off stochastic query[2] to

[2] A technique to perturb a query sample to a point nearby, in order to propagate guiding information to neighboring regions [VHH+19].

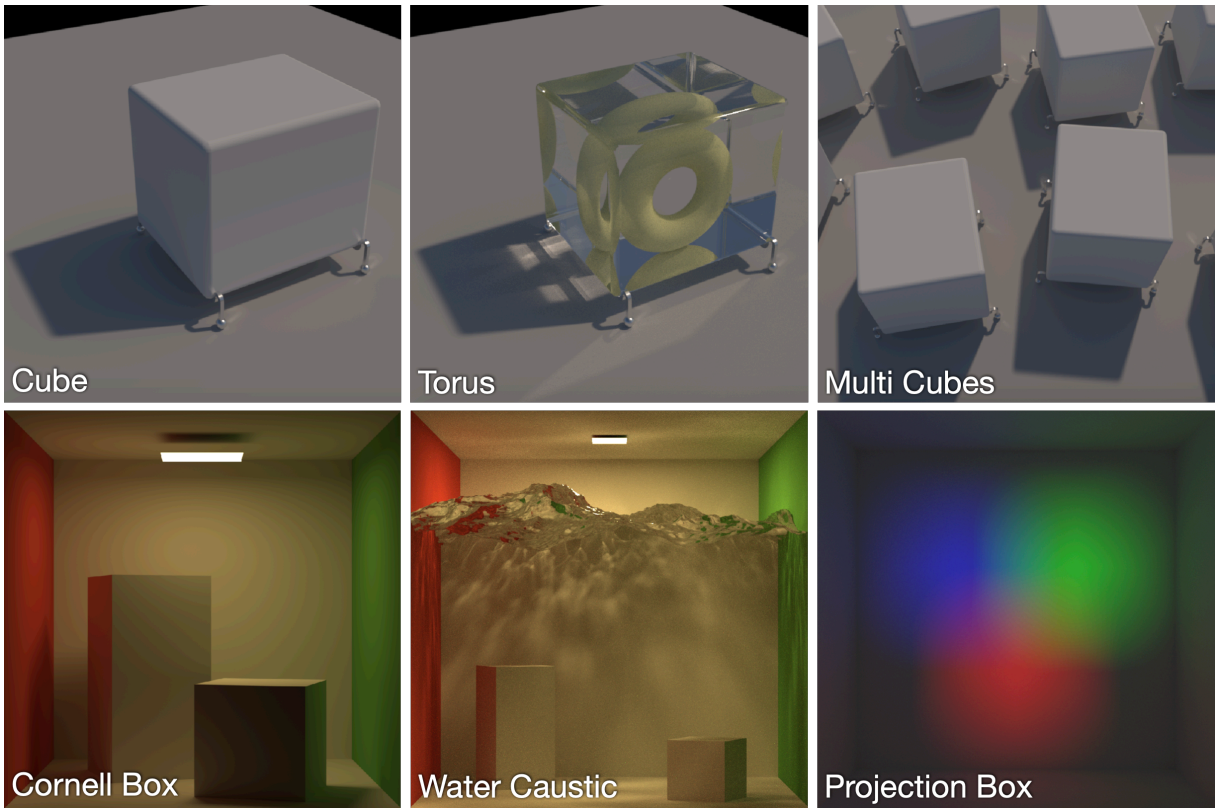


Figure 6.12.: *Small scenes*

attribute guiding cache issues to spatial subdivisions more easily.

To further enhance the comparisons between different subdivisions, the rendering of each scene is divided into two passes. In the first pass (*training pass*), we alternate rendering and training of the guiding cache in the way described in fig. 4.1, until a total n_t samples per pixel (SPP) has been depleted. In the second pass (*evaluation pass*), we freeze the guiding cache, and execute the guided rendering for n_e SPP. The final rendering result is the average of pixel estimates from only the evaluation pass.

For brevity in our following discussions, we refer to Open PGL's subdivision scheme as the **baseline**, which we compare all our models against.

First, we will validate each of the modular algorithms individually. Then, we will conduct large-scale experiments with our complete models. For an overview of our models' performance, please refer to fig. 6.49, which summarizes the results across all scenes .

6.2.1. When to Split

In this experiment, we compare our adaptive subdivision schemes with the baseline that relies on a static sample count threshold. As for the split position and dimension selection, all methods share the baseline strategy eqs. (5.1) and (5.2).

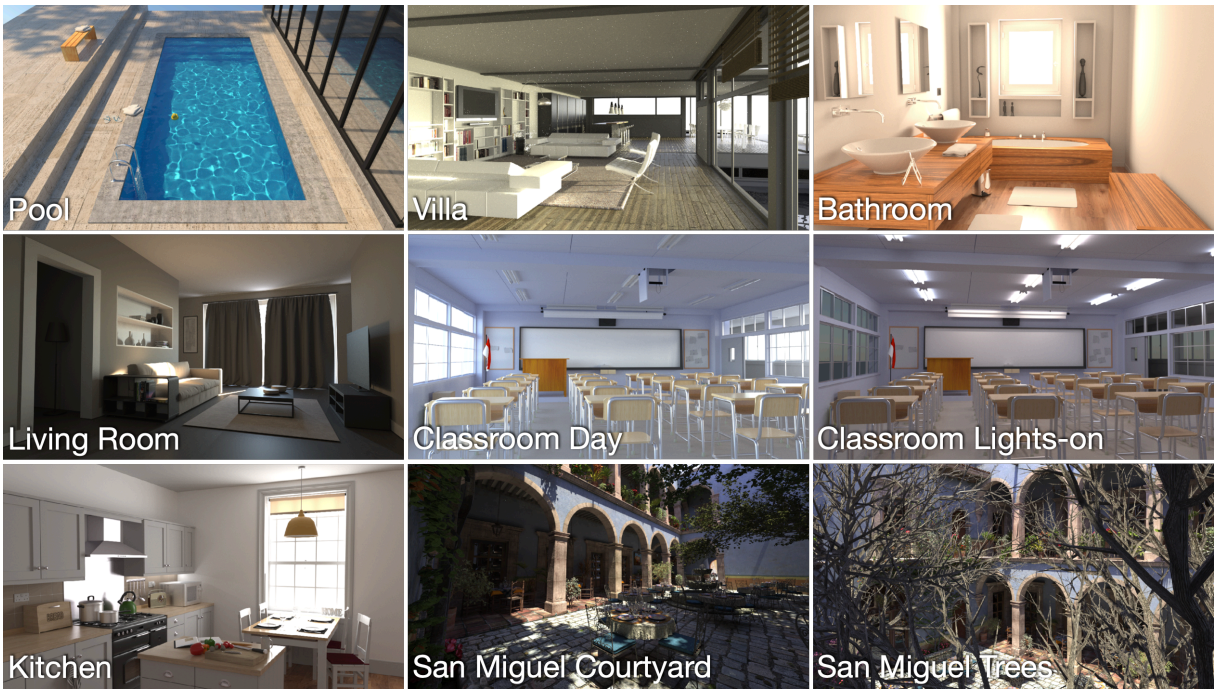


Figure 6.13.: *Large scenes*

Adaptive Splitting Based on Divergence Estimate

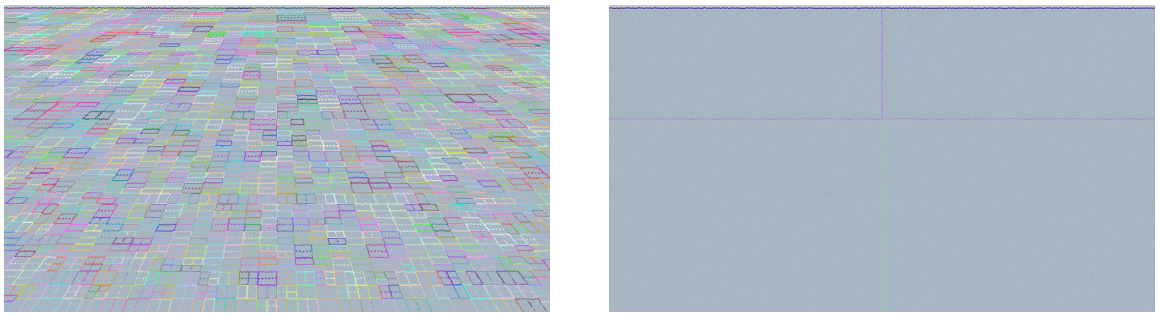
We tested our solution of section 5.1.2 on the planar scenes and small scenes for a qualitative study. The parameters of the adaptive splitting are set as follows:

- The maximum sample count threshold is initially turned off. I.e. $N_{\max} = \infty$.
- The minimum samples per node threshold $N_{\min}$ is set to 1k.
- The divergence threshold D_{ths} is set in the range of $[0, 0.1]$.
- The decay ratio for the sufficient statistics $\lambda = 0.25$.

The baseline's sample count threshold is set to $N_{\max} = 32k$ by default unless otherwise specified. The maximum path tracing depth is set to 1 for planar scenes and 20 for the rest. The training pass SPP n_t is set to 16. For planar scenes, the evaluation pass SPP $n_e = 16$, while for other scenes it's set to 32 for a less noisy image.

Noise in the fitting and evaluation. Let's first validate that our adaptive splitting does not generate over-refinements in a simple setup. The *Plane* scene is uniformly lit by a sky dome without any shadow. Results in *Plane* are shown in fig. 6.14.

Theoretically, this should correspond to the case of fig. 6.2 where splitting will do us no good but reduce the available training samples. However, in fig. 6.14 (left), we still observe quite a few subdivisions when the cross-entropy (CE) threshold is set to 0. We attribute this to the fluctuation of light samples $\langle \Phi_i \rangle$ arriving in each region, caused by the random sampling of the environment light. This fluctuation can result in small differences between the parent's and children's guiding caches, even when their ground truth lighting distributions are the same.

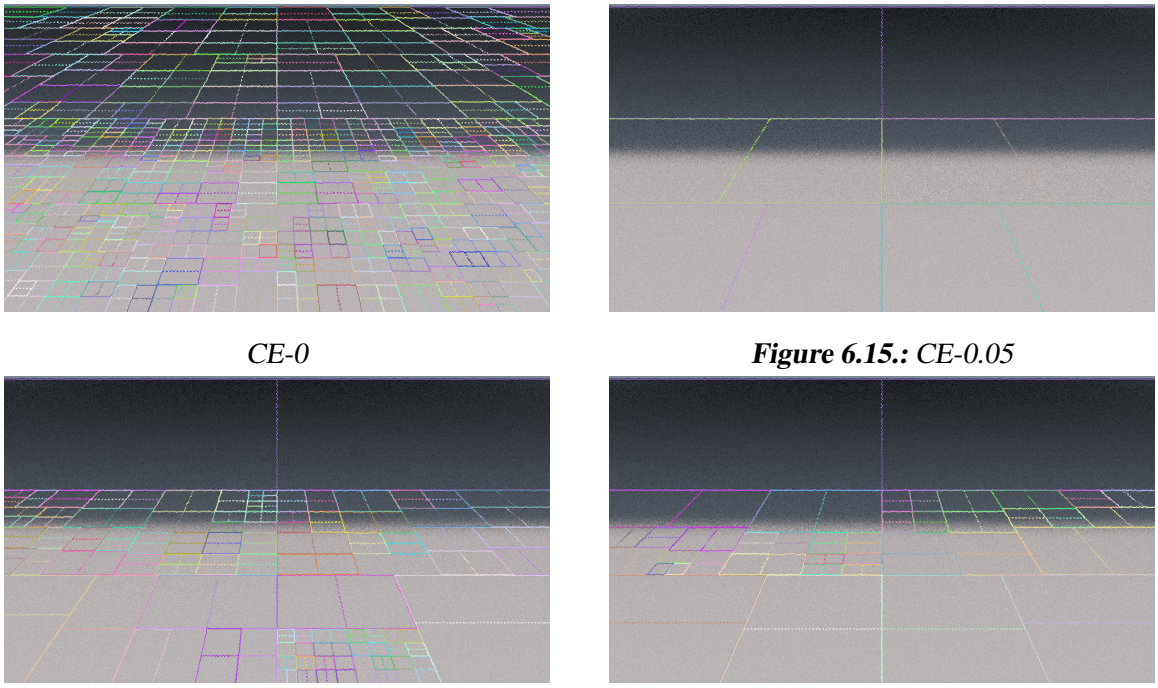


CE-0 *CE-0.05*

Figure 6.14.: *Cross-entropy-based (CE) subdivision with different thresholds in Plane. E.g. "CE-0.05" means splitting when the cross-entropy reduction exceeds 0.05. Dashed lines indicate the candidate splits.*

It is possible that at some iteration, the child model performs slightly better than the parent one, leading to a split. Furthermore, the cache evaluation relies on MC estimation, which is inherently subject to noise.

It is due to both of the two types of noise, that the resulting divergences can lie about the actual cache quality. A positive threshold D_{ths} can effectively contour this effect. After increasing the cross-entropy reduction threshold to 0.05, most unnecessary splits are gone (fig. 6.14 (right)).



CE-0 ***Figure 6.15.:*** *CE-0.05*

χ^2*-0* χ^2*-0.05*

Figure 6.16.: *Divergence-based adaptive subdivision in Shadow*

In the *Shadow* scene fig. 6.16, we show the subdivision using cross-entropy or χ^2-divergence with different thresholds. Again, noise in the training and evaluation appears as some unnecessary splits at the upper fully shadowed or lower fully bright regions in the left images of

fig. 6.16. From the left-and-right comparison, it follows that a higher threshold leads to a coarser subdivision, where over-refinements in the uniform areas are gradually filtered out, leaving the necessary splits near the shadow boundaries.

Another interesting finding is the difference between CE and χ^2-divergence. The former creates more uniform splits across the scene, whereas the latter displays more spatial fluctuation. We attribute this to the sensitivity of the χ^2-divergence formation and the extra bias introduced in its estimator.

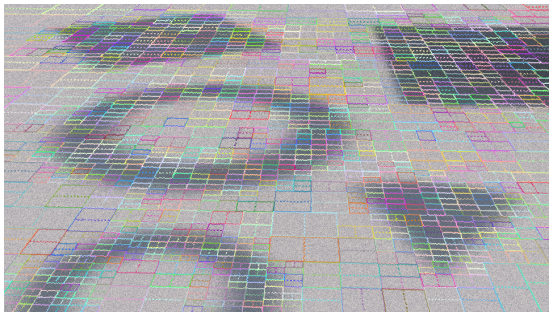


Figure 6.17.: *CE-0*

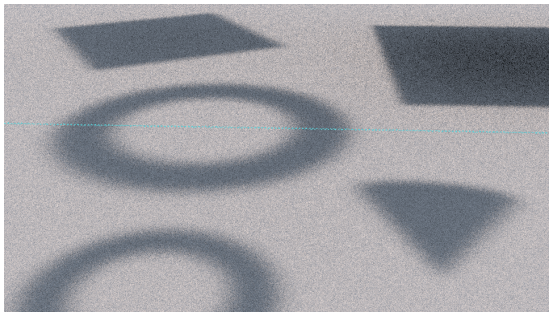


Figure 6.18.: *CE-0.01*

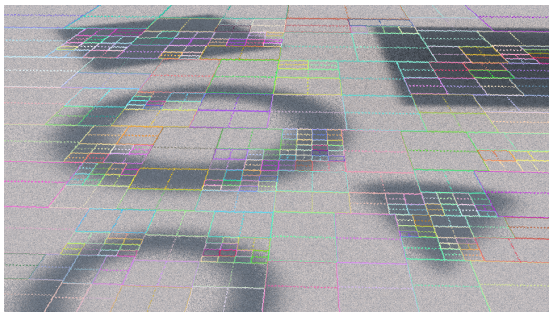


Figure 6.19.: *CE-0.01+128k*

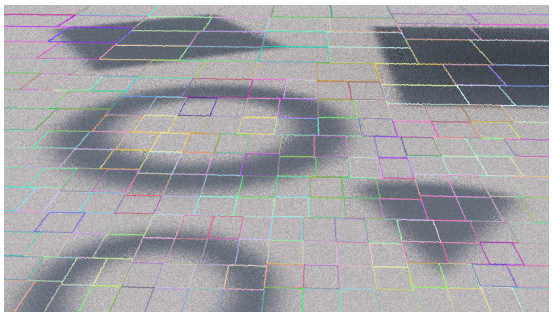


Figure 6.20.: *Baseline*

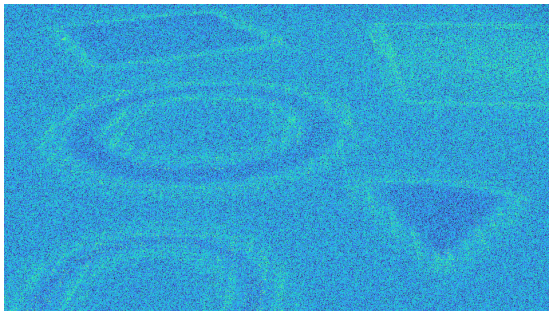


Figure 6.21.: *RAE of CE-0 (0.141)*

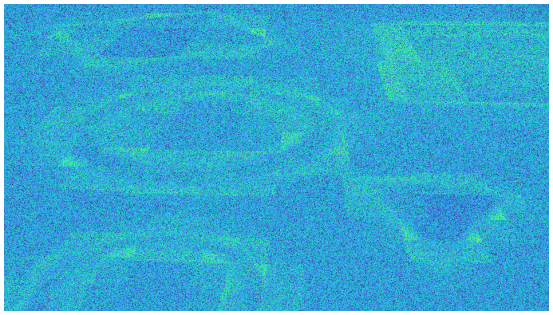


Figure 6.22.: *RAE of Baseline (0.155)*

Figure 6.23.: *Baseline vs. cross-entropy-based subdivision in Complex Shadows. "CE-0.01+128k" means the cross-entropy threshold 0.01 with a maximum sample count threshold of 128k. The last row compares the relative absolute error (RAE) between "CE-0" and the baseline. The warmer the color, the higher the error. Numbers in the brackets indicate the mean error.*

Local Optima. Examining fig. 6.15, we can see the process gets stuck in the middle regions that contain the shadow boundary. Ideally, if the candidate splits are parallel to the shadow

edge, they would enhance the guiding field, and thus be adopted. However, proposed splits are orthogonal to the shadow edge, resulting in lookahead regions no better than the parent. This suboptimality from the baseline strategy prevents our adaptive solution from seeing the advantages of future splits there.

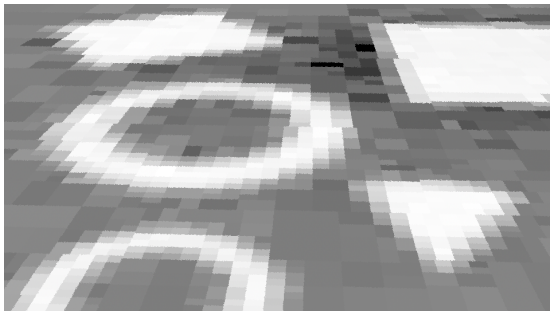
This phenomenon inspires our proposed fix of using the maximum sample count threshold, as introduced in section 5.1.4.

In *Complex Shadows* (fig. 6.23), the cross-entropy-based subdivision with a zero threshold (fig. 6.17) successfully adapts to the change of lighting. Shadowed areas have a higher cache resolution than the bright parts, contrary to the baseline that subdivides uniformly (fig. 6.20). Still, we see some unnecessary splits coming from noise. Increasing the CE threshold to even 0.01 will get us stuck in a local minimum at the beginning (fig. 6.18). In fig. 6.19, a defensive sample count threshold can encourage splits in the beginning to escape local minima, resulting in a subdivision with fewer over-refinements than fig. 6.17.

From the error map figs. 6.21 and 6.22, we can see that the adaptive solution significantly reduces the error near the shadows due to the higher cache resolution there.

Estimated Fluence

Ground Truth Fluence



Child CE

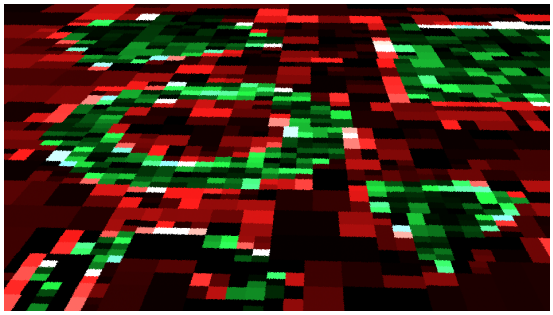


Figure 6.24.: *Parent CE - Child CE*

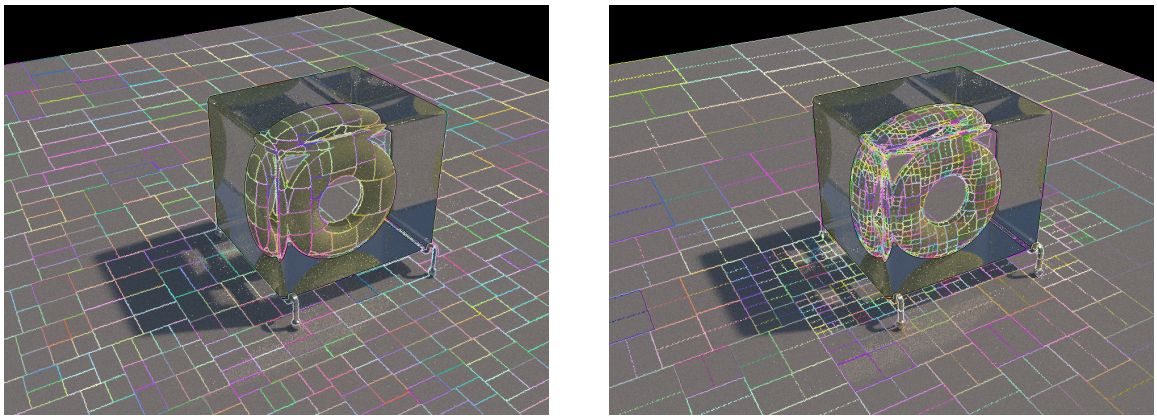
Figure 6.25.: *Visualization of sufficient statistics of fig. 6.17. The fluence and CE maps are normalized. The brighter the color, the higher the fluence and cross-entropy. In the CE reduction map fig. 6.24, green means the child model has a lower CE than the parent's (better), red means the child has a higher CE, while black means they have the same CE (tie).*

Some visualization of the sufficient statistics are shown in fig. 6.25. The ground truth fluence is obtained by dividing the incident radiance by the sampling PDF at the primary intersection point from path tracing. We observe some similarities between the estimated fluence, resulting from averaging the MC samples eq. (5.17), and the ground truth. From the CE reduction map

fig. 6.24, we see some structures corresponding to the shadow shapes: the child CE is indeed lower than the parent's, suggesting a higher tendency to split than the red parts.

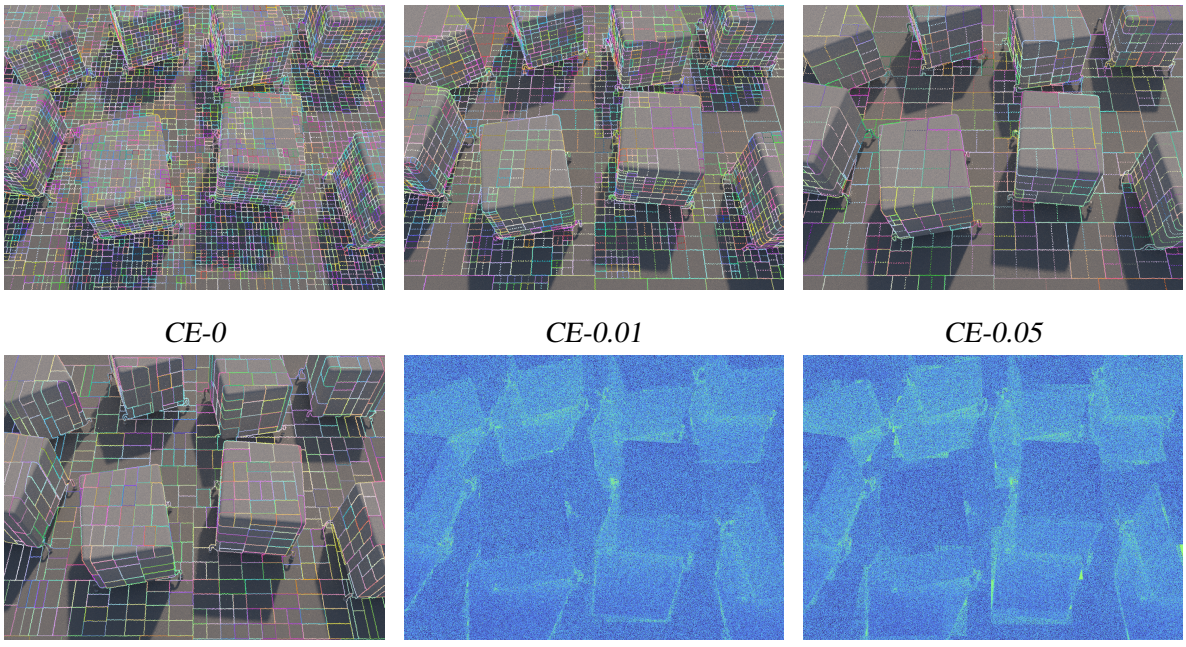
We also compared the CE-based subdivision with the baseline in our small scenes. The defensive maximum sample count threshold is enabled with 128k. Selected results are shown below.

In *Torus* (fig. 6.26), we notice more subdivisions near the caustics, torus, and shadow regions than the baseline, where there is a high-frequency change of lighting.



Baseline *CE-0.01*

Figure 6.26.: *Baseline vs. cross-entropy-based subdivisions in Torus*



CE-0 *CE-0.01* *CE-0.05*

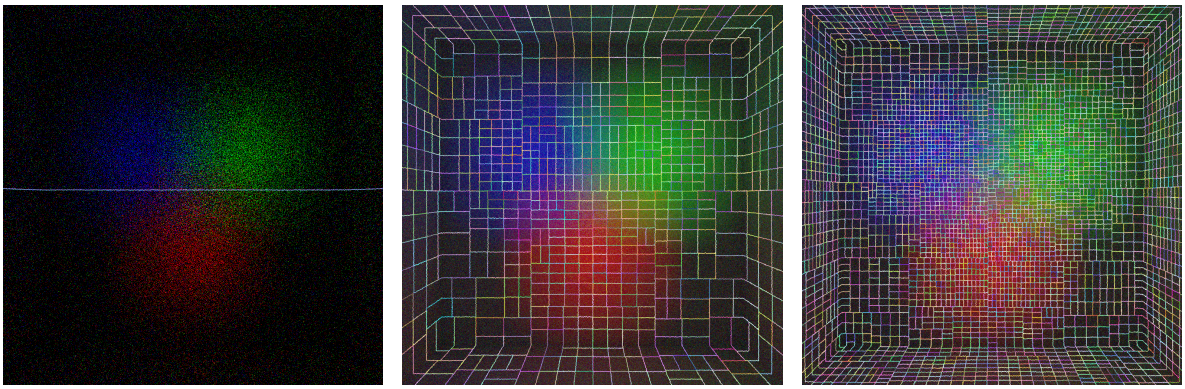
Baseline *RAE of CE-0 (0.102)* *RAE of Baseline (0.109)*

Figure 6.27.: *Baseline vs. cross-entropy-based subdivisions in Multi Cubes*

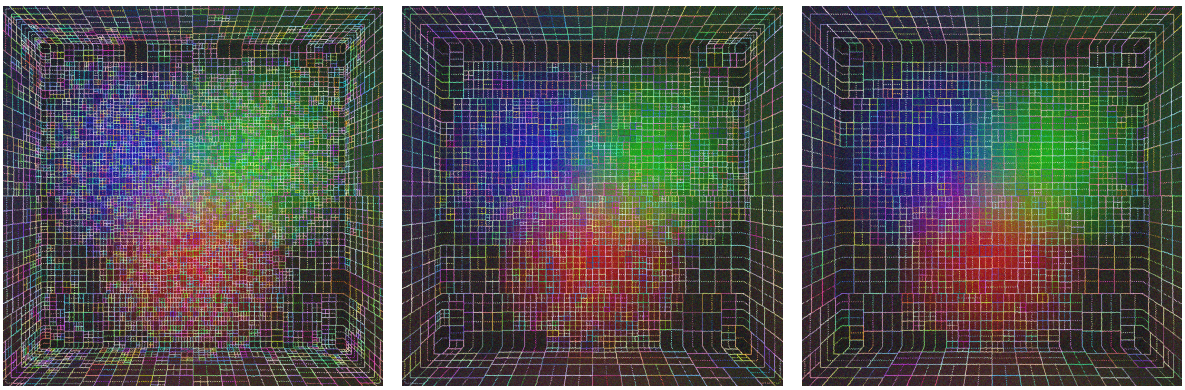
In *Multi Cubes* (fig. 6.27), we sweep the divergence threshold from 0 to 0.05, observing a decreasing number of regions. Regions near the shadow boundaries exhibit a slower decrease than the uniform ones. From the error map, we can CE represents the shadows more finely than

the baseline, which produces higher noise there.

Projection Box (fig. 6.28) is a scene where we stare at the inside of a pinhole camera. Three small area lights from outside project three spots on the camera film (red, green, and blue). The small size of the pinhole makes it challenging to sample the light source in initial iterations. The default baseline with a 32k threshold subdivides too slowly to catch any meaningful signals. Reducing $N_{\max}$ to 8k or 2k can make it work normally, at the risk of getting higher noise in the fitting with fewer samples. The optimal $N_{\max}$ that results in the lowest error in this experiment is 8k. However, tuning this parameter requires the user to be aware of the bias-variance trade-off. All of our solutions, on the other hand, produce adaptive subdivisions with lower errors than any of the baselines, and thus, require less effort in parameter tuning. We attribute the reduction of error to the improvement in selecting the right timing of a split.



Baseline-32k (1.426) *Baseline-8k (0.569)* *Baseline-2k (0.577)*



CE-0 (0.521) *CE-0.05 (**0.516**)* *CE-0.1 (0.519)*

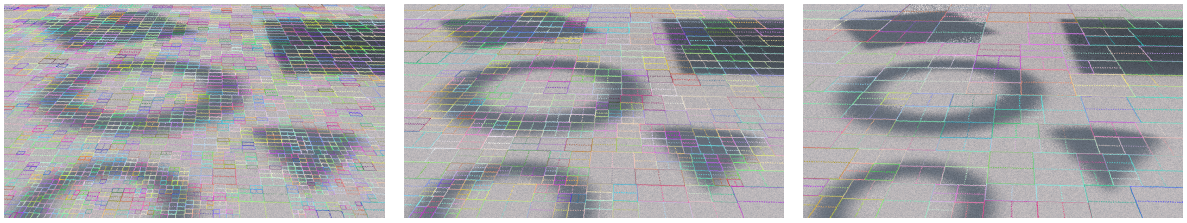
Figure 6.28.: *Baseline vs. cross-entropy-based subdivisions in Projection Box. Numbers in the brackets indicate the mean relative absolute error (MRAE).*

Adaptive Splitting Based on Multiple Importance Sampling

Similarly, we tested our MIS-based split approach (section 5.1.3) with the same rendering configuration. Below are the parameters of the adaptive splitting:

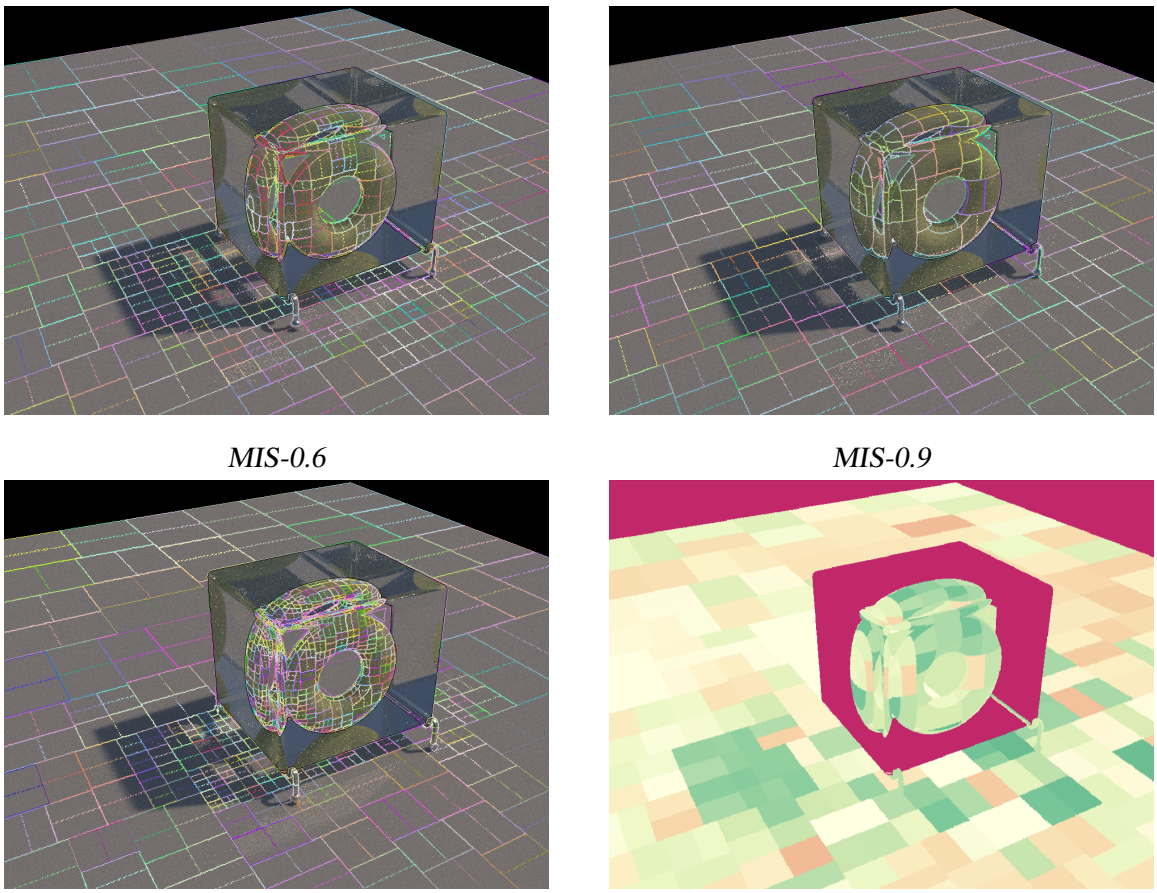
- The maximum sample count threshold $N_{\max}$ is set to 128k.

- The minimum samples per node threshold $N_{\min}$ is set to 1k.
- The MIS weight threshold α_{ths} is set in the range of $[0.5, 0.95]$.
- During queries, we turn off the MIS between the leaf region and lookahead region cache (illustrated in fig. 5.4) by default.



MIS-0.5 *MIS-0.6* *MIS-0.9*

Figure 6.29.: *MIS-based subdivisions in Complex Shadows. "MIS-0.5" indicates splitting when the MIS weight exceeds 0.5.*



MIS-0.6 *MIS-0.9*

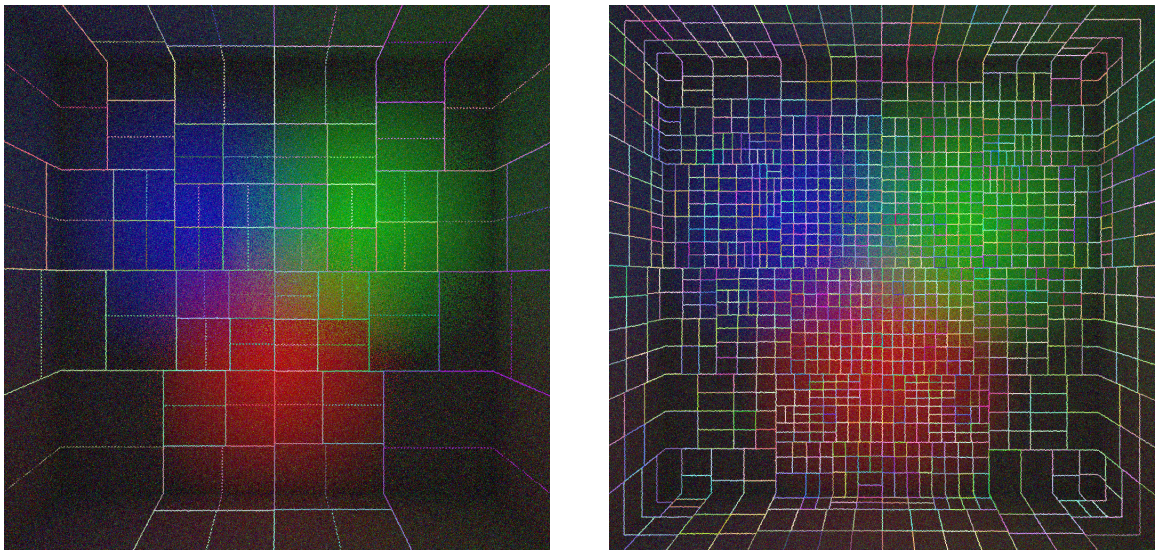
CE-0.01 *Weights of MIS-0.9*

Figure 6.30.: *MIS- vs. CE-based subdivisions in Torus. The lower right image visualizes the α weights of MIS-0.9. Green means $\alpha > 0.5$, red means $\alpha < 0.5$, otherwise white. The red in the far background and the transparent part of the cube result from a missing guiding cache.*

In fig. 6.29, we observe a similar pattern in the change of cache resolution when increasing the MIS weight threshold. The subdivision somewhat adapts to the shadow shapes.

In fig. 6.30, adaptivity to lighting is evident for MIS-0.6. However, when increasing the threshold to 0.9, the MIS-based approach tends to subdivide more uniformly across the scene, compared to CE. The latter is more sensitive to signals and can propose more aggressive splits at high-frequency regions. We speculate this is caused by the spatial marginalization effect and the adapted MIS weight optimization target eq. (5.36). Still, the visualization of MIS weights shows some correlation with the signal frequencies.

MIS for guiding. We experimented with the idea of returning guiding queries with the mixture distribution defined in eq. (5.28) and fig. 5.4. A notable result is shown in fig. 6.31. Despite having the same split threshold, returning the mixture distribution (MIS-Mix-0.9) can encourage exploring the directions suggested by the lookahead child regions. In this scene, the child model happens to be more accurate about lighting than the parent one, hence reinforcing its MIS weight, leading to more subdivisions and lower errors. However, we find this effect insignificant in other scenes.



MIS-0.9 (0.674) *MIS-Mix-0.9 (0.613)*

Figure 6.31.: *Guiding without (MIS-0.9) vs. with (MIS-Mix-0.9) the mixture distribution in Projection Box. Numbers in the brackets indicate the MRAE.*

Quantitative Study

To provide insights into our proposed methods' behavior quantitatively, we use two metrics: the number of regions and the rendering error throughout rendering iterations. We run the rendering with baseline and CE subdivision in our small or large scenes for 64 SPP in the training pass and 64 SPP in the evaluation and compile the metrics into plots. The defensive maximum sample count threshold for CE is set to 512k and only enabled for the first 12 depths of the tree. Selected results are shown in figs. 6.32 and 6.33.

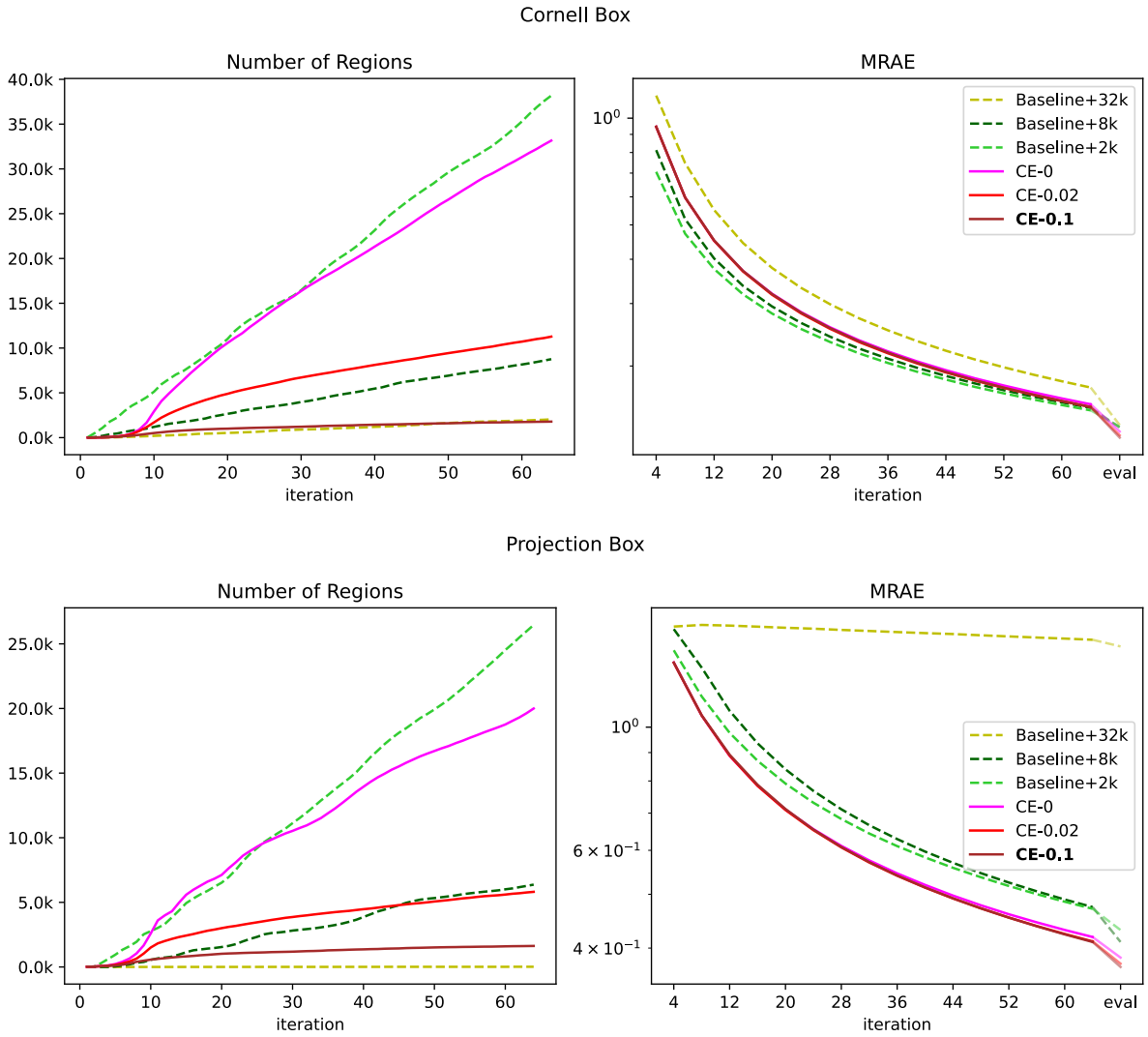


Figure 6.32.: *Number of regions and MRAE over rendering iterations: baseline vs. cross-entropy-based subdivisions with different thresholds (a). E.g. "Baseline-32k" means the baseline with a 32k maximum sample threshold. Bold labels highlight the method with the lowest evaluation error.*

Overall, we have the following findings:

1. Both the baseline and the adaptive methods produce more partitions as the split threshold decreases.
2. Baseline's performance regarding the rendering error is sensitive to the choice of the maximum sample count threshold, while the adaptive method generally performs similarly as long as the divergence threshold is below 0.1.
3. Baseline's number of regions grows linearly over rendering time.
4. The adaptive method exhibits a "burn-in" period: during the first 10 iterations, its number of regions grows sub-linearly, slower than the baseline. We attribute this to be related to the local optima as described before, especially when the region is large.

5. After the burn-in period of the adaptive method, the region count then grows linearly. The growth slows down and even displays a trend of convergence with a large split threshold such as CE-0.02 and CE-0.1 in *Cornell Box* and *Projection Box*.

6. In best cases, the adaptive method requires a much lower region budget while maintaining similar or even better rendering performances. For instance, in *Cornell Box* (and *Living Room*), the best adaptive method CE-0.1 requires fewer than 1k (10k) regions, whereas the best baseline-2k produces around 40k (50k) regions when the rendering ends.

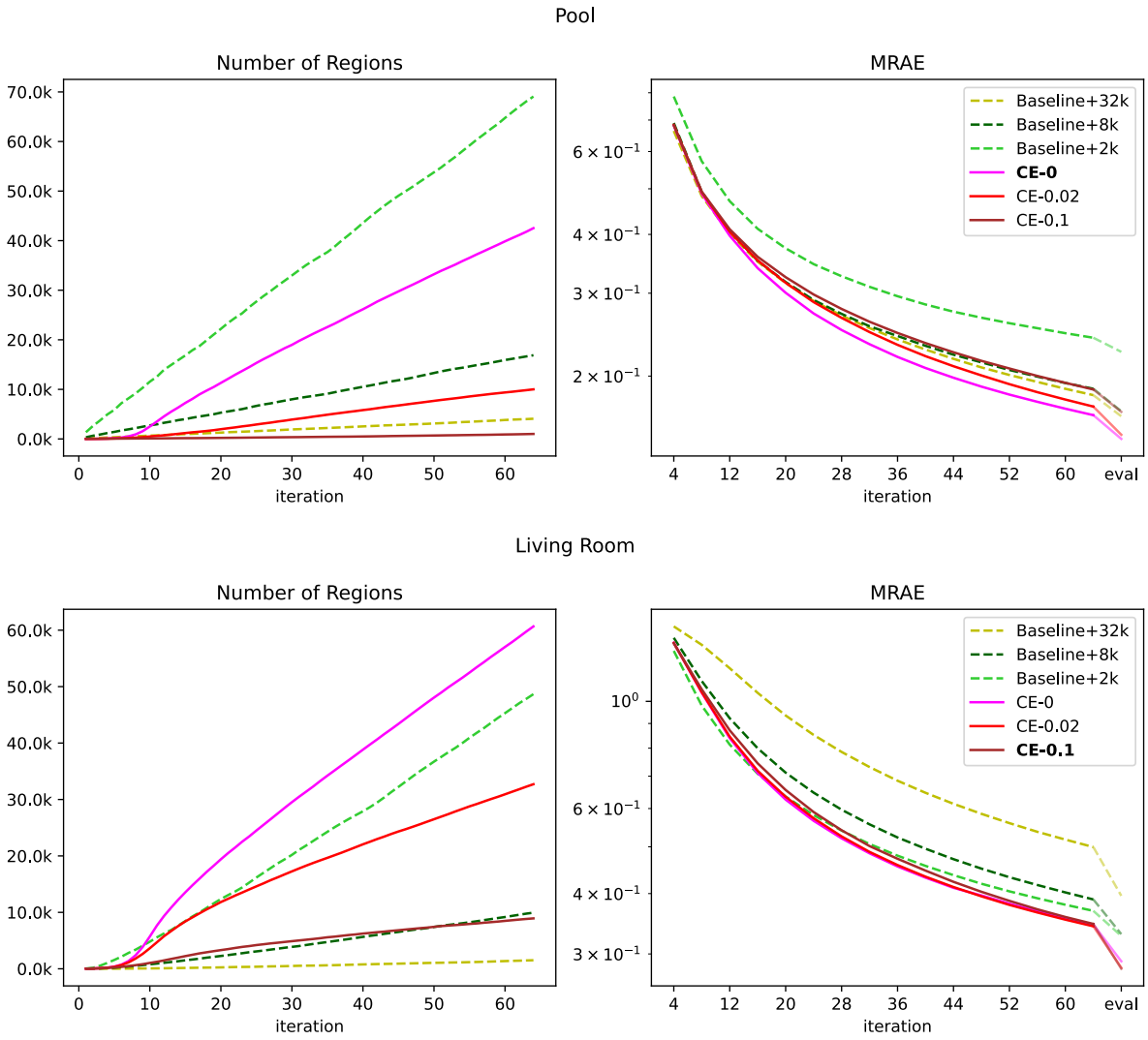
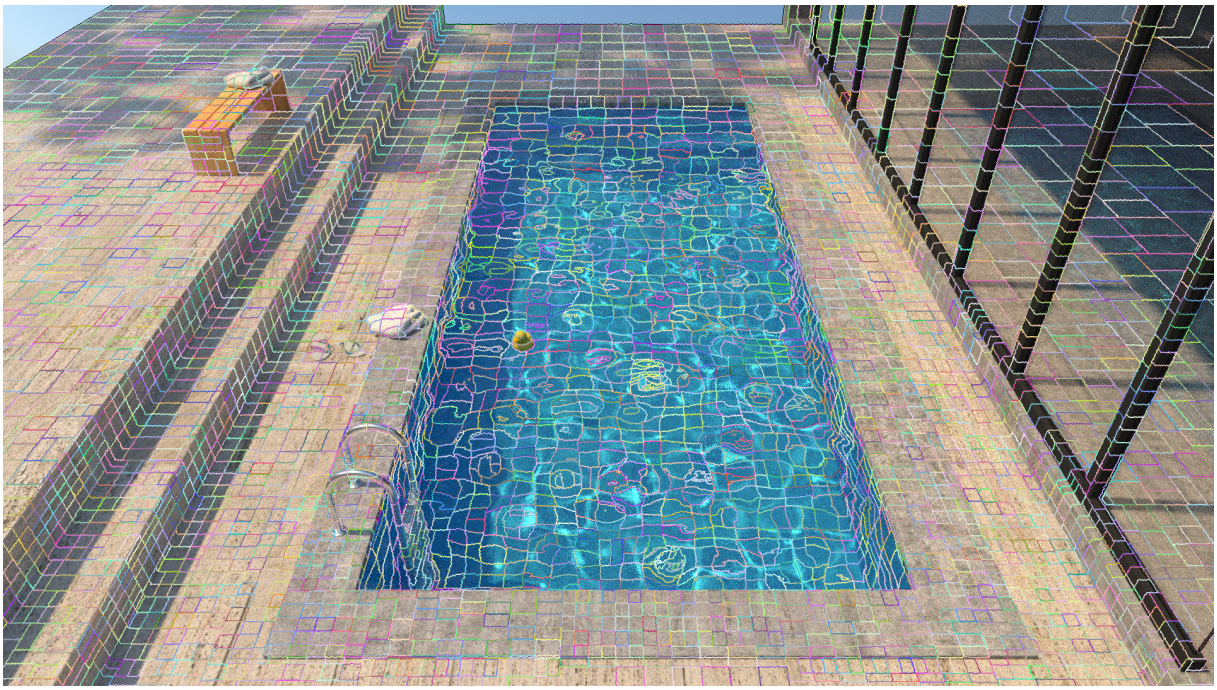
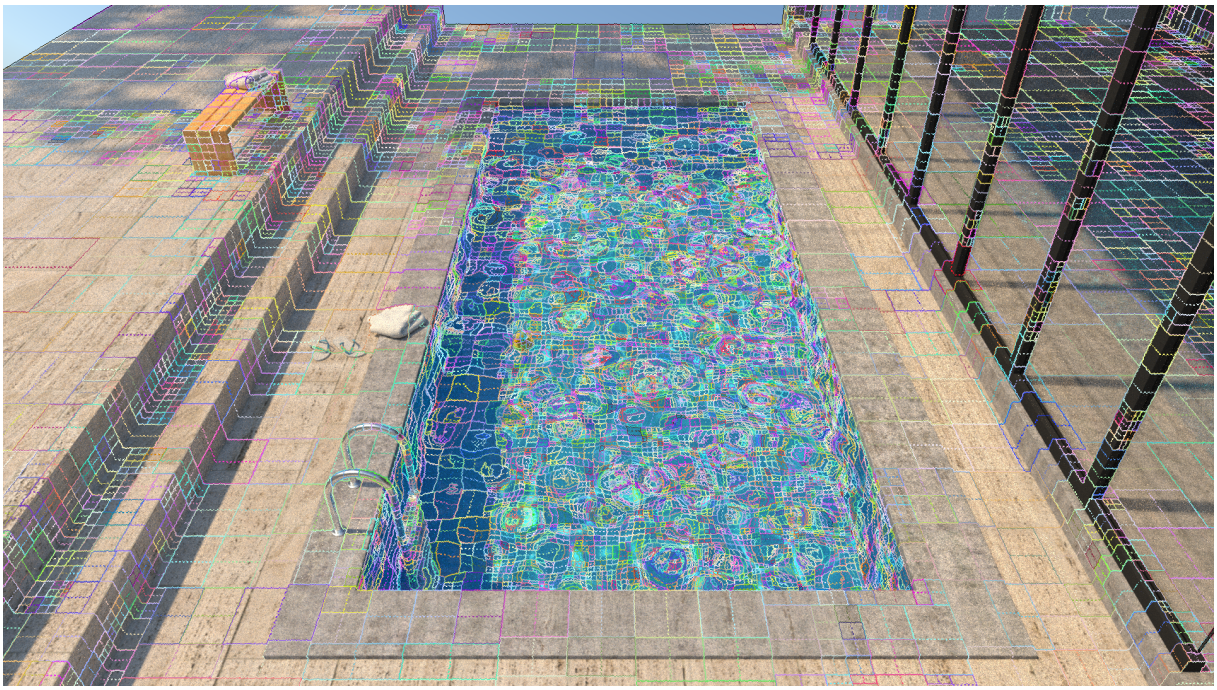


Figure 6.33.: *Number of regions and MRAE over rendering iterations: baseline vs. cross-entropy-based subdivisions with different thresholds (b).*

A representative subdivision result is demonstrated in fig. 6.34. We select the second-best CE with a threshold of 0.02 for a clean visualization. The baseline subdivides uniformly. CE-0.02 subdivides adaptively to the frequency of the lighting in the scene. Coarse guiding caches are used for directly lit areas at the pool bank. Fine caches are applied to the shadowed areas on the bank, the caustics inside the pool, and the caustics within the house on the right.



Baseline-32k (0.164)



CE-0.02 (0.150)

Figure 6.34.: *Spatial subdivision of the baseline vs. CE-0.02 (64 SPP, Pool).*

6.2.2. Where to Split

This experiment section provides a qualitative study for our proposed scanning algorithms in sections 5.2.1 and 5.2.2. We compare our solutions with the Open PGL's and PPG's "where to split" strategy eqs. (5.1), (5.2), (6.1) and (6.2). To recap, PPG's strategy is given as by:

$$\text{Split at dimension} \quad d = \text{RoundRobin}(d_p) \tag{6.1}$$

$$\text{and pivot} \quad t = \left(\frac{\mathbf{x}_{\min} + \mathbf{x}_{\max}}{2} \right)_d \tag{6.2}$$

Where $\mathbf{x}_{\min}, \mathbf{x}_{\max}$ are the bounds of the region, d_p is the split dimension of the parent region.

The "when to split" policies of all methods share the same maximum sample count threshold $N_{\max} = 32k$. Training SPP n_t is set to 8 to magnify the effect of early splits which are more crucial for a spatial subdivision. Evaluation SPP $n_e = 32$.

We turn on the defensive fallback as described in section 5.2.5, with the following parameters for each method:

- Variance-scanning (VS): no defensive fallback.
- Covariance-scanning (COVS): $g_{\text{ths}} = 0.2$
- Information-gain-scanning (IGS): $g_{\text{ths}} = 1.0$ bit
- Fluence-scanning (FS): $g_{\text{ths}} = 0.03$

Geometry-Aware Splitting

We first compare the spatial subdivisions of PPG, Open PGL, and our geometry-aware approaches in *Cube* scene (fig. 6.35). Two geometric issues are identified for previous methods:

1. PPG is prone to generate splits that do not adapt well with the geometry edges, evidenced by some thin regions near the cube corners. This is because its split positions are fixed, regardless of the sample distribution.
2. Open PGL creates "sticky" partitions that mix up geometries, especially near the contact points between the cube and the ground.

We observe improvements with our geometry-aware methods, albeit with some failure cases:

1. VS produces more isotropic regions with the help of the distance-based metric but still mixes up geometries.
2. COVS separates the cube and the ground while maintaining a high isotropy on average except for one problematic thin region intersecting the upper left contact point (green).
3. IGS also separates the geometries. It splits more aggressively than COVS in that it identifies all the faces of the cube. Failure cases occur at the upper edges of the cube, which are isolated into separate regions from the faces. We explain this to be the reduction of volume when cutting the "L"-shaped joint into two planar parts, similar to the *Lines* and *L* examples in fig. 6.10.

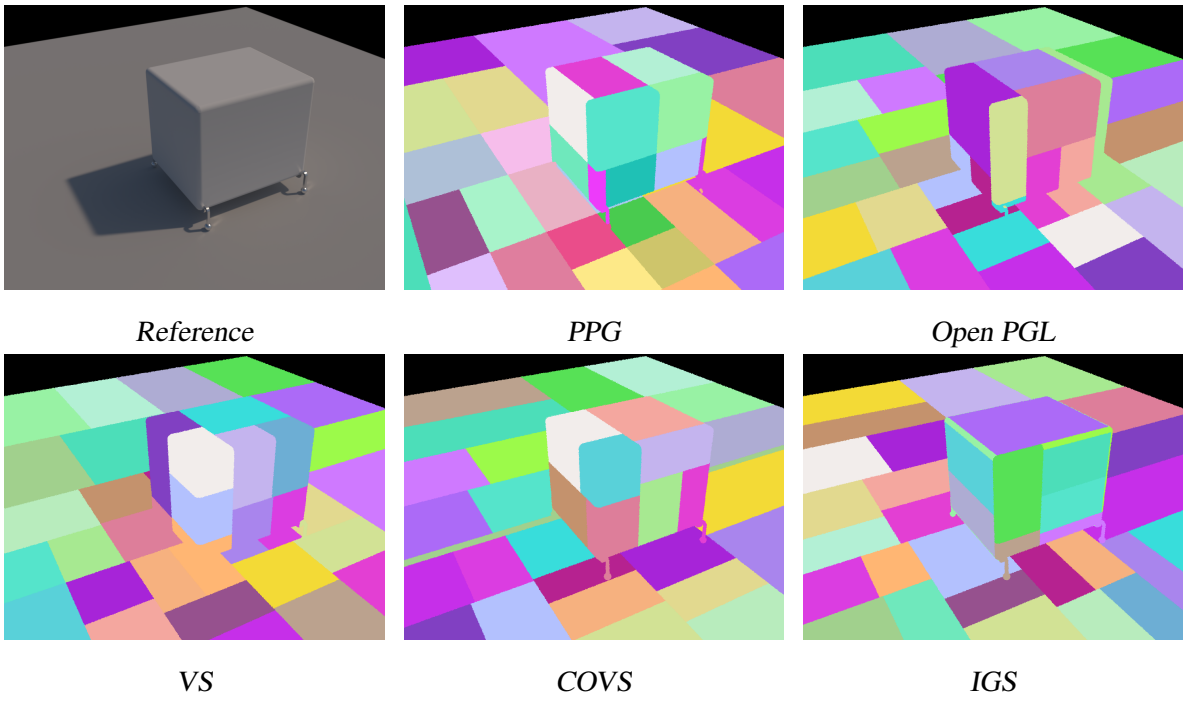


Figure 6.35.: *Comparing the spatial subdivisions of PPG, Open PGL, VS, COVS, and IGS in Cube. We visualize their subdivision as the region IDs at the hit location, during the 2nd iteration of the training pass.*

In *Multi Cubes* scene (fig. 6.36), we see similar issues of thin and sticky subdivisions with PPG and the Open PGL. VS has the highest isotropy visually. Both COVS and IGS create "sharp" partitioning that separates geometries.

In fig. 6.37, the benefit of a geometry-aware splitting is significant. From the region visualization, we can see that IGS immediately detects the walls, the ground, the ceiling, and two cuboids, while PPG and Open PGL attempt to fit all the geometries into several large bounding boxes. At the end of the training, IGS produces a subdivision where all faces of cuboids are sharply divided into their respective partitions, whereas the other two still have problems with mixing geometries, which was pointed out in fig. 3.8. The advantages of IGS brought by its early splits enhance the fitting of guiding caches, resulting in the lowest rendering error in this scene.

Our approaches can scale up to larger scenes. *Classroom Lights-on* (fig. 6.38) contains densely arranged desks and chairs, as well as geometrically complex objects such as flagpoles, blackboard frames, and light tubes. From the reference rendering, we know that the lighting distribution on these various surfaces differs drastically. Hence, the geometry assumption should hold. All methods seem to somewhat identify the objects.

However, zooming into the region IDs, we notice Open PGL's subdivision exhibits two problems: regions that glue the ground with the chairs and desks, chop-offs caused by inaccurate split locations (e.g. two legs of the leftmost desk, the upper edges of the two chairs on the right, and the podium). In contrast, COVS and IGS sharply detach the desks from the ground and isolate each desk-chair pair to respective partitions. IGS even identifies fine geometries: the blackboard frames, the podium, and even the light tubes. The enhancement in meeting the

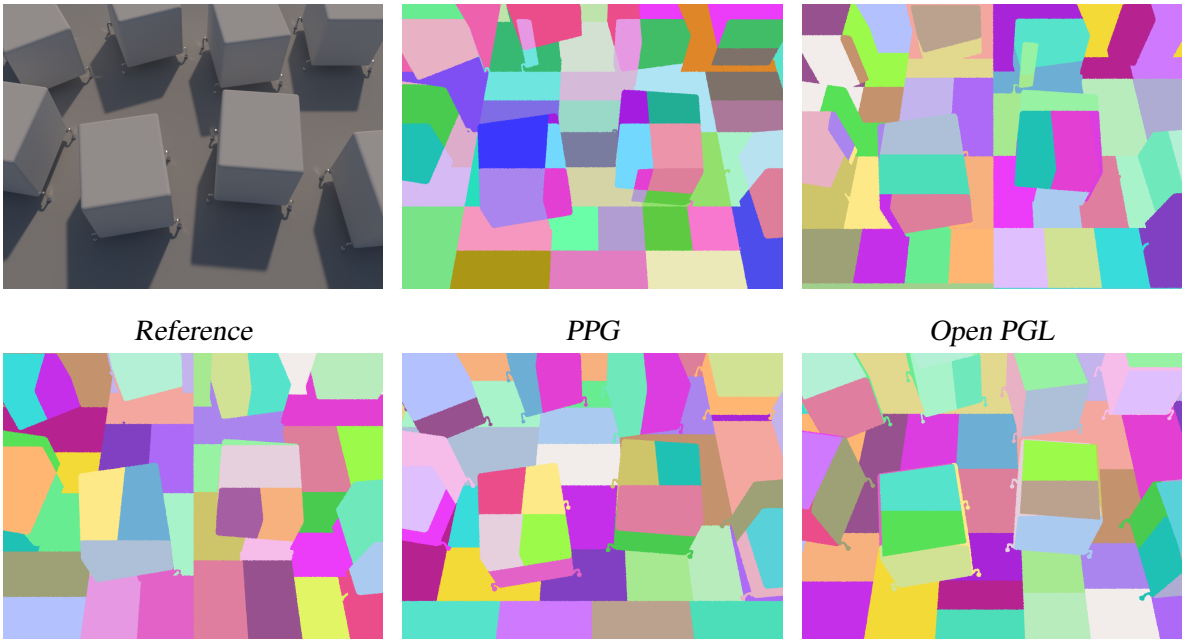


Figure 6.36.: *Comparing 2nd-training-iteration spatial subdivisions of PPG, Open PGL, VS, COVS, and IGS in Multi Cubes.*

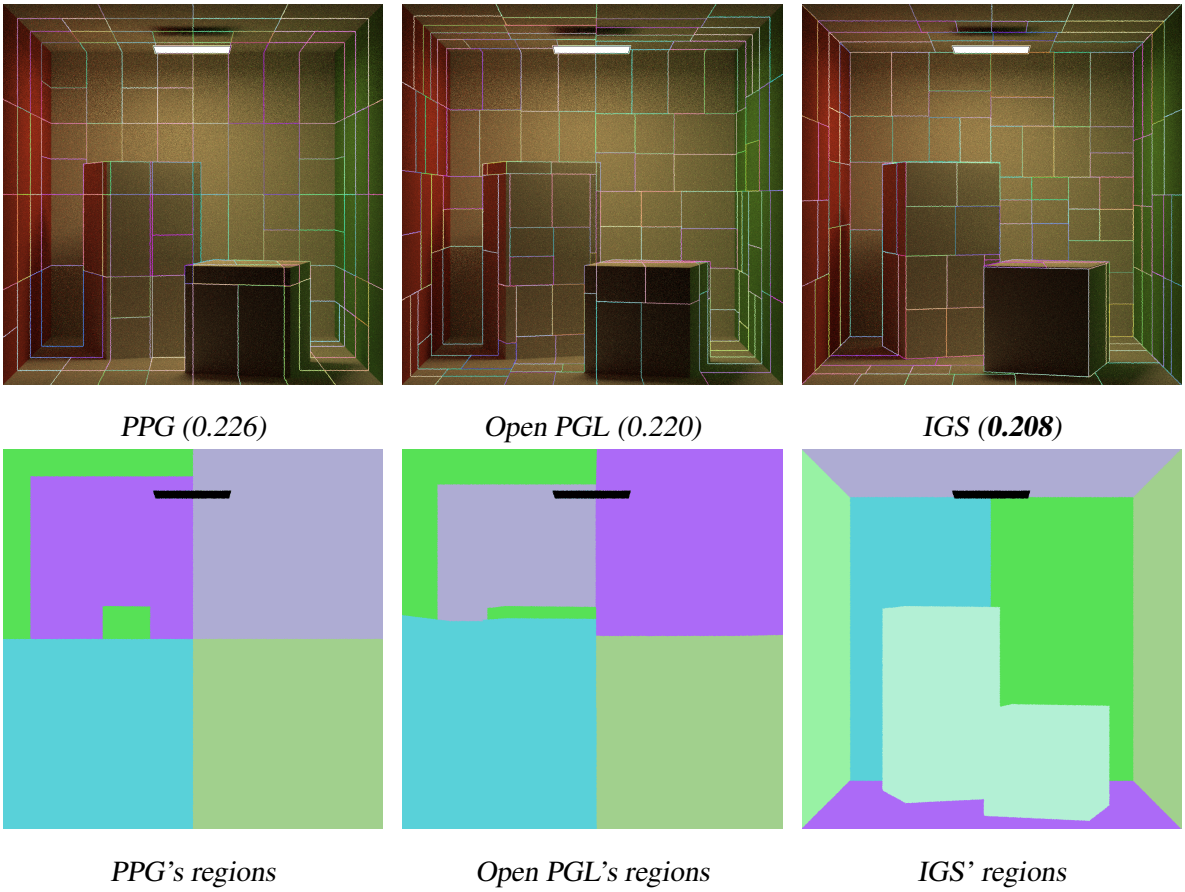
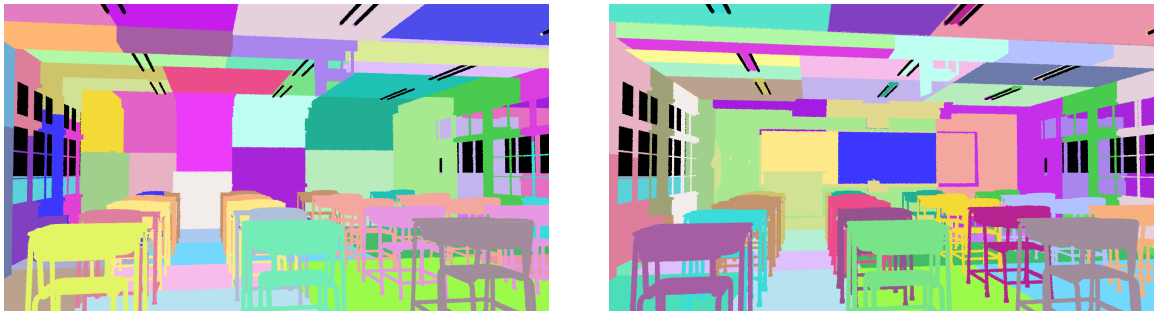


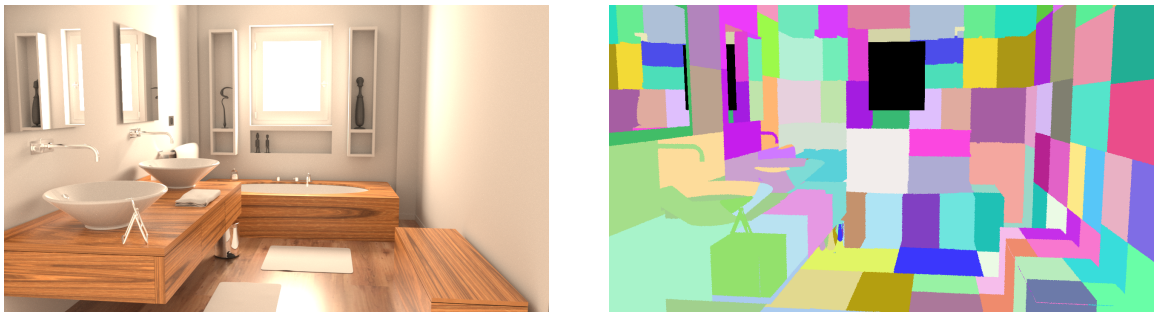
Figure 6.37.: *Comparing PPG, Open PGL, and IGS in Cornell Box. The second row shows the region IDs at the 3rd training iteration. Numbers in the brackets indicate the MRAE.*

Reference | Open PGL (0.733)

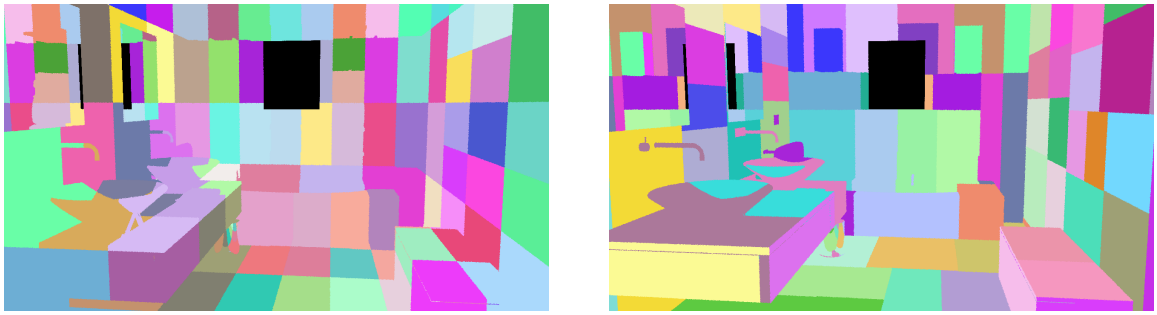


COVS (**0.714**) | IGS (0.726)

Figure 6.38.: *Comparing 4th-training-iteration spatial subdivisions of Open PGL, COVS, and IGS in Classroom Lights-on. Numbers indicate the MRAE.*



Reference | Open PGL (0.537)



PPG (**0.429**) | IGS (0.515)

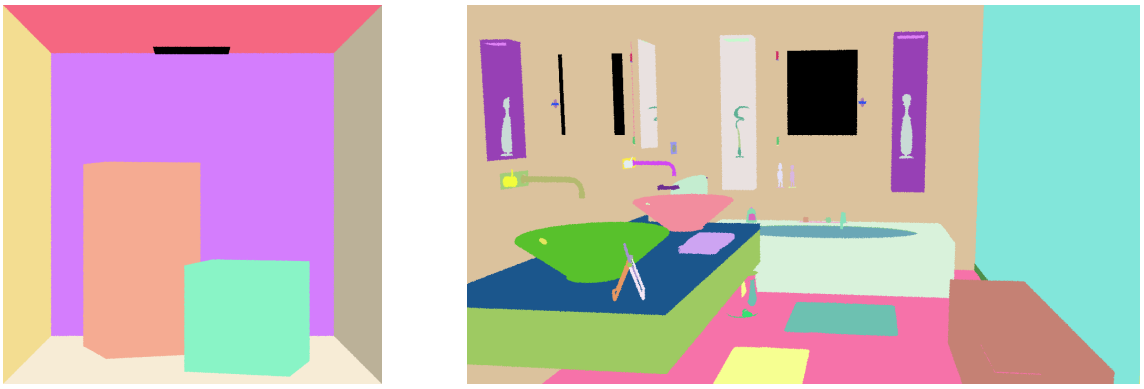
Figure 6.39.: *Comparing spatial subdivisions of Open PGL, PPG, and IGS in Bathroom. Numbers indicate the MRAE.*

geometry assumption makes COVS and IGS stand out in terms of rendering errors.

For the *Bathroom* scene in fig. 6.39, surprisingly, PPG works the best error-wise. Upon an inspection of the region IDs, we can see PPG's middle splits happen to occur at key locations such as the upper face of the wash basin and the contact line between the bath and the floor. Likewise, IGS is also able to detect these objects, albeit with more aggressive and precise cuts at all the edges. We speculate that in this scene, the geometry assumption does not always hold, as points from nearby geometries can still share similar lighting distributions. Thereby, a more conservative algorithm that finds a balance between the geometry assumption and the proximity assumption may perform better in reality.

Geometric performance. The figures shown above provide a qualitative assessment of the goodness of subdivision either based on visual judgments or rendering errors. One may argue that the former is too subjective to serve as an evaluation, while the latter is sensitive to the random seed of the rendering. Stepping back, we have yet to validate whether our models satisfy the fundamental geometry assumption. Namely, they should separate different geometries into different regions.

Therefore, we developed a novel metric to evaluate the performance of achieving the geometry assumption quantitatively, called the *geometric performance*.



Cornell Box *Bathroom*

Figure 6.40.: *Ground truth labels (shape IDs)*

To begin with, we notice the shape IDs that are available in the scene description. Distinct objects are assigned with different IDs. For instance, the shape IDs in *Cornell Box* and *Bathroom* are shown in the right image of fig. 6.40.

We can compare spatial subdivision to a classification task, where the input is a 3D coordinate and the output is a region ID, essentially a decision tree with 3D features and a discrete target. This way, under the geometry assumption, the desired output of the classifier should be exactly the IDs of the objects, which means we can somehow utilize the shape IDs as the ground truth labels to evaluate the classifier. For example, we can calculate the accuracy score between the model output from a group of samples and their ground truths.

However, two problems arise with this idea. First, the ID values of the ground truth do not necessarily match the model output, even if we have a perfect model. For instance, the model

output "1" for the left wall, while its ground truth label is "42". We care about the distinctiveness of the output instead of its actual values. Second, if we solved the first issue, a model can still trivially achieve 100% accuracy by subdividing super finely until one sample gets on one region. That kind of model is undesirable in practice because of too much memory consumption and instability in the cache fitting. A desirable one, instead, should achieve a high score while keeping a low tree height.

Inspired by the metrics from decision trees, we came up with evaluating the geometry performance based on the impurity. Specifically, we feed a set of spatial samples labeled with shape IDs $S = \{(\mathbf{x}_i, c_i)\}_{i=1}^{N}$ to the k-d tree. Then we compute the impurity score of the sample labels at each node (both internal and leaf), either with Shannon entropy (section 2.7.1) or the Gini impurity (section 2.7.3). Finally, we average the impurity scores of all the nodes at a certain depth d weighted by the number of samples, defined as the "impurity at depth d":

$$\mathrm{Imp}(d) \coloneqq \frac{1}{N} \sum_{i \in \mathrm{nodes}^*(d)} N_i \, \mathrm{score}(S_i) \tag{6.3}$$

$$\mathrm{score} \sim \text{(2.20) or (2.25)} \tag{6.4}$$

Where N_i is the number of samples arriving at node i. $\mathrm{nodes}^*(d)$ denote the set of **leaf nodes of the sub-tree** with a maximum depth of d, as illustrated in fig. 6.41. It's defined this way instead of just the nodes at depth d because the union of the latter's regions does not necessarily cover the whole space. The depth d is between 1 and the tree height.

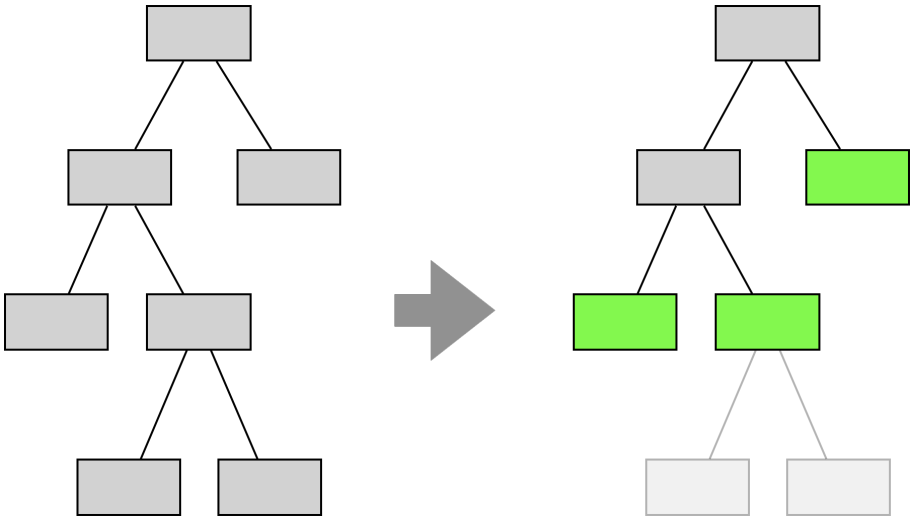


Figure 6.41.: *An example k-d tree (left) and the definition of* $\mathrm{nodes}^*(d)$ *(right). The green ones illustrate* $\mathrm{nodes}^*(3)$*, used for computing* $\mathrm{Imp}(d)$.

In particular, a high $\mathrm{Imp}(d)$ means the tree nodes at this depth mix a lot of objects. While a zero $\mathrm{Imp}(d)$ means all objects are perfectly separated into different nodes.

Once we have computed the $\mathrm{Imp}(d)$ for all p, we obtain an *impurity curve*. It starts from the highest impurity at the root node because all the geometries are initially grouped together. As d increases, the impurity will monotonically decrease, due to the submodularity of the impurity metrics. The rate of the decrease reflects the tree's ability to separate geometries. Therefore, the impurity curve can be used to measure the geometric performance of a spatial subdivision.

Specifically, $\mathrm{Imp}(d)$ measures the geometric performance at a maximum subdivision depth of d. A faster decrease and a lower minimum imply a better performance.

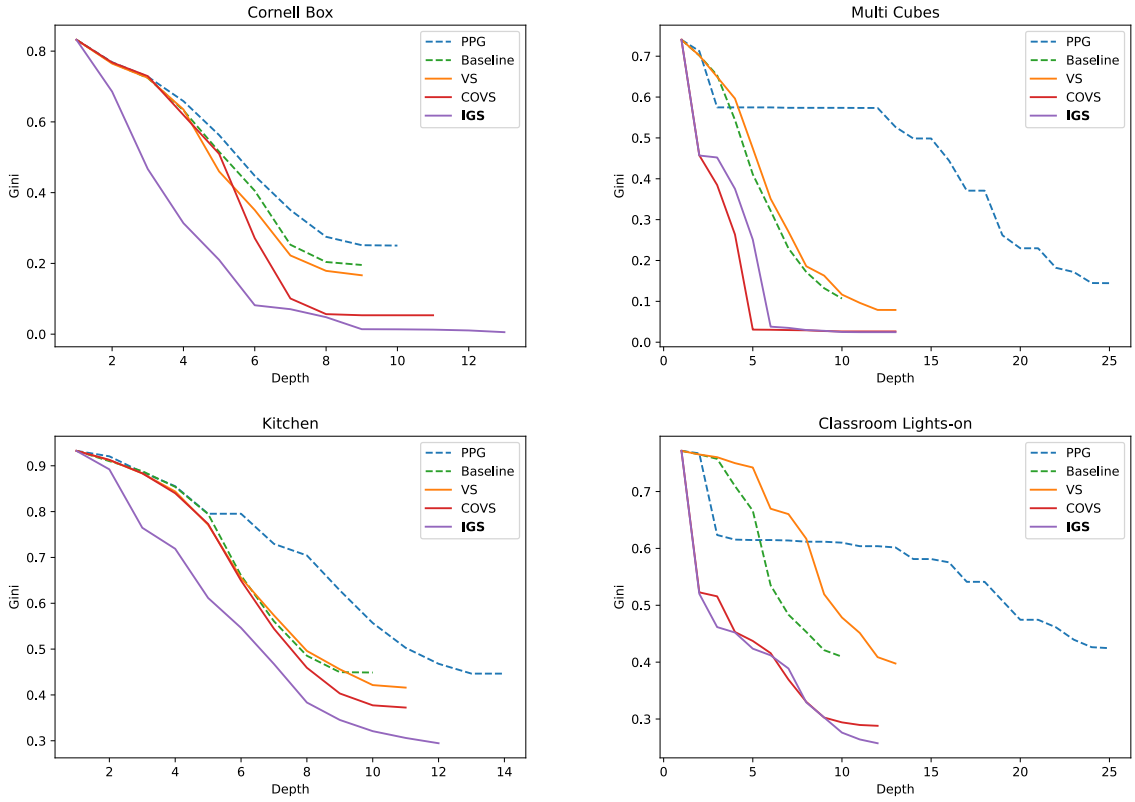


Figure 6.42.: *Gini impurity curves of the spatial subdivision of different methods across various scenes. Bold labels mark the method with the lowest minimum impurity.*

With this metric, we can evaluate the geometric performance of our proposed methods vs. the baseline (Open PGL) and PPG. Shannon entropy and Gini impurity results are similar, so we only report the Gini-impurity-based curves. Selected results are shown in fig. 6.42.

We observe the following:

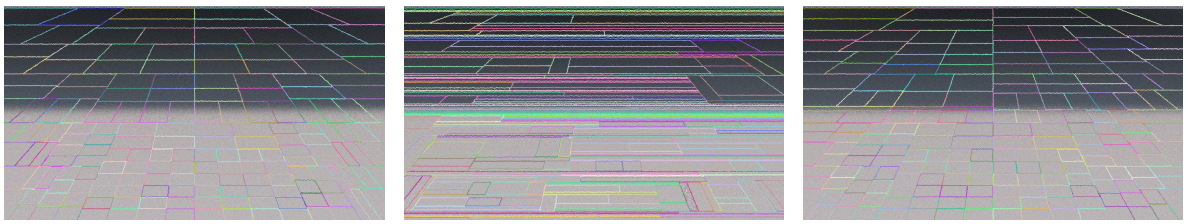
1. All methods reduce the impurity as the k-d tree grows.
2. PPG's impurity curve sometimes gets stuck during part of the subdivision. This happens when all samples are within one side of the middle-point subdivision.
3. VS behaves similarly to the baseline.
4. COVS and IGS reduce the impurity faster than the baseline, PPG, and VS in general. The fastest reduction happens in the initial splits.
5. In all of the results, IGS consistently achieves the most impurity reduction.

We conclude that the proposed covariance-scanning and information-gain-scanning meet the geometric assumption most successfully.

However, it is important to emphasize that the model's geometric performance does not always indicate its guiding cache quality.

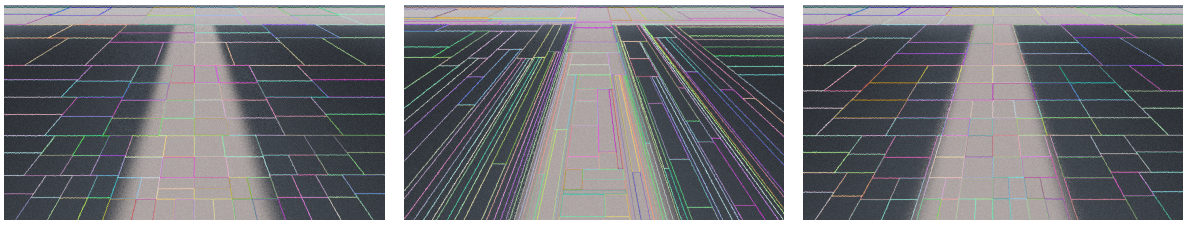
Lighting-Aware Splitting

We evaluated fluence-scanning (FS) in planar and simple scenes. Qualitative results are presented as follows.



Open PGL *FS-0* *FS-0.03*

Figure 6.43.: *Open PGL vs. FS with different defensiveness in Shadow.*

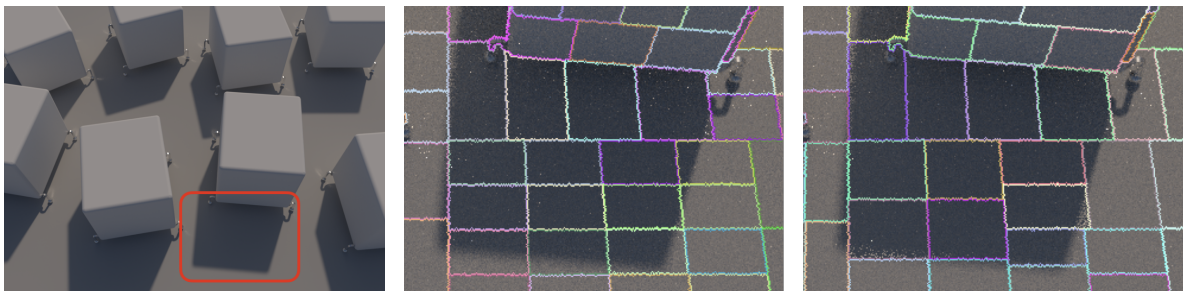


Open PGL *FS-0* *FS-0.03*

Figure 6.44.: *Open PGL vs. FS with different defensiveness in Two Shadows.*

In fig. 6.43, FS-0 splits firmly along the shadow edge because it detects the largest lighting variation along the front-back direction. However, without defensive fallback, this model aggressively creates thin regions at the lower bright regions. With a defensiveness level $g_{\text{ths}} = 0.03$, FS-0.03 finds a better balance between lighting-awareness and isotropy. This supports our handling in section 5.2.5.

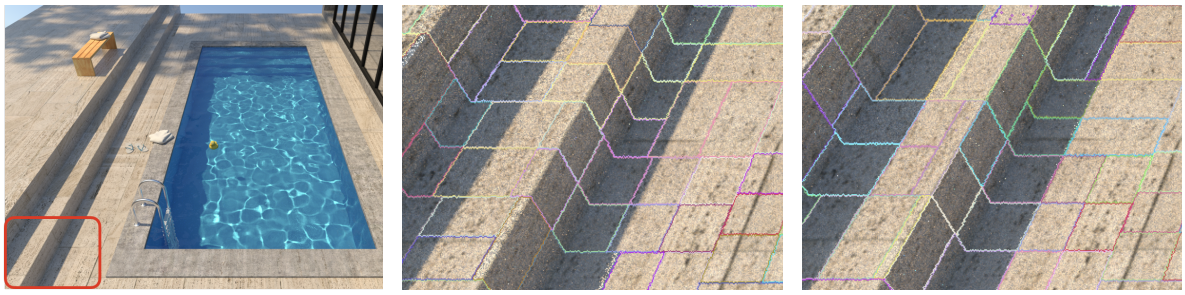
Similar effects are observed in fig. 6.44, where both two FS accurately fit region boundaries to the shape of the shadows.



Reference *Open PGL (0.109)* *FS (0.107)*

Figure 6.45.: *Open PGL vs. FS in Multi Cubes. Numbers indicate the MRAE of the full image.*

We conclude FS' ability to propose split positions while being aware of the shadow shapes in the scene. For robustness concerns, we use a defensiveness of 0.03 in later experiments, denoted by "FS" for brevity.



Reference *Open PGL (0.249)* *FS (0.244)*

Figure 6.46.: *Open PGL vs. FS in Pool*

In figs. 6.45 and 6.46, we notice the improvement of FS in the adaptivity to the shadow boundaries, which brings a slight error reduction.

Quantitative Study

We give a quantitative analysis of the scanning algorithms compared to the baseline (Open PGL) and PPG by plotting the error-iteration curves.

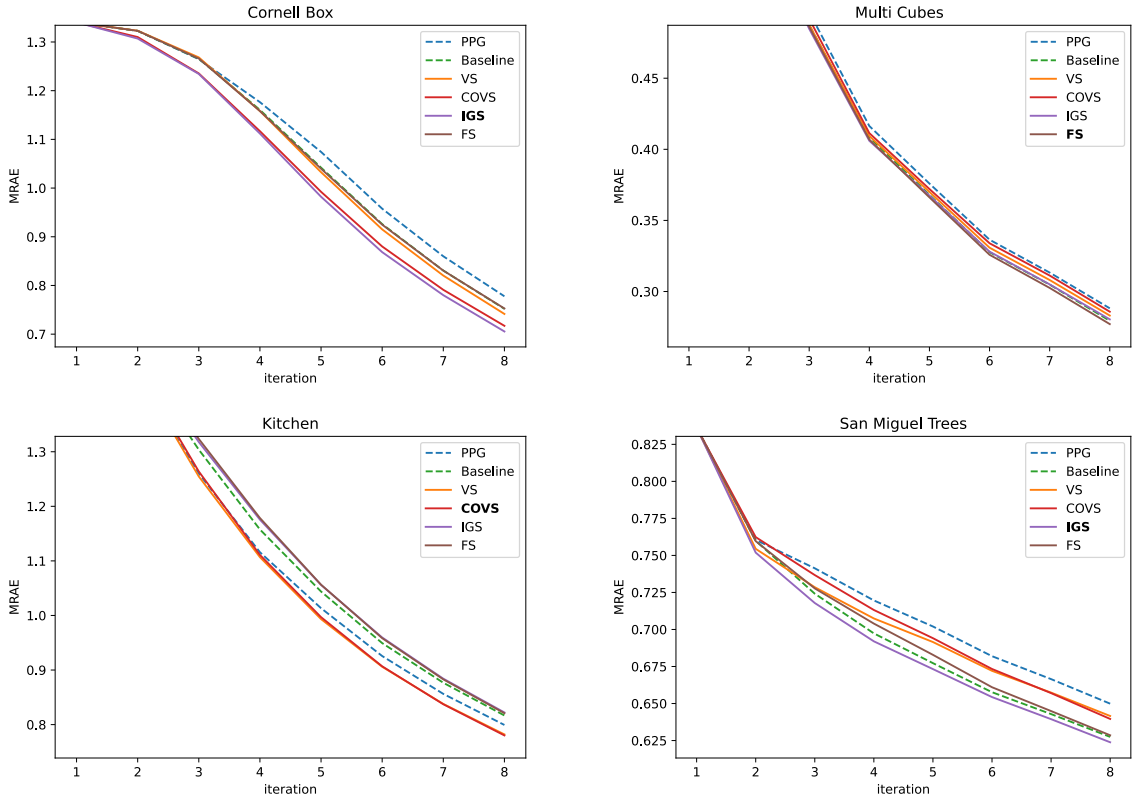


Figure 6.47.: *MRAE over rendering iterations: PPG and baseline vs. our proposed scanning algorithms. Bold labels mark the method with the lowest error in the final iteration.*

From fig. 6.47, we have the following observations:

1. The performance of each method varies from scene to scene. The winner goes to IGS, COVS, and FS based on their suitability for the test scene.
2. Scanning algorithms' performance gain over the baseline is less significant compared to adaptive splitting (figs. 6.32 and 6.33).
3. In best cases such as *Cornell Box* and *Kitchen*, the performance gain of COVS and IGS becomes evident from the 3rd iteration onwards. We attribute this to the improvement of early splits.

In other scenes, however, the performance differences between the scanning algorithms and the baseline are less pronounced.

6.2.3. Full Model

In this section, we ran $n_t = n_e = 64$ SPP comprehensive rendering experiments in all small and large scenes with our full models in section 5.3 vs. reference models across different configurations of the split threshold. The common parameters of our models are set in the table below.

Name	Notation	Value
Defensive maximum sample count threshold	$N_{\max}$	$\begin{cases} 512k & \text{if depth less than } 12 \\ \infty & \text{otherwise} \end{cases}$
Minimum sample count threshold	$N_{\min}$	1000
Decay ratio for the sufficient statistics	λ	0.25
Defensive gain threshold for IGS	g_{ths}	1.0 bit
Defensive gain threshold for FS	g_{ths}	0.03

Table 6.1.: *Common parameters for our full models*

All full models and reference ones are listed in table 6.1, where CE-D_{ths} adopts Open PGL's where-to-split strategy, used as an ablation model to verify the impact of the scanning algorithms.

Full Name	Abbreviation	Parameter Range
Baseline (Open PGL) with maximum sample count threshold $N_{\max}$ (Reference)	Baseline-$N_{\max}$	$\{2k, 8k, 32k\}$
Adaptive splitting with cross-entropy threshold D_{ths} (Reference)	CE-D_{ths}	$\{0, 0.02, 0.1\}$
Adaptive splitting with variance-scanning and cross-entropy threshold D_{ths}	CE-VS-D_{ths}	$\{0, 0.02, 0.1\}$
Adaptive splitting with information-gain-scanning and cross-entropy threshold D_{ths}	CE-IGS-D_{ths}	$\{0, 0.02, 0.1\}$
Adaptive splitting with fluence-scanning and cross-entropy threshold D_{ths}	CE-FS-D_{ths}	$\{0, 0.02, 0.1\}$

Table 6.2.: *All of the models used in the comprehensive experiments*

Quantitative results regarding the rendering error vs. iteration are shown below. For each type of model in table 6.2 we display the most performant one in the plots per scene.

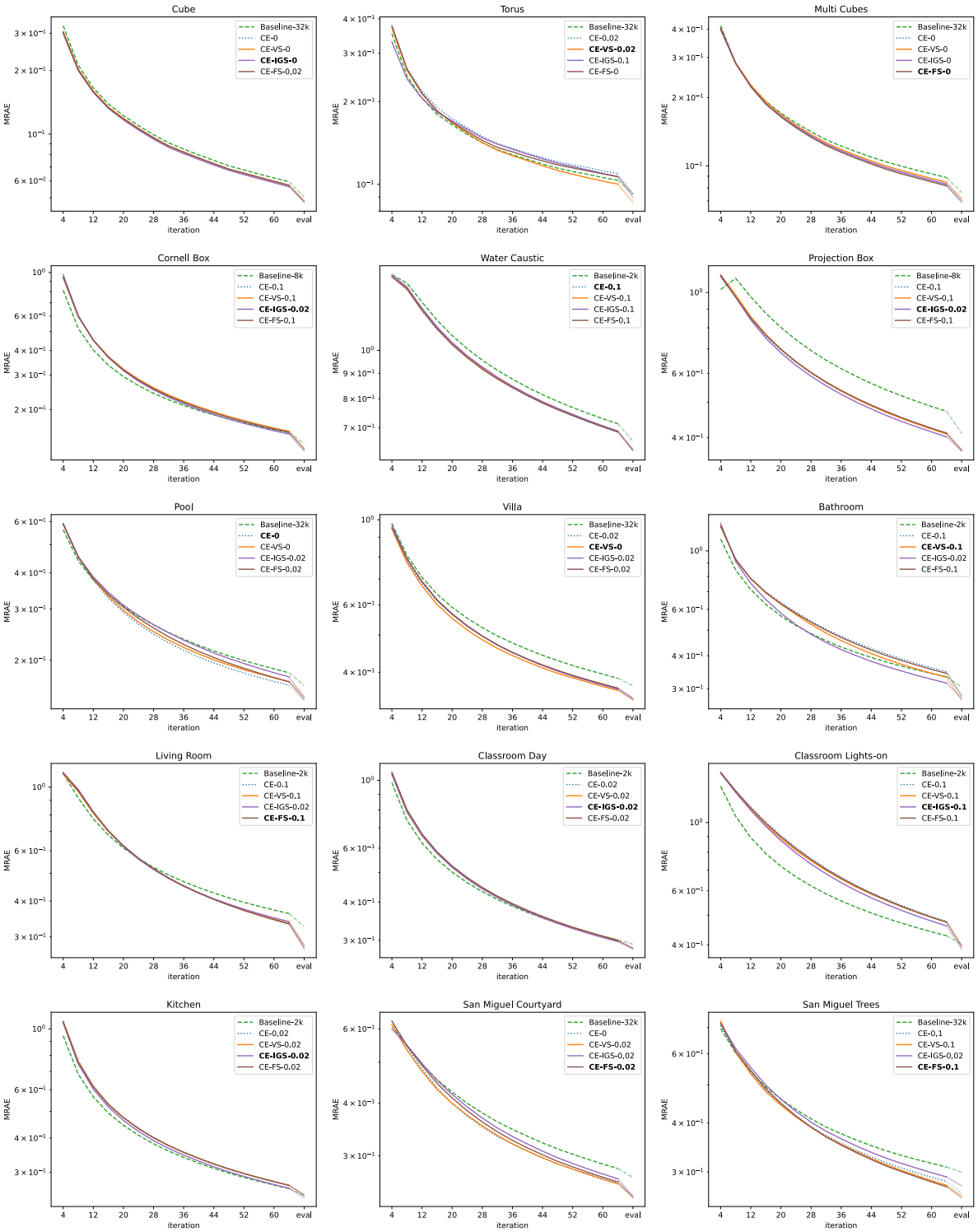


Figure 6.48.: *MRAE over rendering iterations: comprehensive experiments. Bold labels mark the method with the lowest error in the evaluation.*

Finally, for a quick impression of the strengths and weaknesses of each method, we plot their performance gains over the baseline in each scene. Results are collected in fig. 6.49. Specifically, for each scene, we select the best baseline (among Baseline-2k, Baseline-8k, and Baseline-32k) and the best instance of each proposed method in terms of evaluation MRAEs. Then we compute the gain of method m by

$$\text{PerformanceGain}(m) \coloneqq \frac{\text{MRAE}(\text{Baseline}) - \text{MRAE}(m)}{\text{MRAE}(\text{Baseline})} \tag{6.5}$$

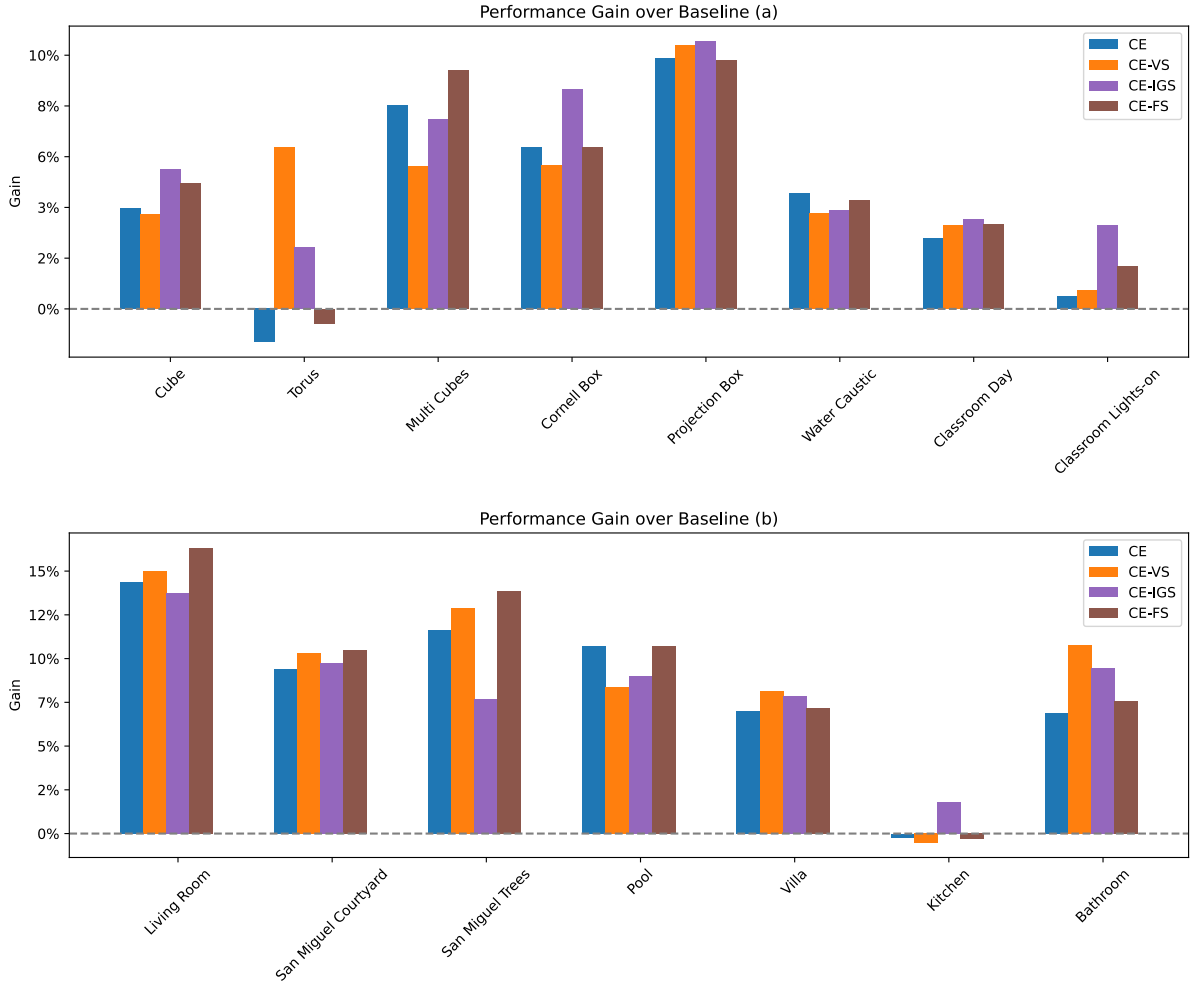


Figure 6.49.: *Performance gains of all proposed models per scene*

From fig. 6.49, we conclude the effectiveness of all our when-to-split and where-to-split models in the spatial subdivision for path guiding. The performance gain from adaptive splitting (CE) is around 3-10%, while the gain further brought by the scanning algorithm to decide "where to split" is only about 0-5%.

On top of the adaptive CE, variance-scanning (in orange) shows an almost consistent improvement. The more sophisticated geometries there are in the scene (e.g. *Classroom Day*, *Classroom Lights-on*, and *Kitchen*), the less significant the improvement.

Information-gain-scanning with CE is the only method that always has positive gain over the baseline. In its most successful cases such as *Cube*, *Cornell Box*, and *Classroom Lights-on*, the scene usually has regular shaped objects (e.g. flat walls and cubes) which are easily detectable by the algorithm. However, a failure case arises for scenes with fine non-regular geometries such as *San Miguel Trees*. A counterexample of our geometry assumption is also found in *Living Room*, where CE-IGS performs slightly worse than CE alone.

The inclusion of fluence-scanning over CE displays minor gains. We find it superior in scenes with non-trivial shadow or shade shapes (*Multi Cubes*, *Living Room*, and *San Miguel Trees*).

Additionally, by inspecting fig. 6.48, we observe the scanning algorithms can sometimes accelerate the error reduction from around 12 SPP onwards (*Bathroom*, *Classroom Lights-on*, and *San Miguel Trees*). We suspect this is due to key progress in the object detection and that the holding of geometry assumption.

As for the divergence threshold for "when to split", we find any of the parameters $\{0, 0.02, 0.1\}$ have their winning scenes. Thus, there is no general rule of thumb on which is the best.

7

Bottom-up Subdivision Approaches

So far, all our presented methods are modifications of k-d trees, which subdivide the scene in a top-down fashion. As we pointed out in section 2.6.4, this way of subdivision has an intrinsic problem: the cutting plane is axis-aligned and will inevitably break the continuity of geometries in scenes with very sophisticated shapes.

In the early stage of this thesis, we sought another direction for spatial structures that achieve the goal of "subdivision" without actually subdividing at all. Having noticed the nature of a spatial subdivision algorithm is to take a query location and output a region ID, we reframe the subdivision into a clustering problem. I.e. given a set of unlabeled point samples, output their labels corresponding to a guide cache ID.

Unfortunately, our attempts to bottom-up approaches turned out less successful. In this chapter, we will briefly present three tentative approaches that follow the concept of clustering. We refer to them as *bottom-up subdivision approaches*, in contrast to the top-down ones.

7.1. Bounding Volume Hierarchy

The "bounding volume hierarchy" (BVH) we used is defined in the narrow sense: bounding volume hierarchies that are constructed bottom-up. We implemented a fast, agglomerative BVH construction algorithm [GHFB13], introduced in section 2.6.4.

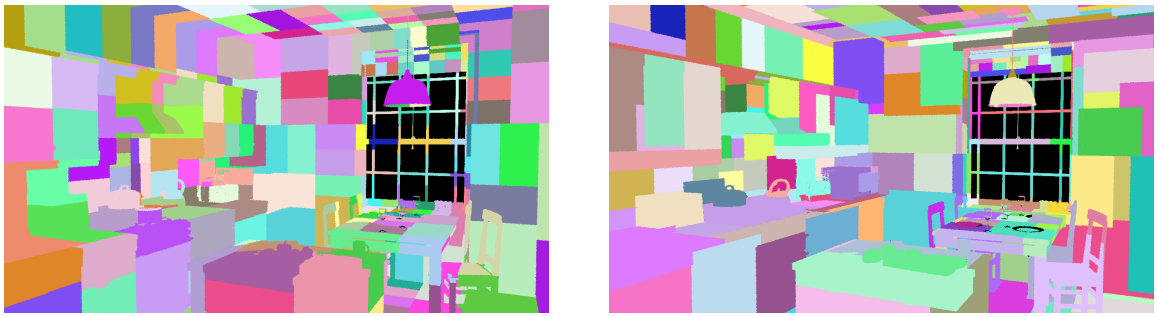
Roughly speaking, we initialize each MC sample as an axis-aligned bounding box (cluster) in the working set. Then we iteratively select the closest pair of clusters out of the set based on some distance metric, merge them into a new cluster whose bounds are the union of the pair, and add it to the working set. The loop stops when there is only one cluster left, the top-level cluster that bounds the whole scene. The selection-merging relationship is memorized between the new cluster and the old pair. Hence a hierarchical structure of clusters is formed.

The most frequently used distance metric in ray tracing applications is the *surface area heuristic* (SAH) that computes the surface area of the merged cluster as the distance between the selected cluster pair.

A crucial problem now is to decide where specifically in the BVH are the guiding regions created. To meet the minimal sample count requirement, a naive way is to track the (monotonically growing) sample count in each cluster as the merging process goes on. Once the sample count in the merged cluster exceeds the minimal sample count N_{min}, create a guiding region with the same shape as the cluster.

Gaps and overlaps. A flaw with the BVH is that the regions it creates do not usually form a valid spatial subdivision. There can be gaps in space where no guiding region is available. Moreover, regions could overlap with each other, making guiding queries ambiguous. Overlaps also compromise the training efficiency of the guiding cache, as samples within the overlaps are repeatedly used for EM fitting.

To solve the first issue, when a query miss happens, we return the closest region to that query location. For the second issue, we return one of the regions intersecting that query point either randomly or deterministically based on e.g. the region size.



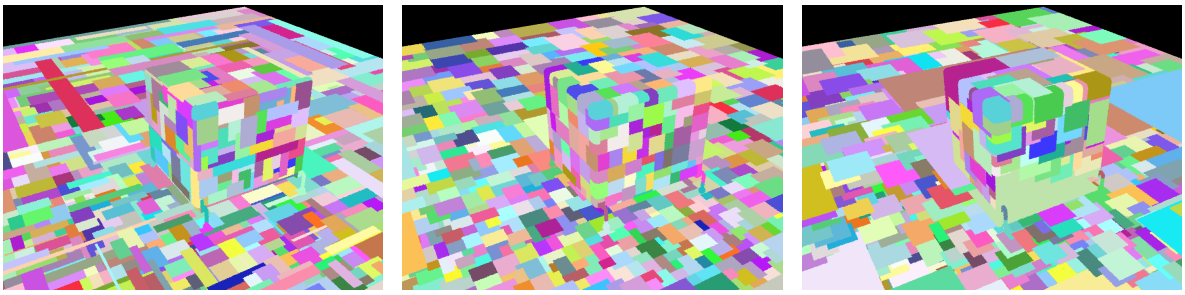
K-d Tree *BVH*

Figure 7.1.: *K-d tree vs. BVH spatial subdivision in Kitchen with* $N_{\text{min}} = 8k$

An example of the spatial subdivision created by the BVH using SAH compared to a k-d tree is shown in fig. 7.1. Visually, the BVH subdivision maintains the shape of the objects somewhat better than the k-d tree. However, severe overlapping issues are identified near the densely arranged regions. Voids without regions are found on a plate on the right table. We also observe more variation in the region size and a higher anisotropy compared to the k-d tree result.

Alternative merge handlings. We also experimented with different merge handlings in a BVH construction (fig. 7.2) to address the overlapping issue.

We found that the default SAH distance metric is prone to generate thin and elongated regions. We tried replacing SAH with the *centroid distance heuristic* (CDH) that calculates the Euclidean distance between the centroids of the cluster pair. It turned out that CDH yields much more isotropic and uniformly sized regions. The amount of overlaps decreases as well.



SAH *CDH* *Cubic Merge*

Figure 7.2.: *BVH subdivision with different merge handlings (Cube)*

In addition, we proposed "cubic merge", which enlarges the clusters to perfect cubes after the merge. We found that this handling can ensure high isotropy but does not resolve the overlaps.

Conclusion. Despite the advantage of a fast linear construction time, the freedom in the choice of merge handlings, and the potential to better fit the shape of scene objects, we still abandoned the BVH subdivision due to the following factors:

1. The overlapping issue that creates ambiguity for the queries and hurts the performance of training.
2. Higher anisotropy of the regions, making it less robust than the k-d tree subdivision.
3. Higher variation of the region size, causing more fluctuation in the sample count per region.
4. Unpredictability: a slight change in the input samples can result in a drastically different subdivision.
5. Non-trivial incremental update: the agglomerative construction described in section 2.6.4 cannot be trivially extended to support incremental refinements iteratively.

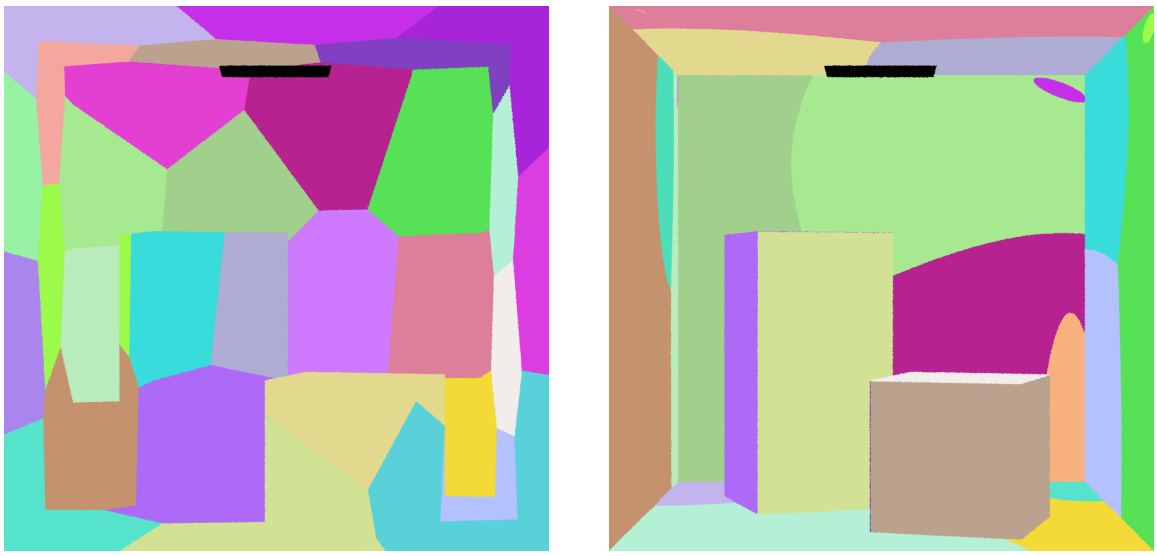
7.2. Clustering

We explored applying two commonly used clustering methods to the spatial subdivision: k-means and Gaussian mixture models (GMMs).

K-means is an iterative algorithm that partitions the space into k clusters. It first assigns each input sample to a random cluster ID from 1 to k. Then it repeats the following steps until convergence or reaching the maximum number of iterations:

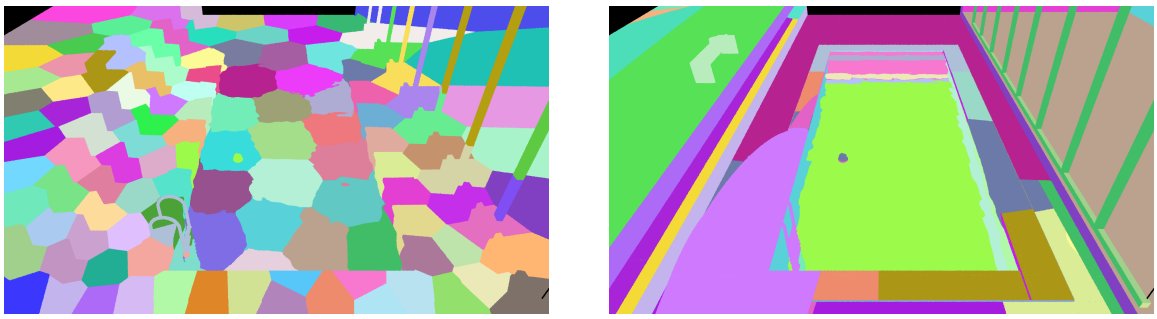
1. Compute the centroid of each cluster by taking the average of samples assigned to it.
2. Re-assign samples to their nearest cluster centroid.

Once the process is done, we can define k guiding regions as the Voronoi sets[1] of the cluster centroids. The parameter k can determine the granularity of the subdivision that is inversely proportional to the maximum sample count threshold.



K-means *GMM*

Figure 7.3.: *K-means vs. GMM subdivision with 32 regions (Cornell Box)*



K-means *GMM*

Figure 7.4.: *K-means vs. GMM subdivision with 128 regions (Pool)*

Figure 7.3 and 7.4 (left) display the results of k-means subdivisions in two scenes. We can see that the regions exhibit Voronoi diagrams, which favor high isotropy and close-to-uniform sizes. However, the downside is that the k-means fitting process is unaware of the radiance field, causing samples with distinct lighting profiles to be grouped into one cluster (e.g. the edge above the water and the wall inside the water in *Pool*). Moreover, the Euclidean distance metric used in the second step of the fitting does not deal with the geometry assumption, resulting in regions with mixed geometries similarly to the Open PGL k-d tree (e.g. the lower left and the lower right regions in *Cornell Box*).

Gaussian mixture models are another technique to cluster unsupervised samples. The GMM assumes samples are generated from a mixture of Gaussian distributions. We fit GMMs

[1] A Voronoi set of cluster i is the subspace whose distance to the centroid i is closer to any other centroids.

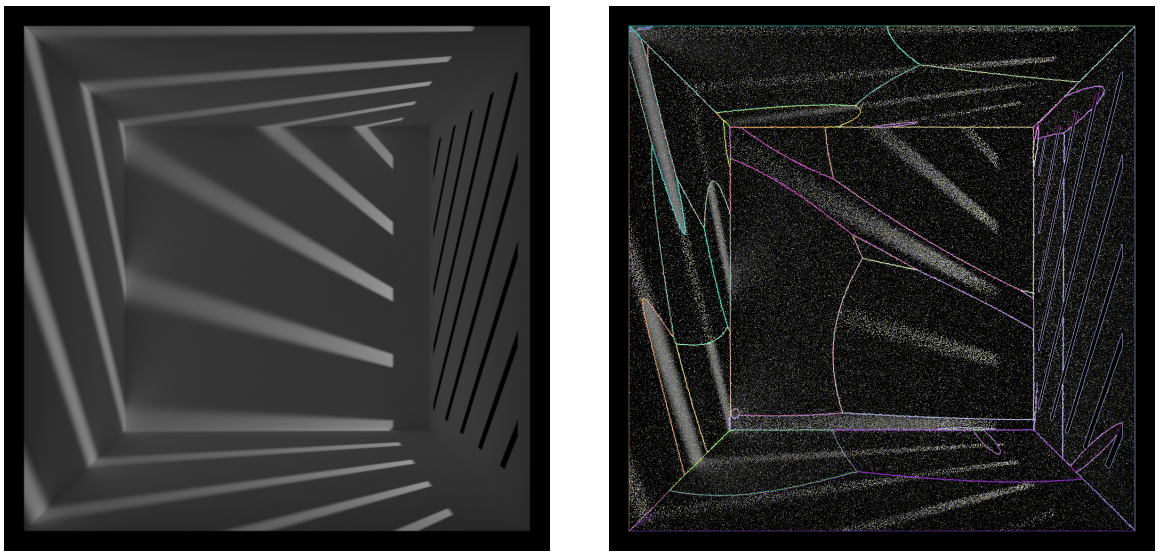
to samples using the expectation-maximization (EM) algorithm that works in an iterative manner, similar to k-means. EM first initializes k 3D Gaussian distributions (i.e. k clusters) with random means and covariance matrices. Then it alternates between the *expectation* and *maximization* steps. Broadly speaking, the procedure of each iteration can be described as:

1. E-step: compute the probability of each sample being assigned to each cluster.
2. M-step: adjust the parameters of the mixture weights and Gaussian distributions to maximize the probabilities from the E-step.

When this process converges or reaches the maximum iteration count, we obtain a GMM that maps a 3D coordinate $\mathbf{x}$ into an array of PDFs $p_i(\mathbf{x})$ for each component and their corresponding weights w_i ($i \in \{1, \ldots, k\}$). We define a guiding region for each of the components as the subspace with a higher likelihood than any other components. Specifically, the region for the i-th component is given by

$$X_i \coloneqq \{\mathbf{x} \mid w_i p_i(\mathbf{x}) \geqslant w_j p_j(\mathbf{x}) \; \forall j \in \{1, \ldots, k\}\} \tag{7.1}$$

Two spatial subdivision results of GMMs are demonstrated in figs. 7.3 and 7.4 (right). Interestingly, GMMs show great potential in fitting the shapes and surfaces of the scene objects. In a complex setting (fig. 7.5) where very few non-zero-weight samples are projected into the box through narrow slits, GMMs can surprisingly capture the shape of several skewed projections, efficiently guiding the samples toward the slits.



Reference *GMM*

Figure 7.5.: *GMM subdivision in Complex Projections*

Nevertheless, GMMs produce less isotropic and highly non-uniformly sized regions compared to k-means. They have instability issues when the number of components k does not match the number of scene geometries, as evidenced by some extremely small regions in figs. 7.3 and 7.5.

Conclusion. Though clustering-based techniques have great potential in representing non-linear shapes, which top-down methods struggle to, we decided not to continue in this research

direction for the following reasons:

1. They require a user-set number of clusters or components k, harder to tune than the Open PGL k-d tree's maximum count threshold because the clustering result is sensitive to k.
2. Instability in the clustering process: both k-means and GMMs rely on random initialization, resulting in unpredictable spatial subdivisions. This makes it challenging to analyze their performance as well as to address the requirements of adapting to geometries and lighting in section 4.4.
3. High anisotropy and size variation of regions generated by GMMs makes them less robust for subdivisions.
4. Non-trivial incremental update: both k-means and EM fitting naturally support incremental update by adjusting the cluster centroids or the GMM parameters with new samples. However, path guiding involves the refinement of regions that requires changing the number of clusters k. Implementing advanced k-means or GMM algorithms that support the incremental update of k and ensuring their robustness needs considerable efforts.

8

Discussion

8.1. Conclusion

In this thesis, we presented multiple novel spatial subdivision schemes for path guiding. We first analyzed a few existing problems of state-of-the-art spatial subdivision techniques. Next, we motivated the requirements for an improved spatial subdivision with 2D experiments. We then broke down the k-d tree subdivision problem into two aspects: "when to split" and "where to split".

For "when to split", we derived two adaptive splitting methods by extending the k-d tree with lookahead regions. The first solution is by comparing the quality of the leaf region guiding distribution with the lookahead regions', based on cross-entropy or χ^2-divergence estimates. The second solution creates a mixture distribution from the leaf and lookahead regions. It optimizes the MIS weight and uses it to guide the splitting decision. In our experiments, we found the cross-entropy-based (CE) splitting most numerically stable and performant regarding rendering errors. We observed that in ideal cases, CE offers better rendering quality with a much lower region budget than the static baseline subdivision schemes.

For "where to split", we developed several schemes aware of the geometries or lighting in the scene: variance-scanning (VS), covariance-scanning (COVS), information-gain-scanning (IGS), and fluence-scanning (FS). They all work in a way that sorts and scans the samples through each dimension. We proposed a novel metric that proves the geometric performance of COVS and IGS. Through rendering experiments, we analyzed their pros and cons. COVS and IGS perform the best when there are regular, distinguishable geometries in the scene. FS refines the region boundaries to fit the shadows. We also found that the rendering improvements brought by the scanning algorithms are less significant than those achieved by the adaptive splitting.

Altogether, we built our full models by integrating the adaptive splitting with one of the scan-

ning algorithms: CE-VS, CE-IGS, and CE-FS. We conducted rendering experiments across various scenes to show the performance gain of the full models over Open PGL's scheme and the ablation model that only considers "where to split". We observed an improvement of around 3-15% with our most performant model in terms of the mean relative absolute error, compared to the state-of-the-art spatial subdivision.

Lastly, we explored another way of subdivision by building the spatial structure bottom-up with bounding volume hierarchies or clustering algorithms. In spite of their potential to capture fine or nonlinear shapes, we ceased pursuing this direction mainly because of the less predictable subdivisions they generate compared to those of the top-down k-d trees.

8.2. Limitations

Local optima. An intrinsic problem of all our adaptive splitting methods originates from looking at only one split ahead. The subdivision can get stuck when there are similar lighting distributions marginalized in the parent and child regions. Our current naive handling based on sample count (section 5.1.4) can slow down the subdivision process. Unfortunately, extending the candidate split depth will incur an exponential growth of lookahead regions, both computationally and implementation-wise challenging.

No recursive splits for adaptive splitting. The current implementation of our adaptive splitting cannot perform recursive splits (except those triggered by the defensive sample count threshold). When a candidate split is adopted and the new leaf nodes still contain enough samples for another candidate split, this new split cannot be immediately adopted in the current iteration. This is because, constraint by our pipeline fig. 5.3 (lower), the evaluation for this new split will not arrive until the next iteration.

Fortunately, the slowdown caused by not having recursive splits is gone over time. The average number of incoming samples per region is decreasing as we have more and more regions, leading to lower chances of triggering recursive splits.

The inter-correlation between "when and where to split". In our full models, we found that the decision of "when to split" depends on the quality of the "where to split" proposal. Adaptive splitting functions effectively when the candidate splits can benefit the guider. However, if the candidate split occurs at the wrong dimension or location (e.g. the candidate splits of the middle regions of fig. 6.15), it may take a longer time to meet the adaptive split criteria or may never meet them.

No incremental update for scanning algorithms. As explained in section 5.2.4, our scanning algorithms are purely based on the most recent wave of samples. As the subdivision goes on, the average number of samples for new splits is diminishing. At a point when there are not enough samples to run the scanning algorithms, we have to resort to Open PGL's "where to split" approach that's based on mean and variance statistics.

8.3. Outlook

Better criteria for adaptive splitting. Our adaptive splitting approaches trigger the split by only looking at the most recent divergence estimate or MIS weight. The divergence-iteration plots in section 6.1.1 already showed that there is significant noise in the evaluation of the cache. Fluctuation in the divergence can mislead the algorithm to trigger unnecessary or late splits. Our current way to counter this noise is the use of a fixed threshold D_{ths} for the entire scene. Understanding the pattern of this noise is crucial, as it will allow us to propose more robust and dynamic methods to manage the noise in the future.

Recursive splits for adaptive splitting. A possible solution to enable recursive splits for adaptive splitting is to decompose the training samples into two groups A and B, similar to the cross-validation technique used in machine learning. We can train the guiding cache with only group A while evaluating it with group B to avoid the underestimation of divergences. This way, the evaluation phase can come immediately after the training, allowing the adoption of recursive candidate splits.

Adaptive dimension selection. We envisioned one prospective method to address the third limitation. To fully de-correlate "when and where to split", an ultimate solution is to propose all possible candidate split locations and dimensions, fit the lookahead regions, evaluate all of them, and select the best candidate split. This is, of course, impractical. Yet, if we restrict to use only one candidate split position per dimension, we are left with only three pairs of lookahead regions, whose cost is affordable. We can then let the adaptive splitting algorithm determine not only the split timing but also which candidate dimension to adopt. It is hoped that this adaptive dimension selection mechanism can yield a simpler and shallower k-d tree while maintaining the same guide cache quality.

More sensible ways to combine modules. Our current full model combines adaptive splitting with only one of the scanning algorithms, leaving the user to choose which one among VS, COVS, IGS, and FS to use per scene. It would be meaningful to find a more sensible way to integrate the scanning algorithms. A possible solution is to let the model dynamically choose which scanning algorithm to use based on the scene's local information. This way, we will have the potential to combine the strength of each one at the right stage of subdivision, hopefully achieving a higher performance than using any of the algorithms statically.

For instance, we can first employ IGS to detect the main objects in the scene during the first few iterations. Then, we resort to FS to adapt the subdivision to the shadows and shades. Finally, we resort to Open PGL's splitting for a more robust splitting at smaller scene regions.

Finding a robust strategy to decide when to switch between methods will be an interesting challenge.

Equal-time study. All comparisons in section 6.2 are done with equal SPP. Due to limited time, we haven't optimized our implementations to be on par with Open PGL's original spatial

subdivision scheme. Comparing them with equal render time can be unfair, and was not done in our thesis. To understand the gain of our methods in real-world applications, we will optimize our codes and make equal-time comparisons with Open PGL.

Support for participating media. Due to time limits, we didn't explore applying the proposed subdivision schemes to the path guiding for volumetric rendering, an active research avenue in academia.

In that scenario, state-of-the-art path guiders leverage two guiding fields: one to approximate the surface radiance field, and one for the volume radiance field. The presence of participating media may introduce new sources of indirect light paths, leading to more noise in the fitting and evaluation of the cache. Cautious efforts would be required to handle this extra intricacy.

Prospectively, it would be meaningful if our work, orthogonal to other research in path guiding, could contribute to volume path guiding.

9

Acknowledgments

I would like to extend my sincere gratitude to my supervisors Sebastian Herholz, Dr. Marco Manzi, and Dr. Marios Papas, who provided me with tremendous support and guidance throughout the six months. The discovery of this fruitful problem, the development of our methodologies, the completion of the thesis, as well as my current attitude to scientific research would not have been possible without their insightful experience and invaluable help.

I am immensely grateful to my parents and my girlfriend Yixuan. Despite the distance, they have always accompanied me spiritually, giving me unwavering encouragement and emotional support. My academic passion is fueled by their unconditional love.

I would like to thank my friend Kehan, who has deeply inspired me in the world of rendering. Her valuable feedback helped me refine the writing of the thesis greatly.

Special thanks to my lab mates Shinjeong and Alan, whose companionship and assistance in math derivation facilitated certain aspects of our research.

A

Appendix

A.1. Derivation of Optimal Guiding PDF and MRSE

eq. (4.11) expands to

$$\mathrm{NEV}(q) = \mathbb{E}_x \mathrm{Var}_\omega \left[\frac{p(\omega \mid x)}{q(\omega)} \right] = \mathbb{E}_x \left[\mathbb{E}_\omega \left[\frac{p(\omega \mid x)^2}{q(\omega)^2} \right] - \mathbb{E}_\omega^2 \left[\frac{p(\omega \mid x)}{q(\omega)} \right] \right] \tag{A.1}$$

$$= \mathbb{E}_x \left[\int_\Omega \frac{p(\omega \mid x)^2}{q(\omega)^2} q(\omega) \mathrm{d}\omega - \left(\int_\Omega \frac{p(\omega \mid x)}{q(\omega)} q(\omega) \mathrm{d}\omega \right)^2 \right] \tag{A.2}$$

$$= \mathbb{E}_x \left[\int_\Omega \frac{p(\omega \mid x)^2}{q(\omega)} \mathrm{d}\omega \right] - 1 \tag{A.3}$$

The optimal PDF q^* reads

$$q^* = \underset{q \text{ s.t. } \int q(\omega)\mathrm{d}\omega = 1,\ q(\omega) > 0}{\arg\min} \mathrm{NEV}(q) \tag{A.4}$$

Expressing the minimization problem into the variational calculus form

$$q^* = \arg\min_q \int_\Omega \frac{1}{q(\omega)} \mathbb{E}_x \left[p(\omega \mid x)^2 \right] \mathrm{d}\omega + \lambda \left(\int_\Omega q(\omega) \mathrm{d}\omega - 1 \right) \tag{A.5}$$

$$= \arg\min_q \int_\Omega \frac{1}{q(\omega)} \mathbb{E}_x \left[p(\omega \mid x)^2 \right] + \lambda \left(q(\omega) - \frac{1}{|\Omega|} \right) \mathrm{d}\omega \tag{A.6}$$

$$= \arg\min_q \int_\Omega J(\omega, q, \lambda) \mathrm{d}\omega \tag{A.7}$$

q^* should satisfy the Euler-Lagrange equation $\frac{\partial J}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}\omega}\frac{\partial J}{\partial q'} = 0$, which follows

$$\frac{\partial J}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}\omega}\frac{\partial J}{\partial q'} = \frac{\partial J}{\partial q} = -\frac{1}{q^2}\mathbb{E}_x\left[p(\omega \mid x)^2\right] + \lambda = 0 \tag{A.8}$$

The optimal PDF is immediately found

$$q^*(\omega) = \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}/C \tag{A.9}$$

The normalizer C is calculated by integrating q^* to one

$$C = \int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}\mathrm{d}\omega \tag{A.10}$$

Substituting it back to eq. (4.11) , we get the minimum MRSE:

$$\mathrm{NEV}\left(q^*\right) = \left(\int_\Omega \sqrt{\mathbb{E}_x\left[p(\omega \mid x)^2\right]}\mathrm{d}\omega\right)^2 - 1 \tag{A.11}$$

A.2. Development of the Divergence Estimators

A.2.1. Marginalized Cross-Entropy

We can eliminate the ground truth PDF terms by simplifying eq. (5.14) into an expectation form

$$\begin{aligned}
\tilde{H}(q) &= \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}[H(p_{\omega|\mathbf{x}}, q_{\omega|\mathbf{x}})] && \text{(A.12)}\\
&= \mathbb{E}_{\mathbf{x}}\left[\int -p(\omega \mid \mathbf{x}) \log q(\omega \mid \mathbf{x})\mathrm{d}\omega\right] && \text{(A.13)}\\
&= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\omega\sim q_s(\cdot|\mathbf{x})}\left[-\frac{p(\omega \mid \mathbf{x})}{q_s(\omega|\ \mathbf{x})}\log q(\omega|\ \mathbf{x})\right]\right] \quad \text{rewriting integral into expectation} && \text{(A.14)}\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{L(\mathbf{x},\omega)}{\Phi(\mathbf{x})q_s(\omega|\ \mathbf{x})}\log q(\omega|\ \mathbf{x})\right] \quad \text{substituting } p(\omega \mid \mathbf{x}) && \text{(A.15)}\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\mathbb{E}[\langle L(\mathbf{x},\omega)\rangle]}{\Phi(\mathbf{x})q_s(\omega \mid \mathbf{x})}\log q(\omega \mid \mathbf{x})\right] && \text{(A.16)}\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\langle L(\mathbf{x},\omega)\rangle}{\Phi(\mathbf{x})q_s(\omega \mid \mathbf{x})}\log q(\omega \mid \mathbf{x})\right] && \text{(A.17)}\\
&= \mathbb{E}_{\mathbf{x},\omega}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\log q(\omega \mid \mathbf{x})\right] \quad \text{substituting } \frac{\langle L(\mathbf{x},\omega)\rangle}{q_s(\omega \mid \mathbf{x})} && \text{(A.18)}
\end{aligned}$$

We can build an **unbiased** MC estimator for eq. (A.18), using training samples arriving at this region eq. (4.2):

$$\tilde{H}(q) \approx \frac{1}{N}\sum_{i=1}^{N} -\frac{\langle\Phi(\mathbf{x}_i)\rangle}{\Phi(\mathbf{x}_i)}\log q(\omega_i \mid \mathbf{x}_i) \tag{A.19}$$

Plugging in the parent and child guiding distribution, we get the unbiased cross-entropy estimates for the parent and child models:

$$\langle \tilde{H}(q_p) \rangle_{\text{unbiased}} = \frac{1}{N} \sum_{i=1}^{N} -\frac{\langle \Phi(\mathbf{x}_i) \rangle}{\Phi(\mathbf{x}_i)} \log q_p(\omega_i \mid \mathbf{x}) \tag{A.20}$$

$$\langle \tilde{H}(q_c) \rangle_{\text{unbiased}} = \frac{1}{N} \sum_{i=1}^{N} -\frac{\langle \Phi(\mathbf{x}_i) \rangle}{\Phi(\mathbf{x}_i)} \log q_c(\omega_i \mid \mathbf{x}) \tag{A.21}$$

$$= \frac{1}{N} \left(\sum_{i:\ \mathbf{x}_i \in X_l} -\frac{\langle \Phi(\mathbf{x}_i) \rangle}{\Phi(\mathbf{x}_i)} \log q_l(\omega_i \mid \mathbf{x}) + \sum_{i:\ \mathbf{x}_i \in X_r} -\frac{\langle \Phi(\mathbf{x}_i) \rangle}{\Phi(\mathbf{x}_i)} \log q_r(\omega_i \mid \mathbf{x}) \right) \tag{A.22}$$

Note that both estimators are evaluated on the same spatial domain-parent region X_p.

Unfortunately, we have no access to the ground truth fluence $\Phi(\mathbf{x}_i)$ during rendering. So instead, we attempt to avoid this term by building **biased** estimators.

We continue from eq. (A.18), aiming to replace the ground truth fluence with its estimate

$$\tilde{H}(q) = \mathbb{E}_{\mathbf{x},\omega} \left[-\frac{\langle \Phi(\mathbf{x}) \rangle}{\Phi(\mathbf{x})} \log q(\omega \mid \mathbf{x}) \right] \tag{A.23}$$

$$\doteq \frac{\mathbb{E}_{\mathbf{x},\omega}[-\langle \Phi(\mathbf{x}) \rangle \log q(\omega \mid \mathbf{x})]}{\mathbb{E}_{\mathbf{x}}[\Phi(\mathbf{x})]} \qquad \text{bias introduced} \tag{A.24}$$

$$= \frac{\mathbb{E}_{\mathbf{x},\omega}[-\langle \Phi(\mathbf{x}) \rangle \log q(\omega \mid \mathbf{x})]}{\mathbb{E}_{\mathbf{x},\omega}[\langle \Phi(\mathbf{x}) \rangle]} \tag{A.25}$$

$$\doteq \frac{\frac{1}{N} \sum_{i=1}^{N} -\langle \Phi(\mathbf{x}_i) \rangle \log q(\omega_i \mid \mathbf{x}_i)}{\frac{1}{N} \sum_{i=1}^{N} \langle \Phi(\mathbf{x}_i) \rangle} \qquad \text{biased estimator} \tag{A.26}$$

$$= \frac{\sum_{i=1}^{N} -\langle \Phi(\mathbf{x}_i) \rangle \log q(\omega_i \mid \mathbf{x}_i)}{\sum_{i=1}^{N} \langle \Phi(\mathbf{x}_i) \rangle} \tag{A.27}$$

Where the $\doteq$ steps eqs. (A.24) and (A.26) introduce bias, since

$$\mathbb{E}\left[\frac{X}{Y}\right] \neq \frac{\mathbb{E}X}{\mathbb{E}Y} \tag{A.28}$$

$$\frac{1}{\mathbb{E}X} \neq \mathbb{E}\left[\frac{1}{\frac{1}{N}\sum_{i=1}^{N} x_i}\right], \quad x_i \overset{\text{i.i.d.}}{\sim} p_X,\ X > 0 \tag{A.29}$$

eq. (A.29) is a result of Jensen's inequality $\frac{1}{\mathbb{E}X} = \frac{1}{\mathbb{E}\overline{X}} \leqslant \mathbb{E}\left[\frac{1}{\overline{X}}\right]$.

Good, eq. (A.27) gives us a formula to estimate the marginalized cross-entropy directly with sample weight $\langle \Phi(\mathbf{x}_i) \rangle$ and the guiding PDF $q(\omega_i \mid \mathbf{x}_i)$ that can be evaluated.

However, if we look closely at step eq. (A.24), it is essentially replacing the fluence with its average across the whole region to be able to extract it out of the expectation, which is a very crude approximation.

Let's reduce this bias by making a more "fine-grained" approximation on the child half-spaces. Restarting from eq. (A.18), we have

$$\tilde{H}(q) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\log q(\omega \mid \mathbf{x})\right] \tag{A.30}$$

$$= \mathbb{P}(\mathbf{x}\in X_l)\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_l}}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\log q(\omega \mid \mathbf{x})\right] + \mathbb{P}(\mathbf{x}\in X_r)\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_r}}\left[-\frac{\langle\Phi(\mathbf{x})\rangle}{\Phi(\mathbf{x})}\log q(\omega \mid \mathbf{x})\right] \tag{A.31}$$

$$\doteq \mathbb{P}(\mathbf{x}\in X_l)\frac{\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_l}}[-\langle\Phi(\mathbf{x})\rangle\log q(\omega \mid \mathbf{x})]}{\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_l}}[\langle\Phi(\mathbf{x})\rangle]} + \mathbb{P}(\mathbf{x}\in X_r)\frac{\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_r}}[-\langle\Phi(\mathbf{x})\rangle\log q(\omega \mid \mathbf{x})]}{\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_r}}[\langle\Phi(\mathbf{x})\rangle]} \tag{A.32}$$

$$\doteq \frac{N_l}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_l} -\langle\Phi(\mathbf{x}_i)\rangle\log q(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_l}\langle\Phi(\mathbf{x}_i)\rangle} + \frac{N_r}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_r} -\langle\Phi(\mathbf{x}_i)\rangle\log q(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_r}\langle\Phi(\mathbf{x}_i)\rangle} \tag{A.33}$$

Likewise, steps with $\doteq$ incur bias. N_l and N_r denote the number of samples falling into X_l and X_r.

Plugging in the parent and child guiding distributions yields

$$\langle\tilde{H}(q_p)\rangle = \frac{N_l}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_l} -\langle\Phi(\mathbf{x}_i)\rangle\log q_p(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_l}\langle\Phi(\mathbf{x}_i)\rangle} + \frac{N_r}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_r} -\langle\Phi(\mathbf{x}_i)\rangle\log q_p(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_r}\langle\Phi(\mathbf{x}_i)\rangle} \tag{A.34}$$

$$\langle\tilde{H}(q_c)\rangle = \frac{N_l}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_l} -\langle\Phi(\mathbf{x}_i)\rangle\log q_l(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_l}\langle\Phi(\mathbf{x}_i)\rangle} + \frac{N_r}{N}\frac{\sum_{i:\ \mathbf{x}_i\in X_r} -\langle\Phi(\mathbf{x}_i)\rangle\log q_r(\omega_i \mid \mathbf{x}_i)}{\sum_{i:\ \mathbf{x}_i\in X_r}\langle\Phi(\mathbf{x}_i)\rangle} \tag{A.35}$$

A.2.2. Marginalized χ^2-Divergence

We express eq. (5.21) in the expectation form, to get rid of p

$$\tilde{D}_{\chi^2}(q) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_X}\left[\int \frac{p(\omega \mid \mathbf{x})^2}{q(\omega\mid \mathbf{x})}\mathrm{d}\omega - 1\right] \tag{A.36}$$

$$= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\omega\sim q_s(\cdot\mid\mathbf{x})}\left[\frac{p(\omega \mid \mathbf{x})^2}{q_s(\omega \mid \mathbf{x})q(\omega \mid \mathbf{x})}\mathrm{d}\omega\right]\right] - 1 \tag{A.37}$$

$$= \mathbb{E}_{\mathbf{x},\omega}\left[\frac{L(\mathbf{x},\omega)^2}{\Phi(\mathbf{x})^2 q_s(\omega \mid \mathbf{x})q(\omega \mid \mathbf{x})}\right] - 1 \qquad \text{substituting } p(\omega \mid \mathbf{x}) \tag{A.38}$$

Unlike the KL divergence case, an unbiased estimator for eq. (A.38) cannot be easily derived. We have to move on to build **biased** estimators.

$$\tilde{D}_{\chi^2}(q) \doteq \mathbb{E}_{\mathbf{x},\omega}\left[\frac{\langle L(\mathbf{x},\omega)\rangle^2}{\Phi(\mathbf{x})^2 q_s(\omega \mid \mathbf{x})q(\omega \mid \mathbf{x})}\right] - 1 \tag{A.39}$$

$$= \mathbb{E}_{\mathbf{x},\omega}\left[\frac{\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega \mid \mathbf{x})}{\Phi(\mathbf{x})^2 q(\omega \mid \mathbf{x})}\right] - 1 \qquad \text{substituting } \langle L(\mathbf{x},\omega)\rangle^2 \tag{A.40}$$

The bias in the first step arises from

$$\mathbb{E}\left[X^2\right] \neq \mathbb{E}[X]^2 \tag{A.41}$$

Again, to avoid the ground truth fluence term, we have to further split the expectation of division, introducing the bias shown in eqs. (A.28) and (A.29). Finally, we arrive at a tractable MC estimator

$$\tilde{D}_{\chi^2}(q) \doteq \mathbb{E}_{\mathbf{x},\omega}\left[\frac{\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega \mid \mathbf{x})/q(\omega \mid \mathbf{x})}{\Phi(\mathbf{x})^2}\right] - 1 \tag{A.42}$$

$$\doteq \frac{\mathbb{E}_{\mathbf{x},\omega}\left[\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega \mid \mathbf{x})/q(\omega \mid \mathbf{x})\right]}{\mathbb{E}_{\mathbf{x}}\left[\Phi(\mathbf{x})^2\right]} - 1 \tag{A.43}$$

$$\doteq \frac{\mathbb{E}_{\mathbf{x},\omega}\left[\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega \mid \mathbf{x})/q(\omega \mid \mathbf{x})\right]}{\mathbb{E}_{\mathbf{x}}[\langle\Phi(\mathbf{x})\rangle]^2} - 1 \tag{A.44}$$

$$\doteq \frac{\frac{1}{N}\sum_{i=1}^{N}\langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega|\ \mathbf{x}_i)/q(\omega \mid \mathbf{x})}{\left(\frac{1}{N}\sum_{i=1}^{N}\langle\Phi(\mathbf{x}_i)\rangle\right)^2} - 1 \tag{A.45}$$

$$= \frac{N\sum_{i=1}^{N}\langle\Phi(\mathbf{x}_i)\rangle^2 q_s(\omega|\ \mathbf{x}_i)/q(\omega \mid \mathbf{x})}{\left(\sum_{i=1}^{N}\langle\Phi(\mathbf{x}_i)\rangle\right)^2} - 1 \tag{A.46}$$

Likewise, we use the same trick to reduce the bias in eq. (A.46), by decomposing the expectation eq. (A.40) to the child half-spaces

$$\begin{aligned}\mathbb{E}_{\mathbf{x},\omega}\left[\frac{\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega \mid \mathbf{x})}{\Phi(\mathbf{x})^2 q(\omega|\ \mathbf{x})}\right] =& \mathbb{P}(\mathbf{x} \in X_l)\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_l}}\left[\frac{\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega|\ \mathbf{x})}{\Phi(\mathbf{x})^2 q(\omega|\ \mathbf{x})}\right] \\ &+\mathbb{P}(\mathbf{x} \in X_r)\mathbb{E}_{\mathbf{x}\sim\hat{p}_{X_r}}\left[\frac{\langle\Phi(\mathbf{x})\rangle^2 q_s(\omega|\ \mathbf{x})}{\Phi(\mathbf{x})^2 q(\omega|\ \mathbf{x})}\right]\end{aligned} \tag{A.47}$$

A.3. Proof of the Scale-Invariance Property of Mutual Information

From eqs. (5.68) and (5.70), the mutual information of samples $S = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ of a split (d, t) reads

$$\mathrm{IG}(d,t) \coloneqq H(S) - \left(\frac{N_l}{N}H(S_l(d,t)) + \frac{N_r}{N}H(S_r(d,t))\right) \tag{A.48}$$

$$H(S) = \frac{1}{2}\log\det\mathrm{Cov}(S) \tag{A.49}$$

In the scope of this proof, we omit the ϵ term.

We will show that, if the sample positions undergo a scale transform $S' = AS,\ A = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$, the resulting new mutual information **at the same normalized split location** $\mathrm{IG}'(d, \lambda_d t)$ will stay the same.

Let's first consider the entropy of the new samples

$$H(S') = H(AS) = \frac{1}{2}\log\det \mathrm{Cov}(AS) \tag{A.50}$$

Using the linear transform property of the covariance matrix

$$\mathrm{Cov}(AS) = A\mathrm{Cov}(S)A^{\mathrm{T}} \tag{A.51}$$

we have

$$\begin{aligned} H(AS) &= \frac{1}{2}\log\det(A\mathrm{Cov}(S)A^{\mathrm{T}}) && \text{(A.52)}\\ &= \frac{1}{2}\log\left((\det A)^2 \det \mathrm{Cov}(S)\right) && \text{(A.53)}\\ &= \log\det A + \frac{1}{2}\log\det \mathrm{Cov}(S) && \text{(A.54)}\\ &= \log\det A + H(S) && \text{(A.55)} \end{aligned}$$

Meanwhile, we notice splitting S' at $\lambda_d t$ gives the same fractions of left and right samples as before the transform. Namely, $N'_l = N_l, N'_r = N_r, S'_l = AS_l, S'_r = AS_r$. Plugging these terms into eqs. (A.48) and (A.49), and applying eq. (A.55), we have

$$\begin{aligned} \mathrm{IG}'(d, \lambda_d t) &= H(AS) - \left(\frac{N_l}{N}H(AS_l) + \frac{N_r}{N}H(AS_r)\right) && \text{(A.56)}\\ &= \log\det A + H(S) - \left(\frac{N_l}{N}(\log\det A + H(S_l)) + \frac{N_r}{N}(\log\det A + H(S_r))\right) && \text{(A.57)}\\ &= H(S) - \left(\frac{N_l}{N}H(S_l) + \frac{N_r}{N}H(S_r)\right) && \text{(A.58)}\\ &= \mathrm{IG}(d, t) && \text{(A.59)} \end{aligned}$$

Therefore, the normalized mutual information curve (IG vs. normalized split location) remains unchanged after scale transforms. In other words, the mutual information is scale-invariant.

Bibliography

[BAJ08] Brian C Budge, John C Anderson, and Kenneth I Joy. Caustic forecasting: Unbiased estimation of caustic lighting for global illumination. In *Computer Graphics Forum*, volume 27, pages 1963–1970. Wiley Online Library, 2008.

[DK17] Ken Dahm and Alexander Keller. Learning light transport the reinforced way. In *ACM SIGGRAPH 2017 Talks*, pages 1–2. 2017.

[DWL23] Honghao Dong, Guoping Wang, and Sheng Li. Neural parametric mixtures for path guiding. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023.

[GHFB13] Yan Gu, Yong He, Kayvon Fatahalian, and Guy Blelloch. Efficient bvh construction via approximate agglomerative clustering. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 81–88, 2013.

[HD22] Sebastian Herholz and Addis Dittebrandt. Intel® open path guiding library, 2022. https://www.openpgl.org.

[Hes95] Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

[Jen95] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Rendering Techniques' 95: Proceedings of the Eurographics Workshop in Dublin, Ireland, June 12–14, 1995 6*, pages 326–335. Springer, 1995.

[KA13] Tero Karras and Timo Aila. Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 89–99, 2013.

[Kaj86] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Knu97] Donald Ervin Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

[Loh11] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[MESK22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[MGN17] Thomas Müller, Markus Gross, and Jan Novák. Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum*, volume 36, pages 91–100. Wiley Online Library, 2017.

[MMR+19] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (ToG)*, 38(5):1–19, 2019.

[PJH23] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering, fourth edition: From Theory to Implementation*. MIT Press, 2023.

[RGH+20] Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. Variance-aware path guiding. *ACM Transactions on Graphics (TOG)*, 39(4):151–1, 2020.

[RHL20] Lukas Ruppert, Sebastian Herholz, and Hendrik PA Lensch. Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG)*, 39(4):147–1, 2020.

[SL06] Joshua Steinhurst and Anselmo Lastra. Global importance sampling of glossy surfaces using the photon map. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 133–138. IEEE, 2006.

[SSK22] Mateu Sbert and László Szirmay-Kalos. Robust multiple importance sampling with tsallis φ-divergences. *Entropy*, 24(9):1240, 2022.

[Vea98] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[VHH+19] Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. Path guiding in production. In *ACM SIGGRAPH 2019 Courses*, pages 1–77. 2019.

[VKŠ+14] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. Online learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[Wel62] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[ZXS+21] Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Hen-

rik Wann Jensen, Hao Su, and Ravi Ramamoorthi. Photon-driven neural reconstruction for path guiding. *ACM Transactions on Graphics (TOG)*, 41(1):1–15, 2021.

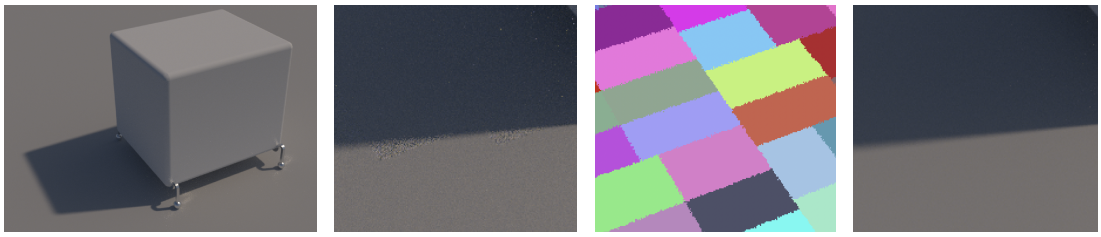
[ZZ19] Quan Zheng and Matthias Zwicker. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, volume 38, pages 169–179. Wiley Online Library, 2019.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

Master Thesis

Spatial Subdivision for Path Guiding



Left to right: test scene, guided path tracing, guiding cache ids, and ground truth.

Introduction

Path guiding is an importance-sampling technique for Monte-Carlo-based rendering algorithms integrating global information of a scene's light distribution into local sampling decisions. Compared to traditional importance sampling methods that only consider local information, such as the BSDF, path guiding improves sampling efficiency for complex global light transport effects such as indirect diffuse/glossy illumination, caustics, and multiple scattering in volumes. A crucial component of every path-guiding algorithm is the representation of the scene's 5D light distribution. The 5D distribution is commonly separated into spatial 3D structures containing a 2D approximation of each leaf node's directional incoming light distribution.

While much work has been done on optimizing the directional representations (e.g., quad-trees, VMMs, or parallax-aware VMMs), only a little work dealt with optimizing the spatial structures and their subdivision schemes. Most current algorithms use a kd-tree in combination with a somewhat naive subdivision scheme based on the number of (valid) samples. Since these subdivision schemes do not consider the scene's actual light distribution characteristics, the resulting guiding structure can be sub-optimal, leading to inefficient sampling behavior. Examples are regions of higher variance caused by large areas covering high-frequency changes (e.g., shadow boundaries or smaller caustics) or many smaller regions covering the same or similar distribution, leading to unnecessary memory consumption and lookup times.

This thesis aims to develop and evaluate more advanced spatial structures and subdivision schemes countering the previous limitations of the current naive approaches.

Task

- Exploring the limitations of current spatial subdivision schemes in the context of path guiding, such as late refinement, early refinement, unnecessary refinement, and suboptimal splitting planes.
    - Collecting a variety of representative scenes with complex light distributions or geometries which current subdivision methods fail to address.
    - Diagnosing what problems are present in each scene and propose approaches to improve the subdivision.
- Identifying metrics to quantify the quality of a subdivision scheme.
    - Exploring sample statistics that are useful and robust for improving the subdivision criterion.
    - Exploring buffers available during rendering that can help subdivision decisions. (e.g. irradiance and normal)
- Developing new subdivision schemes to address the identified limitations.
- A qualitative and quantitative evaluation of the developed subdivision methods on the collected scenes (e.g., equal sample, equal quality, and equal time comparisons).

Remarks

This thesis is assigned to Fengshi Zheng. A written report and an oral presentation conclude the thesis. The thesis will be overseen by Prof. Markus Gross and supervised by Sebastian Herholz (Intel), Dr. Marco Manzi (DRS), and Dr. Marios Papas (DRS).

Dates

The project starts December 18, 2023 and ends on June 30, 2024.