Lossless Basis Expansion for Gradient-Domain Rendering

by

Qiqin Fang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

© Qiqin Fang 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Gradient-domain rendering utilizes difference estimates with shift mapping to reduce variance in Monte Carlo rendering. Such difference estimates are effective under the assumption that pixels for difference estimates have similar integrands. This assumption is often violated because it is common to have spatially varying BSDFs with material maps, which potentially result in a very different integrand per pixel. We introduce an extension of gradient-domain rendering that effectively supports such per-pixel variation in BSDFs based on basis expansion. Basis expansion for BSDFs has been used extensively in other problems in rendering, where the goal is to approximate a given BSDF by a weighted sum of predefined basis functions. We instead utilize lossless basis expansion, representing a BSDF without any approximation by adding the remaining difference in the original basis expansion. This lossless basis expansion allows us to cancel more terms via shift mapping, resulting in low variance difference estimates even with per-pixel BSDF variation. We also extend the Poisson reconstruction process to support this basis expansion. Regular gradient-domain rendering can be expressed as a special case of our extension, where the basis is simply the BSDF per pixel (i.e., no basis expansion). We provide proof-of-concept experiments and showcase the effectiveness of our method for scenes with highly varying material maps. Our results show noticeable improvement over regular gradient-domain rendering under both L^1 and L^2 reconstructions. The resulting formulation via basis expansion essentially serves as a new way of path reuse among pixels in the presence of per-pixel variation.

Acknowledgements

I would first like to express my gratitude to my supervisor, Toshiya Hachisuka, for his guidance throughout the completion of this thesis. I appreciate his dedication to brainstorming ideas during our meetings and providing feedback on nearly every thought I shared. His efforts not only made this thesis possible but also helped me overcome challenges as I transitioned from industry to academia.

I would also like to extend my thanks to my parents for their unwavering support, both spiritually and financially, when I decided to pursue a research career.

I thank Xiaochun Tong for the implementation advice of our baseline in using LuisaRenderer. I would also like to thank Xin Peng for her valuable suggestions on using Adobe Illustrator to create the figures for this thesis.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

List of Figures vii

1 Introduction 1

2 Background 4

2.1 Path Tracing . . . 4
2.2 Bidirectional Scattering Distribution Function . . . 6
2.3 The Rendering Equation . . . 7
2.4 Path Space Formulation . . . 7
2.5 Monte-Carlo Integration . . . 9
2.5.1 Importance Sampling . . . 10
2.5.2 Multiple Importance Sampling . . . 11
2.6 Gradient-Domain Rendering . . . 12
2.6.1 Overview . . . 12
2.6.2 Shift Mapping . . . 13
2.6.3 Difference Estimator . . . 16
2.6.4 Analysis . . . 19

3 Problem Statement **20**

4 Method **22**

4.1 Basis Expansion 24

4.2 Lossless Basis Expansion. 25

4.3 Reconstruction 27

4.4 Application to Rendering 28

5 Implementation **33**

5.1 Baseline. 33

5.2 Reconstruction. 33

5.3 Mixture with G-PT. 34

5.4 Choice of Basis. 34

6 Results **35**

7 Limitations and future work **43**

References **45**

List of Figures

1.1 Our result using lossless basis expansion compared with traditional gradient-domain rendering 3

2.1 The forward path tracing process: From the light source, rays are traced and bounced in the scene, eventually hitting the camera. 5

2.2 2D illustration of BSDF: The incoming radiance will be scattered in all directions based on distribution proportional to the BSDF evaluated in the incoming direction. 6

2.3 Random sequence replay: The base path shifted (blue) to different pixels. Note that even with simple geometry changes the path (red) can become very far away from the base path. 14

2.4 Vertex reconnection: the second vertex of every shifted path is connected to the second vertex of the base path. This forms geometrically more correlated paths but may lead to low radiance if the surface type of the first vertex or the second vertex (or both) is glossy. 15

2.5 Shift mapping between different integration domains 17

3.1 Simple failure mode of gradient-domain rendering. Top right shows the integration target for lighter and darker blocks of the checkerboard pattern, middle and bottom row shows the reconstruction using per-pixel Monte Carlo integration, regular gradient-domain rendering, and our method and their respective RMSE compared with ground truth. Regular gradient-domain rendering suffers from artifacts due to high variance difference estimates around edges. Our method almost perfectly reconstructs the ground truth. 21

4.1 Overview of our method. We propose to expand a part of the integrand $g(x)$ that significantly varies across pixels into a weighted sum of lossless basis functions. The resulting basis expansion is designed to satisfy the similarity assumption between neighboring pixels i, j, making difference estimates effective with shift mapping. 27

4.2 Top: Comparison of our method with different orders of lossless basis alongside G-PT, with 128 samples per pixel under L^1 reconstruction. Bottom: Respective Convergence MAPE plot. Our method handles both the albedo and normal map better with higher-order bases. 31

4.3 Our result (128 spp) with/without mixing with G-PT. Our method works well for continuous surfaces with high-frequency texture maps. Artifacts that appear around geometry discontinuities could be effectively handled by combining our reconstruction with G-PT. Note that our method under L^1 reconstruction performs well even without such a combination. 32

6.1 Cornell-box test scenes with high-frequency albedo, normal, and roughness variation respectively. The top row shows the rendering result of G-PT and our method with spherical harmonics. The bottom row shows the corresponding MAPE error image. All scenes are rendered with 128 samples by our method and 147 samples by G-PT for same-time comparison. We see that with high-frequency albedo and normal map toggled, our method shows consistent improvement. When the roughness map is toggled our method performs comparable to G-PT, while the error is relatively low compared to the other two cases. 37

6.2 Convergence plot (MAPE vs. Time) for high-frequency albedo, normal, and roughness toggled respectively with the Cornell box scene. The convergence plot with red represents G-PT, green for our method with box function, and blue for our method with spherical harmonics. Notice the error in the roughness toggled plot while G-PT performs better, the error is relatively low compared to the other cases. 38

6.3 Visualization of error images with G-PT and our reconstruction with gradually changing roughness from 0.2 to 1.0. Our method still shows variance reduction even in low roughness settings with high-frequency albedo and normal maps. While a higher roughness value allows more informative basis expansion which will result in better variance reduction overall. 39

6.4 Equal time visual comparison reconstruction result for G-PT, our method with spherical harmonics and box basis. We include rendered result and error image visualization with MAPE metric for both L^1 (red box) and L^2 (green box) reconstruction. corresponding MAPE and RelMSE for the full image are provided as numerical reference. In almost all cases our method with spherical harmonics performs best both visually and numerically. Note that in most of the cases box function still outperforms regular G-PT. . . . 40

6.5 The scene on the left (Living-Room) indeed contains high-frequency information. While the lack of visible features and dark texture limits the qualitative improvement, we can still visualize the quantitative improvement from the error image. The scene on the right (Bathroom) lacks high-frequency features. Although the tiles on the floor are patterned and have similar colors, they dominate many pixels, restricting the benefit of our algorithm. It is worth noting that our method with spherical harmonics (SH) still reduces error for the wooden pattern on the side of the bathtub, and overall, it still performs numerically better than the baseline G-PT. 41

6.6 The convergence plot for both L^1 and L^2 reconstructions is presented. Dashed lines represent L^1, and solid lines represent L^2. It is evident that under the same reconstruction loss, our method with both choices of basis consistently outperforms G-PT. 42

Chapter 1

Introduction

Gradient-domain rendering [16, 14, 10] is an efficient rendering algorithm for scenes with smooth features. The main idea is to estimate the differences between neighboring pixels, in addition to pixels themselves, and reconstruct the final image based on those estimates. Following work [18, 9, 6, 25, 3, 19] extended the vanilla gradient-domain rendering technique to nearly all popular Monte Carlo rendering methods, providing significant variance reduction over regular Monte Carlo counterparts in many cases.

Gradient-domain rendering improves over regular Monte Carlo rendering mainly because of low-variance difference estimates with correlated samples among neighboring pixels. Correlated sampling is achieved by a technique called *shift mapping*, which maps a sample in one pixel to a similar sample in a neighboring pixel. Difference estimates by shift mapping will be effective when neighboring pixels have similar integrands because samples before and after shift mapping have similar contributions. Such contributions cancel each other effectively in Monte Carlo difference estimates. In many practical scenarios, however, this assumption is often violated because it is common to have spatial variation of albedo or shading normals over the image. While gradient-domain rendering can still render images even in such cases, its performance degrades. A practical trick of factoring out albedo [32] before rendering does not work if per-pixel variation come from shading normals.

We propose an extension of gradient-domain rendering based on basis expansion of BSDFs. Our extension can handle per-pixel variation of albedo and shading normals while maintaining the benefits of gradient-domain rendering. We consider an integrand as a product of two functions - one that can vary significantly across pixels and the other that is assumed to be similar among pixels. An example of such an integrand is a product of a BSDF and incident illumination, where the BSDF can vary rapidly among pixels, but

incident illumination is generally smooth over pixels. We propose to apply lossless basis expansion to the fast-varying function of the product to capture variation among pixels as a variation of coefficients on the bases. Unlike regular basis expansion, our lossless basis expansion results in a exact recovery of a given function since it adds back the remaining differences (residuals) to the basis functions. The product of this lossless basis function and the slowly varying part of the integrand will be similar among neighboring pixels, which effectively allows us to cancel more terms via shift mapping in gradient-domain rendering. We also re-design the Poisson reconstruction in gradient-domain rendering to support this basis expansion and to reconstruct the final image by taking into account the results of all the basis integrals simultaneously. Gradient-domain rendering is a special case of our formulation where no basis expansion is performed. Our extended gradient-domain rendering can resolve common artifacts in gradient-domain rendering under per-pixel variations, making it applicable to a larger set of practical scenes. Figure 1.1 shows one of the comparisons of our method with order 2 spherical harmonics against regular gradient-domain rendering. Our result accurately captures per-pixel variation while keeping the benefits of gradient-domain rendering.

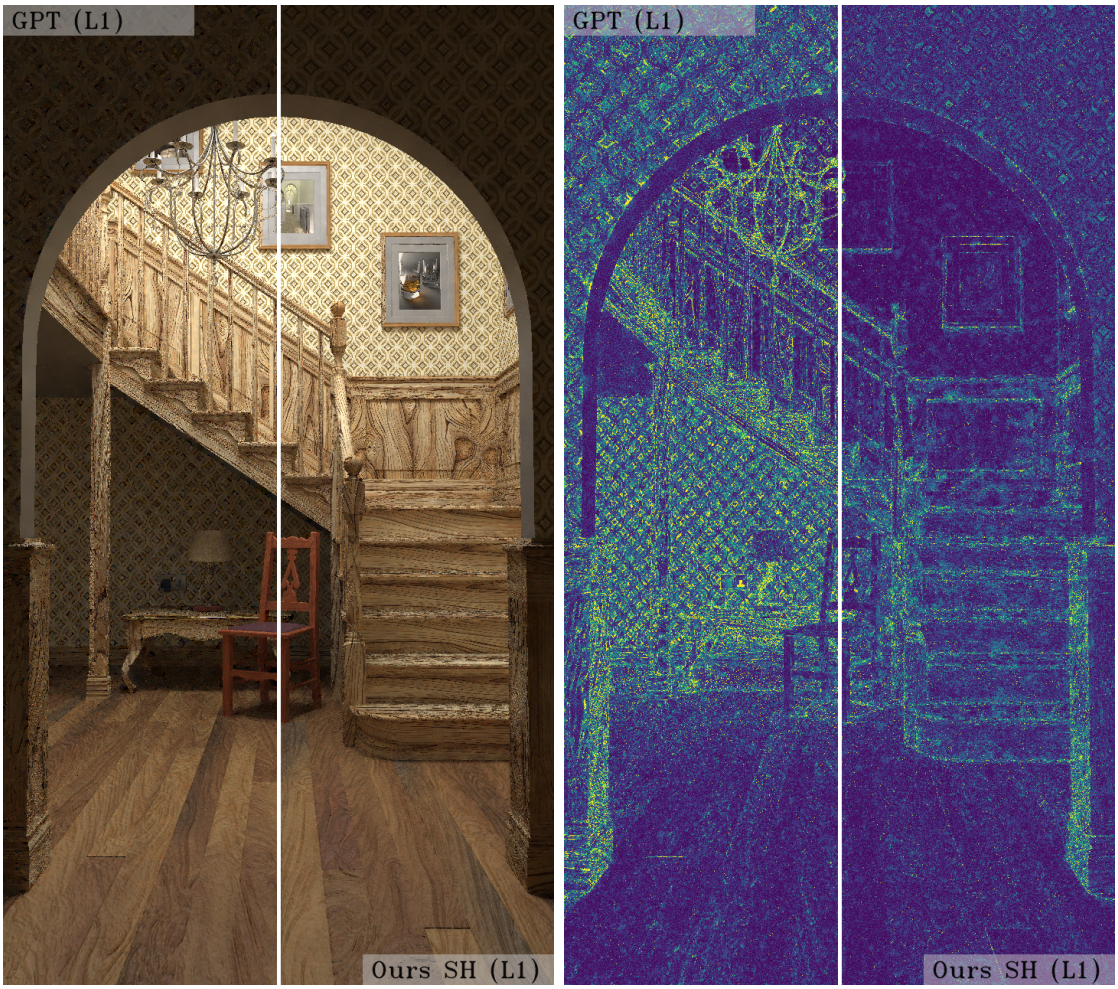


Figure 1.1: *Our result using lossless basis expansion compared with traditional gradient-domain rendering*

Chapter 2

Background

This chapter aims to fill the readers with the background of light transport simulation and gradient-domain rendering. We provide a gentle introduction to path tracing and mathematical details to help the reader understand the rendering equation, its path space formulation, and the Monte Carlo method used to solve the equation. We also explain gradient-domain path tracing in detail, which acts as a baseline for our method.

2.1 Path Tracing

In movie and game production, rendering realistic images/videos with virtually created scenes is crucial. To achieve that, it is typical to simulate the light transport behavior in a physically accurate manner. A common practice in light transport simulation is to assume lights travel through space as rays. The process can be informally described as follows:

1. A light ray is first emitted from a light source, travels through space, and hits a surface.
2. The surface absorbs part of the ray's energy, reflects the light ray if the surface is not transparent, or refracts if the light ray enters a different media passing through the surface.
3. The light ray travels through space until it eventually hits the camera sensor.

as in Figure 2.1. We call this process *forward ray tracing* as it simulates the physical behavior of light rays starting from the light source. This process can mostly accurately

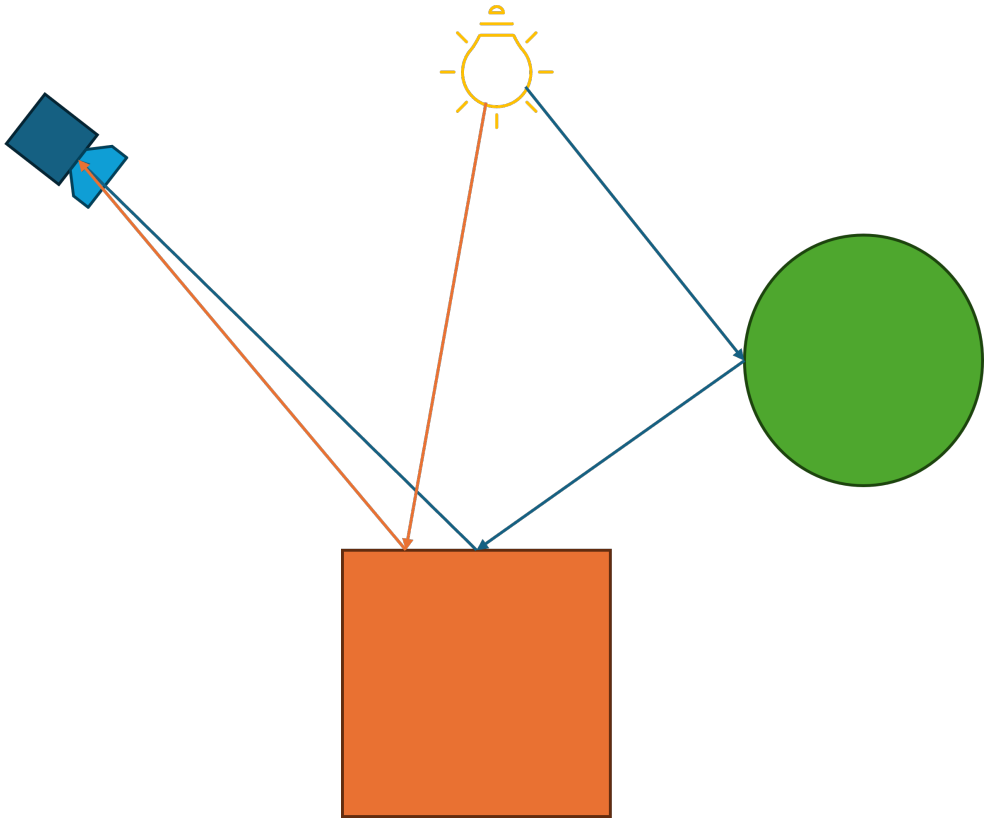


Figure 2.1: *The forward path tracing process: From the light source, rays are traced and bounced in the scene, eventually hitting the camera.*

simulate the color of each pixel in the image, but it is highly inefficient. This is because real-life cameras usually have only small holes. Lights have to eventually pass through that small hole and hit the camera sensor to contribute color to the image. In the extreme case, It is almost impossible to render an image with a pinhole camera with forward ray tracing as the probability that a ray passes through an infinitesimal camera hole is zero.

Fortunately, the path from light to the camera could be backward traced and form the same path. *Backward ray tracing* is introduced. It follows almost the same process as forward ray tracing except that the ray starts from the camera sensor. This ensures that any path that has been constructed will hit the sensor and thus contribute some color to the image.

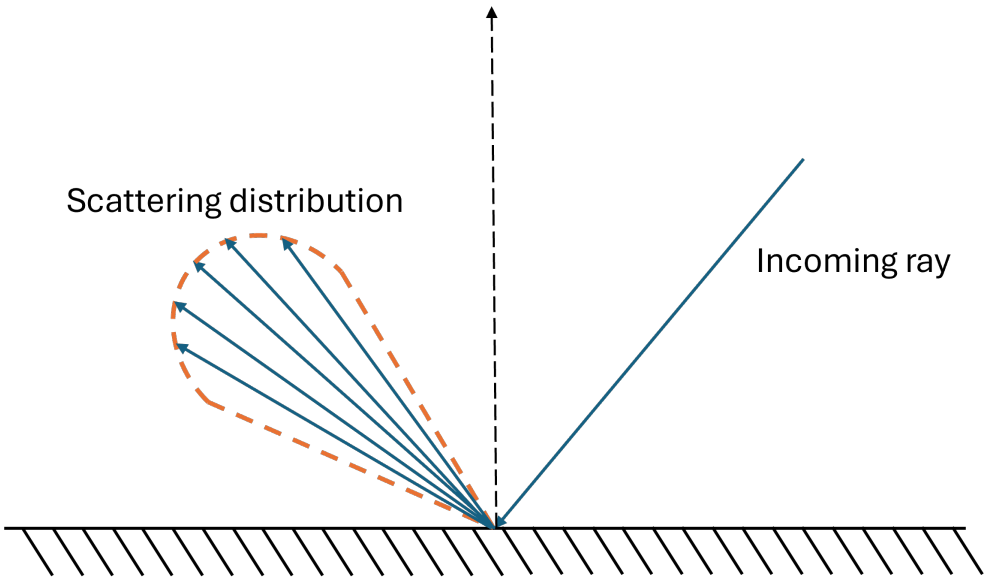


Figure 2.2: *2D illustration of BSDF: The incoming radiance will be scattered in all directions based on distribution proportional to the BSDF evaluated in the incoming direction.*

2.2 Bidirectional Scattering Distribution Function

While we described the interactions of light and surface as deterministic events of either reflection or refraction, it is not practical to simulate light in such a setting considering realistic objects are usually rough, and perfect reflection or refraction only happens at a microscopic level. In computer graphics, it is almost impossible to explicitly represent such microstructure, not to mention that people often do not care about such detail if it is not visible to human eyes.

To encounter that, *Bidirectional Reflectance Distribution Function*, or BRDF, has been used to model the reflectance events that happen within a very small region, simplified as a single point, on the surface. Formally, BRDF at a single point x with normal n can be described as a 4D function $f_r(\omega_o, \omega_i)$ with respect to incoming light ray direction ω_i and outgoing direction ω_o, such that it satisfies the following conditions:

$$\int f_r(\omega_o, \omega_i) \mathrm{d}\omega_i \leq 1 \tag{2.1}$$

$$f_r(\omega_o, \omega_i) = f_r(\omega_i, \omega_o) \tag{2.2}$$

$$f_r(\omega_o, \omega_i) \geq 0 \tag{2.3}$$

where the first condition ensures the preservation of energy, the second term obeys the Helmholtz reciprocity, and the third term ensures positivity. Similarly, *Bidirectional Transmittance Distribution Function*, or BTDF is used to describe transmission events, except that BTDF does not obey the reciprocity property. *Bidirectional Scattering Distribution Function*, or BSDF, denote f_s, takes both BRDF and BTDF into account and models the scattering event around the full spherical domain.

There are many different BSDFs created to represent different kinds of surfaces. For this thesis, we won't be able to cover the details for any specific BSDFs. However, as we will later classify surface types into *diffuse*, *glossy*, and *specular*, it is worth noting that these properties can be fully derived from the BSDFs.

2.3 The Rendering Equation

Kajiya [11] has introduced a more general mathematical formulation for rendering, known as the rendering equation. Specifically, giving a surface point x, surface normal $\mathbf{n}$, and an out-going direction ω_o, the out-going radiance can be computed by solving the following integration over all incoming direction ω_i:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega L_i(x, \omega_i) f_s(x, \omega_o, \omega_i) \left|\mathbf{n} \cdot \omega_i\right| \mathrm{d}\omega_i \tag{2.4}$$

where $L_e(x, \omega_o)$ is the emission from point x to direction ω_o, $f_s(x, \omega_o, \omega_i)$ represents the BSDF at point x, the integrand is weighted by the cosine of the angle between the surface normal and the incoming direction to adjust for quantity change.

While at first glance this integration domain is only over the spherical domain, there is more to it. The incoming radiance L_i has to be computed by solving the same equation $L_i(x, \omega_i) = L_o(x', -\omega_i)$ at the secondary hit point x', which we found by tracing a ray from x through direction ω. This process goes recursively, thus a nested integral

2.4 Path Space Formulation

Veach's thesis [26] in addition has formulated rendering into a path space integration, such that it directly integrates the path contribution weighted by the pixel filter, over all possible

paths that eventually contribute to the pixel. First, the modification starts by changing the notation of the rendering equation to avoid taking direction into account. Specifically,

$$\begin{aligned} L(p_1, \omega_o) &\Rightarrow L(p_1 \to p_0) \\ f(p_1, \omega_o, \omega_i) &\Rightarrow f(p_2 \to p_1 \to p_0) \end{aligned} \tag{2.5}$$

where the normalized vector

$$\frac{p_2 - p_1}{\|p_2 - p_1\|} \equiv \omega_i, \frac{p_0 - p_1}{\|p_0 - p_1\|} \equiv \omega_o \tag{2.6}$$

. Unlike the original rendering equation formulation, now p and p_1 may take any location on surfaces, visibility check must also be there to ensure the validity of the path:

$$V(p_0 \leftrightarrow p_1) = \begin{cases} 1 & \text{if } p_0 \text{ and } p_1 \text{ are visible} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

Now the rendering equation can be written as integration over the surface:

$$\begin{aligned} L(p_1 \to p_0) &= L_e(p_1 \to p_0) + \int_A f(p_2 \to p_1 \to p_0) L(p_2 \to p_1) \underbrace{V(p_2 \leftrightarrow p_1) \left|\cos\theta'\right| \frac{\left|\cos\theta''\right|}{\|p_1 - p_2\|}}_{G(p_1 \leftrightarrow p_2)} \mathrm{d}A(p_2) \\ &= L_e(p_1 \to p_0) + \int_A f(p_2 \to p_1 \to p_0) L(p_2 \to p_1) G(p_1 \leftrightarrow p_2) \mathrm{d}A(p_2) \end{aligned} \tag{2.8}$$

where $\mathrm{d}\omega_i = \frac{\left|\cos\theta''\right|}{\|p_1 - p_2\|} \mathrm{d}A(p_2)$ account for quantity change. An additional geometric term $G(p_1 \leftrightarrow p_2)$ is defined to help us write the integral into a more compact form.

Now if we keep expanding $L(p_2 \to p_1)$ further into its integral form, we obtained

$$\begin{aligned}
L(p_1 \to p_0) &= L_e(p_1 \to p_0) + \int_A f(p_2 \to p_1 \to p_0) L(p_2 \to p_1) G(p_1 \leftrightarrow p_2) \mathrm{d}A(p_2) \\
&= L_e(p_1 \to p_0) + \int_A f(p_2 \to p_1 \to p_0) L_e(p_2 \to p_1) G(p_1 \leftrightarrow p_2) \mathrm{d}A(p_2) \\
&+ \int_A \int_A f(p_2 \to p_1 \to p_0) f(p_3 \to p_2 \to p_1) L_e(p_3 \to p_2) G(p_1 \leftrightarrow p_2) G(p_2 \leftrightarrow p_3) \mathrm{d}A(p_2) \mathrm{d}A(p_3) \\
&+ ... \\
&= \sum_{k=0}^{\infty} \underbrace{\int_A ... \int_A}_{k \text{ times}} L_e(p_{k+1} \to p_k) \underbrace{\prod_{n=1}^{k} f(p_{n+1} \to p_n \to p_{n-1}) G(p_n \to p_{n+1})}_{T(\bar{p}_k)} \mathrm{d}A(p_2) ... \mathrm{d}A(p_{k+1}) \\
&= \sum_{k=0}^{\infty} \underbrace{\int_A ... \int_A}_{k \text{ times}} L_e(p_{k+1} \to p_k) T(\bar{p}_k) \mathrm{d}A(p_2) ... \mathrm{d}A(p_{k+1})
\end{aligned} \tag{2.9}$$

where $T(\bar{p}_k)$ is called the throughput of the path with length $k+1$. Finally, to compute the pixel color for pixel i, we have to integrate $L(p_1 \to p_0)$ over p_0, p_1, where p_0 on the camera lens, weighted by the pixel contribution term $W_e^{(i)}(p_0 \to p_1)$ for pixel i. Thus

$$I_i = \int_{A_{\text{film}}} \int_A W_e^{(i)}(p_0 \to p_1) L(p_1 \to p_0) G(p_0 \leftrightarrow p_1) \mathrm{d}A(p_1) \mathrm{d}A(p_0) \tag{2.10}$$

2.5 Monte-Carlo Integration

Although our problem has already been defined as an integration problem in Equation (2.10), solving such integral analytically is almost always impossible, except for some special cases. Fortunately, we could apply the Monte-Carlo integration technique, by drawing samples randomly and progressively computing the averages. Formally, given the integration problem

$$I_i = \int_{\mathcal{X}_i} f_i(x) \mathrm{d}x \tag{2.11}$$

We can draw K uniform samples $x_1, ...x_k$ and compute the average:

$$I_i \approx \langle I_i \rangle_{US} = \frac{1}{K} \sum_{k=1}^{K} f_i(x_k) \tag{2.12}$$

This forms a Monte-Carlo estimator with uniform sampling $\langle I_i \rangle_{US}$, If we compute the expected value of such estimator:

$$\mathbf{E}\left(\frac{1}{K}\sum_{k=1}^{K} f_i(x_k)\right) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{E}\left(f_i(x_k)\right)) = \mathbf{E}\left(f_i(x_k)\right) = \int_{\mathcal{X}_i} f_i(x)\cdot 1 dx = I_i \tag{2.13}$$

This shows that our estimator is unbiased.

2.5.1 Importance Sampling

However, a uniform sampling technique might not be optimal. For example, a function with only a small peak may have a hard time converging because the value at the peak may never be sampled. A concrete example will be to render a scene with a small light source: It will be extremely hard to sample a path to hit the light. It is more reasonable to draw samples from a distribution that focuses on the important part of the function, for example, sampling the lights. Thus we have an importance-sampled Monte-Carlo estimator:

$$\langle I_i \rangle_{IS} = \frac{1}{K}\sum_{k=1}^{K}\frac{f_i(x_k)}{p_i(x_k)} \tag{2.14}$$

Similar to Equation (2.13), this estimator is unbiased as well because

$$\mathbf{E}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{f_i(x_k)}{p_i(x_k)}\right) = \mathbf{E}\left(\frac{f_i(x_k)}{p_i(x_k)}\right) = \int_{\mathcal{X}_i}\frac{f_i(x_k)}{p_i(x_k)}\cdot p_i(x_k)dx = I_i \tag{2.15}$$

The variance of Equation (2.14) is as follows:

$$\mathbf{Var}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{f_i(x_k)}{p_i(x_k)}\right) = \frac{1}{K}\mathbf{Var}\left(\frac{f_i(x_k)}{p_i(x_k)}\right) = \frac{1}{K}\left(\int_{\mathcal{X}_i}\frac{f_i(x_k)^2}{p_i(x_k)}dx - I_i^2\right). \tag{2.16}$$

Consider the extreme case where $p_i(x_k)$ is very close to the normalized version of $f_i(x_k)^2$, such that $\frac{f_i(x_k)}{p_i(x_k)} \approx I_i$, then the variance in Equation (2.16) will be close to zero. However, such distribution is generally hard to obtain. It is usually helpful to select a particular part of the integrand that is easy to sample, thus we can have at least better samples over that specific part.

Regarding path tracing, path samples are constructed by recursively tracing randomly sampled rays at the hit vertex. Since Equation (2.4) is evaluated recursively at each vertex,

we can use the importance sampling technique at each vertex. If we pay closer attention to the integral of Equation (2.4)

$$\int_{\Omega} \underbrace{L_i(x, \omega_i)}_{\text{incoming light}} \underbrace{f_s(x, \omega_o, \omega_i) \left|\mathbf{n} \cdot \omega_i\right|}_{\text{cosine-weighted BSDF}} \mathrm{d}\omega_i \tag{2.17}$$

The integrand consists of the product of the cosine-weighted BSDF and the incoming light.

We can easily extract important samples from a distribution proportional to the cosine-weighted BSDF term.

For the lighting term, it is almost impossible to derive a distribution proportional to L_i as it must be recursively evaluated with Monte Carlo path tracing. However, we can evaluate the incoming light as a summation of direct light and indirect light:

$$L_i(x, \omega_i) = L_{\text{direct}}(x, \omega_i) + L_{\text{indirect}}(x, \omega_i) \tag{2.18}$$

where the indirect light is the part where we need to evaluate recursively. The direct term is simply the emission of the second hit by tracing rays from x towards direction ω_i. Thus we can sample the light source according to some heuristic and check for visibility by ray tracing. Although, unlike BSDF sampling, obtaining a perfect distribution for direct light sampling is not easy as it depends on both the configuration of lights and the hit point x, light sampling works particularly well for scenes with a small light source as BSDF sampling in this case would need to be very lucky for it to hit the light source.

2.5.2 Multiple Importance Sampling

Now we have at least two available sampling strategies, i.e. BSDF sampling and light sampling, each having its benefit to sample only a part of the integrand $f_i(x)$. The question is, can we combine different sampling strategies and form a better estimator?

We can use *multiple importance sampling*, or MIS [26] to combine samples and form a better estimator. Specifically, given different importance sampling strategies, $p_1, ...p_M$, if we draw K_m samples from p_m, the MIS estimator is:

$$I_i = \sum_{m=1}^{M} \int_{\mathcal{X}_i} w_m(x) f_i(x) \mathrm{d}x \approx \sum_{m=1}^{M} \frac{1}{K_m} \sum_{k=1}^{K_m} w_m(x_k^m) \frac{f_i(x_k^m)}{p_m(x_k^m)} = \langle I_i \rangle_{MIS}, \tag{2.19}$$

where x_k^m is the k'th sample drawn from technique m, $w_m(x_k^m)$ is the MIS weight, such that

$$\begin{cases} \sum_{m=1}^{M} w_m(x) = 1 & \text{when } f_i(x) \neq 0 \\ w_m(x) = 0 & \text{when } p_i(x) = 0 \end{cases} \tag{2.20}$$

For the choice of w_m, Veach[26] has proven that no other heuristic can be significantly better than the balance heuristic:

$$w_m(x) = \frac{K_m p_m(x)}{\sum_{m'=1}^{M} K_{m'} p_{m'}(x)} \tag{2.21}$$

2.6 Gradient-Domain Rendering

The Monte-Carlo method provides a tool for us to estimate I_i in Equation (2.10) for individual pixel i. However, it is reasonable to assume that nearby pixels have strong correlations. If samples from one pixel can somehow help estimate the color in another pixel, it would bring a lot of benefits for such correlation to propagate. Gradient-domain rendering realizes this correlation can be interpreted as constructing a difference estimate between nearby pixels using correlated samples. The difference estimators constructed thus hopefully have far lower variance than primal pixel intensity estimates. Taking these extra lower variance estimates into account, a lower variance image can be reconstructed. In this section, we go over the details of gradient-domain path tracing [18], which is the baseline and the fundamental building block of our method later introduced in Chapter 4 and Section 4.3. We start by providing an overview of the gradient-domain rendering algorithm, specifically a baseline gradient-domain path tracing (G-PT) algorithm. We then provide the mathematical details for the shift mapping process, which is used to obtain correlated samples between different pixels. In the end, we briefly discuss why gradient-domain rendering helps in reducing variance when the smoothness assumption holds.

2.6.1 Overview

The gradient-domain rendering algorithm takes the Monte-Carlo estimator

$$I_i \approx \langle I_i \rangle_{MC} \tag{2.22}$$

for every pixel i. We name this the *primal* estimate for pixel i.

In addition, the algorithm takes the *difference* estimates between pixels i, j:

$$I_j - I_i \approx \langle \Delta_{ij} I \rangle_{MC} . \tag{2.23}$$

Such difference estimates often can be obtained with variance far lower than the primal estimate. The detail of how this estimator is designed will be shown in Section 2.6.2.

To obtain the pixel color with these two different kind of estimators combined, gradient-domain rendering uses Poisson reconstruction to reconstruct all I_i based on the primal estimates $\langle I_i \rangle_{MC}$ and the difference estimates $\langle \Delta_{ij} I \rangle$ by solving a minimization problem. With L^2 loss, the reconstruction target can be expressed as

$$\langle I \rangle_G = \min_I \sum_{i,j} \| I_j - I_i - \langle \Delta_{ij} I \rangle_{MC} \|^2 + \lambda \sum_i \| I_i - \langle I_i \rangle_{MC} \|^2. \tag{2.24}$$

The first term aims to minimize the gap between the reconstructed difference and the difference estimate. The second term with the parameter λ regularizes the reconstruction towards the primal per-pixel Monte Carlo estimates for each I_i.

2.6.2 Shift Mapping

Given a sampled path $\bar{\mathbf{x}}_i$ at pixel i, how can we obtain a correlated path at pixel j? Intuitively, we want to have a similar path, with a similar contribution to pixel j as $\bar{\mathbf{x}}_i$ to pixel i. Techniques used to construct such correlated paths are called *shift mapping*, or *path mapping*. We define such mapping as a function T_{ij}, such that

$$f_j(T_{ij}(\bar{\mathbf{x}}_i)) \approx f_i(\bar{\mathbf{x}}_i) \tag{2.25}$$

There are many different shift mapping techniques, we explain two types of shift mapping used in our implementation: random sequence replay and vertex reconnection.

Random Sequence Replay

Random sequence replay gives an almost trivial technique for obtaining correlated samples. However, random replay usually does not work well for surfaces with geometry or normal changes, as illustrated in Figure 2.3. The path $\bar{\mathrm{x}}_i$ is sampled from a sequence of random numbers $\bar{\mathrm{u}}_i$ sampled between $[0, 1]$. For simplicity we denote $\bar{\mathrm{x}}_i(\bar{\mathrm{u}}_i)$ to be the path sampled at pixel i with random sequence $\bar{\mathrm{u}}_i$ using sampling technique p_i, and $\bar{\mathrm{u}}_i(\bar{\mathrm{x}}_i)$ to be the random sequence which has to be used to sample path $\bar{\mathrm{x}}_i$ at pixel i with the same sampling technique p_i.

Random sequence replay simply copies this sequence into the neighboring pixel and samples according to these random numbers. The shift function is

$$T_{ij}(\bar{\mathrm{x}}_i) = \bar{\mathrm{x}}_j(\bar{\mathrm{u}}_i(\bar{\mathrm{x}}_i)) \tag{2.26}$$

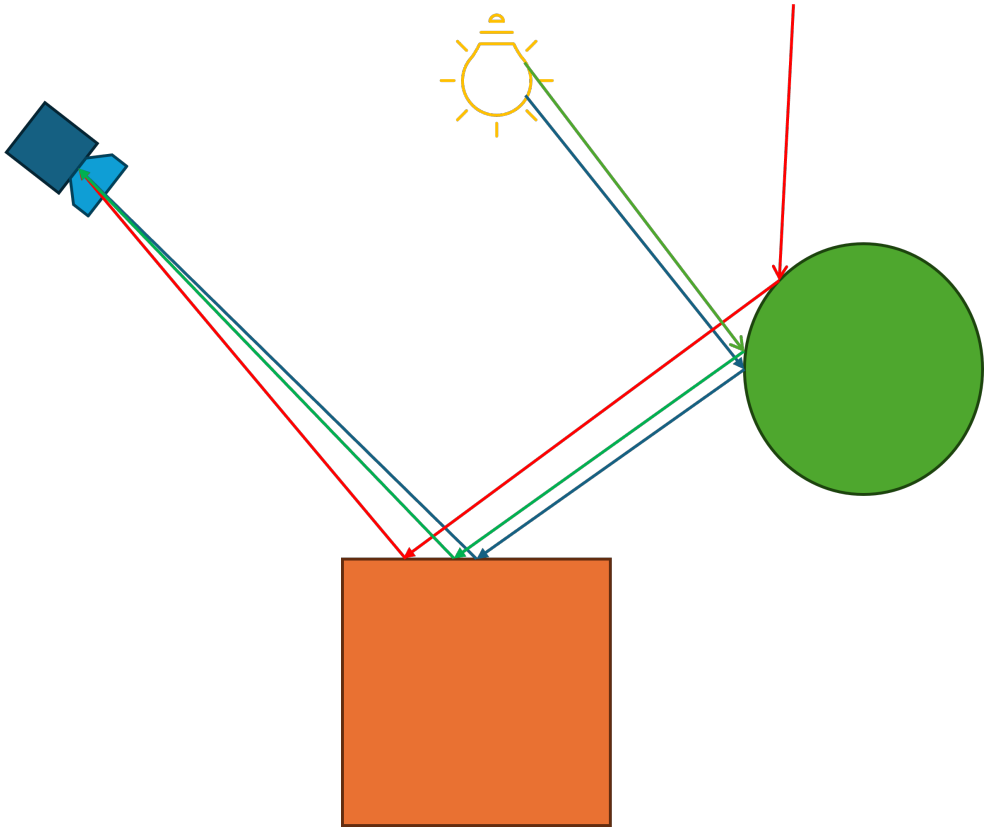


Figure 2.3: *Random sequence replay: The base path shifted (blue) to different pixels. Note that even with simple geometry changes the path (red) can become very far away from the base path.*

It is also a requirement that we compute the Jacobian of the shift mapping:

$$\left|\frac{\mathrm{d}T_{ij}(\bar{\mathrm{x}}_i)}{\mathrm{d}\bar{\mathrm{x}}_i}\right| = \left|\frac{\mathrm{d}\bar{\mathrm{x}}_j}{\mathrm{d}\bar{\mathrm{u}}_i}\right| \left|\frac{\mathrm{d}\bar{\mathrm{u}}_i}{\mathrm{d}\bar{\mathrm{x}}_i}\right| = \frac{p_i(\bar{\mathrm{x}}_i)}{p_j(\bar{\mathrm{x}}_j)} \tag{2.27}$$

Vertex Reconnection

As in Figure 2.4, vertex connection simply connects the shifted prefix path of the current pixel with the suffix path of the neighboring path. Specifically, given a target path

$$\bar{\mathrm{x}}_i = x_i^0 x_i^1 x_i^2 ... x_i^{n-1} x_i^n, \tag{2.28}$$

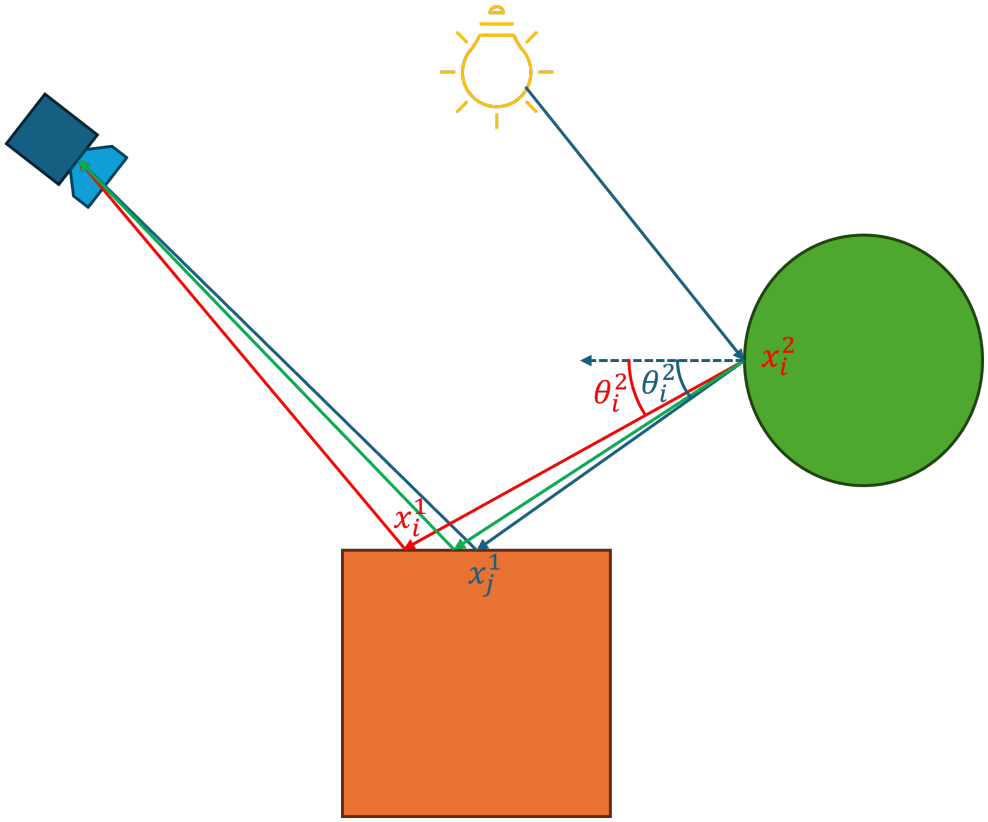


Figure 2.4: *Vertex reconnection: the second vertex of every shifted path is connected to the second vertex of the base path. This forms geometrically more correlated paths but may lead to low radiance if the surface type of the first vertex or the second vertex (or both) is glossy.*

a shifted prefix path with shift function T'_{ij} is

$$x_j^0 ... x_j^k = T'_{ij}(x_i^0 ... x_i^k). \tag{2.29}$$

Vertex reconnection simply reconnects vertex x_j^k with x_i^{k+1} and forms a new path

$$\bar{\mathrm{x}}_j = x_j^0 ... x_j^k x_i^{k+1} .... x_i^n. \tag{2.30}$$

The Jacobian can be computed by chaining the Jacobian of T'_{ij} and the Jacobian of the shift afterward. The Jacobian of the suffix path $x_j^{k+1} ... x_j^n = x_i^{k+1} ... x_i^n$ is simply 1. The

Jacobian of the reconnection $x_j^k x_i^{k+1}$ simply consists the ratio of the geometry terms:

$$\frac{\left|\cos\theta_j^{k+1}\right| \left\|x_i^k - x_i^{k+1}\right\|^2}{\left\|x_j^k - x_i^{k+1}\right\|^2 \left|\cos\theta_i^{k+1}\right|}, \tag{2.31}$$

where θ_i^{k+1} is the angle between the surface normal of x_i^{k+1} and the normalized direction of $\overrightarrow{x_i^k x_i^{k+1}}$, and θ_j^{k+1} is the angle between the surface normal of x_i^{k+1} and the normalized direction of $\overrightarrow{x_j^k x_i^{k+1}}$.

Mixed strategies

Random replay preserves the specular chain but the shifted path can quickly become far away from the original path if geometry or normal changes between paths even by just a small amount. Vertex reconnection on the other hand makes sure the path reconnects and as a consequence, all subsequent vertices remain the same as the base path but can lead to a low contribution path when encountering a glossy or specular surface. As paths are usually constructed sequentially, a common practice is to first adapt random replay and only use vertex reconnection when we find two consecutive diffuse vertices. This strategy makes sure that the specular chain is preserved until the point where path similarities become more important.

2.6.3 Difference Estimator

It seems obvious that we could simply take the difference of the contribution of the two correlated paths as our difference sample, by drawing multiple samples followed by averaging and thereby achieve a low variance difference estimate. However, this can unfortunately lead to incorrect results. To see that, we first write out the actual difference of the pixels:

$$I_j - I_i = \int_{\mathcal{X}_j} f_j(x_j)\mathrm{d}x_j - \int_{\mathcal{X}_i} f_i(x_i)\mathrm{d}x_i. \tag{2.32}$$

We can immediately see the first problem we have is that the integration domain is slightly different between two pixels. If the mapping T_{ij} is not surjective, meaning that some part of domain $\mathcal{X}_j$ is not covered by the integration, bias will arise. Secondly, if the mapping is not reversible (injective), then some elements in domain $\mathcal{X}_j$ will be counted twice, which will also lead to bias in our estimator. In the design of shift mapping, reversibility can

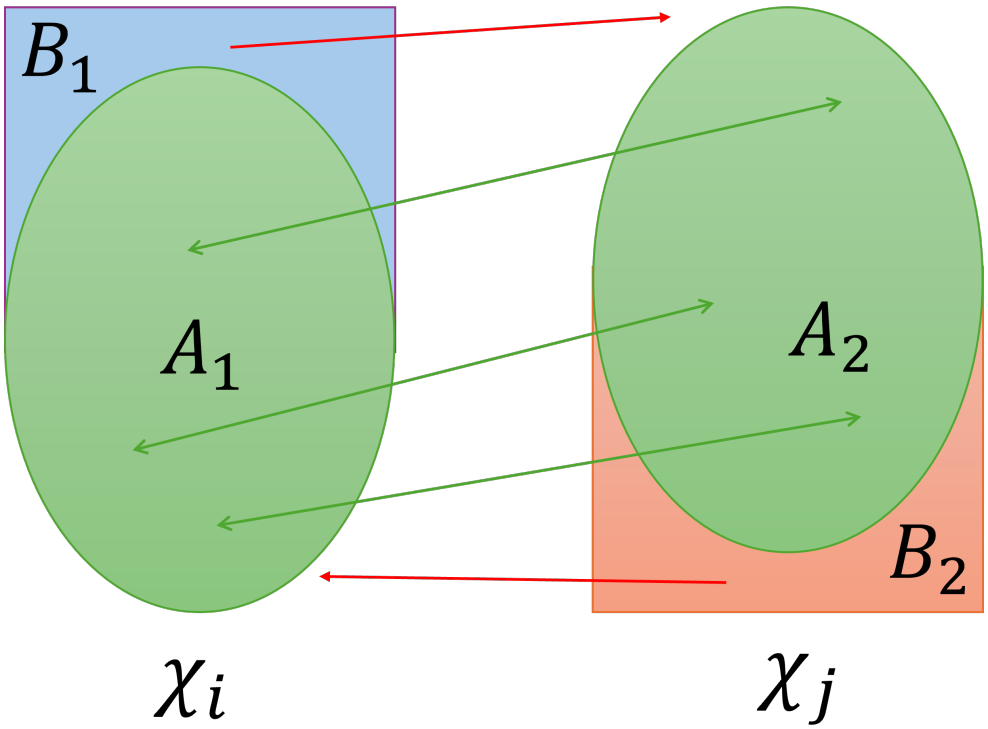


Figure 2.5: *Shift mapping between different integration domains*

be checked easily, but then in the case when shift mapping fails, how can we handle such failure without losing a part of the integration domain $\mathcal{X}_i$?

This problem can be seen more clearly in Figure 2.5. We can see one simple solution is to modify the difference integrals into multiple parts:

$$\begin{aligned} I_j - I_i &= \int_{\mathcal{X}_j} f_j(x_j)\mathrm{d}x_j - \int_{\mathcal{X}_i} f_i(x_i)\mathrm{d}x_i \\ &= \int_{A_1} f_j(x)\mathrm{d}x - \int_{A_2} f_i(x)\mathrm{d}x + \int_{B_2} f_j(x)\mathrm{d}x - \int_{B_1} f_i(x)\mathrm{d}x \\ &= \int_{A_1} f_j(T_{ij}(x))\left|\frac{\mathrm{d}T_{ij}(x)}{\mathrm{d}x}\right| - f_i(x)\mathrm{d}x + \int_{B_2} f_j(x)\mathrm{d}x - \int_{B_1} f_i(x)\mathrm{d}x \\ &\equiv \int_{A_2} f_j(x) - f_i(T_{ji}(x))\left|\frac{\mathrm{d}T_{ji}(x)}{\mathrm{d}x}\right|\mathrm{d}x + \int_{B_2} f_j(x)\mathrm{d}x - \int_{B_1} f_i(x)\mathrm{d}x \end{aligned} . \tag{2.33}$$

While this simple solution has provided a correct formulation for the difference estimate, we have to choose to either integrate over domain A_1 or A_2. However, these two different

formulations are essentially just an MIS formulation with different weighting schemes! For example, the third line of Equation (2.33) essentially sets the MIS weight to 1 when we are sampling domain A_1 and 0 if we are sampling from domain A_2. This shows that we can use the MIS theory in Section 2.5.2 we explained earlier.

In fact, [10] [14] shows that we can write Equation (2.33) in more compact form by simply working with the original domain $\mathcal{X}_i$ and $\mathcal{X}_j$ when applying MIS. Consider that we are just trying to sample the difference from domain $\mathcal{X}_i$ and the difference from domain $\mathcal{X}_j$ with shift mapping T_{ij} and T_{ji} respectively. Since the two sampling strategies cover the entire domain $\mathcal{X}_i \bigcup \mathcal{X}_j$, the condition of applying MIS is satisfied. We can now write the difference as:

$$\begin{aligned}\Delta_{ij} I = &\int_{\mathcal{X}_i} w_{ij}(x) \left(f_j(T_{ij}(x)) \left| \frac{\mathrm{d}T_{ij}(x)}{\mathrm{d}x} \right| - f_i(x) \right) \mathrm{d}x \\ &+ \int_{\mathcal{X}_j} w_{ji}(x) \left(f_j(x) - f_i(T_{ji}(x)) \left| \frac{\mathrm{d}T_{ji}(x)}{\mathrm{d}x} \right| \right) \mathrm{d}x\end{aligned} \tag{2.34}$$

where $\left| \frac{\mathrm{d}T_{ij}}{\mathrm{d}x}(x) \right|$ denote the Jacobian determinant for T_{ij} with respect to x. w_{ij}, w_{ji} are the MIS weights satisfying conditions as mentioned in Section 2.5.2.

One can obtain unbiased estimates by sampling both integrals according to their respective sampling distributions p_i, p_j

$$\begin{aligned}\langle \Delta_{ij} I \rangle_{MC} = &\frac{1}{N} \sum_{k=1}^{N} w_{ij}(x_{i,k}) \frac{f_j(T_{ij}(x_{i,k})) \left| \frac{\mathrm{d}T_{ij}(x_{i,k})}{\mathrm{d}x_{i,k}} \right| - f_i(x_{i,k})}{p_i(x_{i,k})} \\ &+ \frac{1}{N} \sum_{k=1}^{N} w_{ji}(x_{j,k}) \frac{f_j(x_{j,k}) - f_i(T_{ji}(x_{j,k})) \left| \frac{\mathrm{d}T_{ji}(x_{j,k})}{\mathrm{d}x_{j,k}} \right|}{p_j(x_{j,k})}\end{aligned} \tag{2.35}$$

where N denotes the number of samples. As indicated as red arrows in Figure 2.5, when reversibility check failed for strategy p_i at sample x_i, we consider in this case that $f_j(T_{ij}(x_i))$ produce zero contribution and $p_j(T_{ij}(x_i)) = 0$ (as $T_{ij}(x_i)$ is considered as a invalid path). the same rule also applies in reverse. Balance heuristic explained in Section 2.5.2 can be used for the MIS weight, but one has to be careful with quantity change:

$$w_{ij}(x_i) = \frac{p_i(x_i)}{p_i(x_i) + p_j(T_{ij}(x_i)) \left| \frac{\mathrm{d}T_{ij}(x_i)}{\mathrm{d}x_i} \right|} \tag{2.36}$$

For now, we have completed the explanation for gradient-domain rendering.

2.6.4 Analysis

To understand our motivation later in this thesis, we provide a brief analysis of why gradient-domain rendering works under the smoothness assumption.

Shift mapping assumes that the integrand f_i is similar to a neighboring integrand f_j for similar x, and this assumption can be informally written as

$$x_i \approx x_j \Rightarrow f_i(x_i) \approx f_j(x_j). \tag{2.37}$$

where $x_j = T_{ij}(x_i)$. The difference of the sample contributions $f_i(x_i)$ and $f_j(x_j)$ will be small when shift mapping results in $x_i \approx x_j$ and if the above assumption is satisfied.

When the assumption in Equation (2.37) holds and shift mapping results in $x_{i,k} \approx T_{ij}(x_{i,k})$, $x_{j,k} \approx T_{ji}(x_{j,k})$, this difference estimator $\langle \Delta_{ij} I \rangle$ will have low variance because

$$\begin{aligned} \left| \frac{\mathrm{d}T_{ij}(x_{i,k})}{\mathrm{d}x_{i,k}} \right| \approx 1 \quad \text{and} \quad \left| \frac{\mathrm{d}T_{ji}(x_{j,k})}{\mathrm{d}x_{j,k}} \right| \approx 1 \\ f_j(T_{ij}(x_{i,k})) \left| \frac{\mathrm{d}T_{ij}(x_{i,k})}{\mathrm{d}x_{i,k}} \right| - f_i(x_{i,k}) \approx 0. \end{aligned} \tag{2.38}$$

Chapter 3

Problem Statement

While gradient-domain rendering works well when Equation (2.37) holds, this assumption may not hold universally, thereby constraining its application in practice. For example, it is common to have a spatial variation of albedo and/or normals that may result in different integrands per pixel. Figure 3.1 shows a simple example where f_i varies like a checkerboard pattern over i. In this example, f_i is a product of two functions

$$f_i(x) = g_i(x)h_i(x) \tag{3.1}$$

where $g_i(x)$ varies like a checkerboard (i.e., differs among pixels around edges) but $h_i(x) = x$ is identical for all i (i.e., no variation among pixels). This example models a typical scenario in rendering where $g_i(x)$ is a spatially varying feature such as the BSDF term per pixel and $h_i(x)$ is a mostly smooth feature such as incident illumination. The integrand $f_i(x)$ does not satisfy the assumption in Equation (2.37) whenever pixels I_i and I_j have different checkerboard colors, and only a part of the integrand $h_i(x)$ satisfies the assumption. As such, gradient-domain rendering fails to handle even this simple example and results in large errors whenever the integrand $f_i(x)$ varies among neighboring pixels. We present an extension of gradient-domain rendering that can efficiently handle a larger set of problems including this simple example.

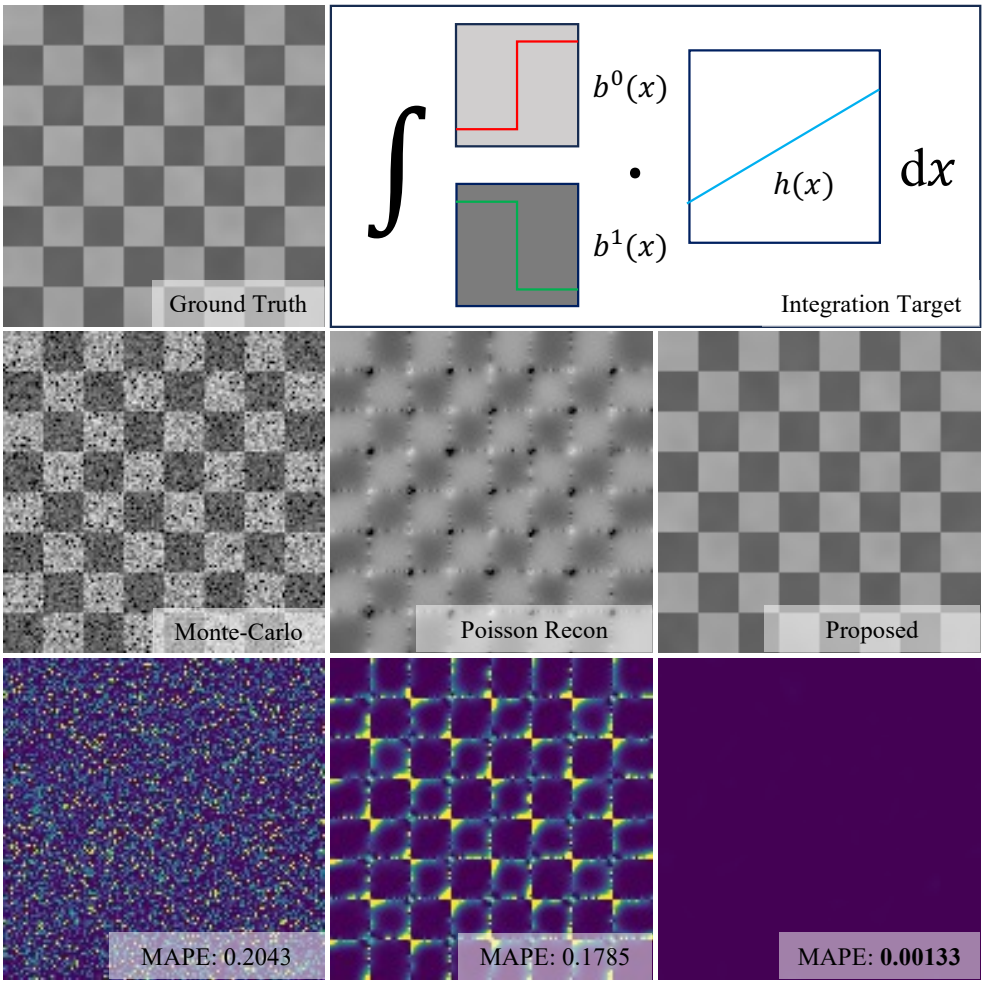


Figure 3.1: *Simple failure mode of gradient-domain rendering. Top right shows the integration target for lighter and darker blocks of the checkerboard pattern, middle and bottom row shows the reconstruction using per-pixel Monte Carlo integration, regular gradient-domain rendering, and our method and their respective RMSE compared with ground truth. Regular gradient-domain rendering suffers from artifacts due to high variance difference estimates around edges. Our method almost perfectly reconstructs the ground truth.*

Chapter 4

Method

To introduce our idea, let us work through the example in Figure 3.1. Suppose that we represent $g_i(x)$ as a weighted sum of two *basis* functions $b_i^0(x)$ and $b_i^1(x)$ as in

$$g_i(x) \stackrel{\text{def}}{=} \alpha_i^0 b_i^0(x) + \alpha_i^1 b_i^1(x) \tag{4.1}$$

where α_i^0 and α_i^1 are the weights. Let us choose the basis functions $b_i^0(x)$ and $b_i^1(x)$ such that each one of them is equal to $g_i(x)$ within the corresponding checkerboard block. The weights α_i^0 and α_i^1 in this case can be trivially defined as

$$\alpha_i^0 = \begin{cases} 0 & (i \in \text{darker block}) \\ 1 & (i \in \text{lighter block}) \end{cases} \quad \alpha_i^1 = 1 - \alpha_i^0 \tag{4.2}$$

which effectively captures the checkerboard pattern via the weights. Note that $b_i^l(x) \stackrel{\text{def}}{=} b^l(x)$ does not vary over i, and the per-pixel variation is now encoded by the weights α_i^0 and α_i^1.

Under this basis expansion of $g_i(x)$, the integral I_i could also be decomposed into a weighted sum:

$$\begin{aligned} I_i &= \int_{\mathcal{X}_i} h_i(x) g_i(x) \mathrm{d}x = \int_{\mathcal{X}_i} h_i(x) \left(\alpha_i^0 b_i^0(x) + \alpha_i^1 b_i^1(x) \right) \mathrm{d}x \\ &= \alpha_i^0 \int_{\mathcal{X}_i} h_i(x) b_i^0(x) \mathrm{d}x + \alpha_i^1 \int_{\mathcal{X}_i} h_i(x) b_i^1(x) \mathrm{d}x = \alpha_i^0 I_i^0 + \alpha_i^1 I_i^1 \end{aligned} \tag{4.3}$$

where we wrote $I_i^0 = \int_{\mathcal{X}_i} h_i(x) b_i^0(x) \mathrm{d}x$ and similarly for I_i^1. The problem changed from solving *one* integral of $h_i(x) g_i(x)$ to *two* integrals for $h_i(x) b_i^0(x)$ and $h_i(x) b_i^1(x)$ weighted

by α_i^0 and α_i^1. The new integrands $h_i(x)b_i^0(x)$ and $h_i(x)b_i^1(x)$ both satisfy the similarity assumption in Equation (2.37) because $b_i^0(x)$, $b_i^1(x)$, and $h_i(x)$ are in fact *exactly the same* for all the pixels i in this particular example. Therefore, gradient-domain rendering can be successfully applied for each of $h_i(x)b_i^0(x)$ and $h_i(x)b_i^1(x)$ independently, and we can reconstruct I_i^0 and I_i^1 to estimate I_i as the weighted sum $\alpha_i^0 I_i^0 + \alpha_i^1 I_i^1$ per pixel. We generalize this process as follows.

Step 1: Decompose $g_i(x)$ into a weighted sum of basis functions that satisfy the assumption, i.e. $b_i^l(x_a) \approx b_j^l(x_b)$ for $x_a \approx x_b$:

$$g_i(x) \stackrel{\text{def}}{=} \sum_{l=1}^{L} \alpha_i^l b_i^l(x). \tag{4.4}$$

Note that the basis functions do not need to be the same functions for all the pixels as long as they satisfy the similarity assumption. There is also no assumption on the weights α_i^l across pixels.

Step 2: Write down the integral of $g_i(x)$ also as a weighted sum:

$$\begin{aligned} I_i &= \int_{\mathcal{X}_i} h_i(x) g_i(x) \mathrm{d}x = \int_{\mathcal{X}_i} h_i(x) \sum_{l=1}^{L} \alpha_i^l b_i^l(x) \mathrm{d}x \\ &= \sum_{l}^{L} \alpha_i^l \underbrace{\int_{\mathcal{X}_i} h_i(x) b_i^l(x) \mathrm{d}x}_{I_i^l} = \sum_{l=1}^{L} \alpha_i^l I_i^l. \end{aligned} \tag{4.5}$$

Step 3: Apply gradient-domain rendering on new integrands $b_i^l(x)h_i(x)$ independently, reconstruct I_i^l, and take the weighted sum to have $I_i = \sum_{l=1}^{L} \alpha_i^l I_i^l$. For problems where $h_i(x)$ satisfies the similarity assumption, the integrand $b_i^l(x)h_i(x)$ should also satisfy the assumption.

For the example in Figure 3.1, in vanilla gradient-domain rendering, the difference estimate between pixels sharing the same color exhibits zero variance, i.e., $b^0(x)h(x) - b^0(x)h(x) = b^1(x)h(x) - b^1(x)h(x) = 0 = \textit{constant}$ for arbitrary x, conversely, the gradient estimate between pixels with different colors entails non-zero variance, i.e., $\pm(b^1(x)h(x) - b^0(x)h(x)) \neq \textit{constant}$. Consequently, high-variance (error) information propagates to other pixels at the reconstruction step. This error manifests as "dipole artifacts" in Figure 3.1 under L^2 reconstruction. The difference image shows significant errors around the boundaries between different colors.

To summarize why basis expansion helps in this case, let us compare the difference between our formulation and vanilla gradient-domain. Assuming that $h_i(x) \approx h_j(x)$, $g_i(x) = \sum_l \alpha_i^l b_i^l(x)$, $g_j(x) = \sum_l \alpha_j^l b_j^l(x)$ and $b_i^l(x) \approx b_j^l(x)$ for all l:

- In gradient-domain rendering, the difference estimate used is

$$\begin{aligned}\Delta_{ij} I &= \int_{\mathcal{X}} f_i(x) - f_j(x) \mathrm{d}x = \int_{\mathcal{X}} h_i(x) g_i(x) - h_j(x) g_j(x) \mathrm{d}x \\ &\approx \int_{\mathcal{X}} h_i(x) \left(g_i(x) - g_j(x)\right) \mathrm{d}x\end{aligned}$$

  which may have large variance when $g_i(x) \neq g_j(x)$.

- In our formulation, the difference estimate used is

$$\begin{aligned}\Delta_{ij} I^l &= \int_{\mathcal{X}} h_i(x) b_i^l(x) - h_j(x) b_j^l(x) \mathrm{d}x \\ &\approx \int_{\mathcal{X}} h_i(x) \left(b_i^l(x) - b_j^l(x)\right) \mathrm{d}x \approx 0 \\ &\qquad\qquad \text{(because } h_i(x) \approx h_j(x)\text{)}\end{aligned}$$

  which results in more cancellations.

4.1 Basis Expansion

Given a function g and a set of basis functions $\{b^l(x)\}_l$, basis expansion approximates g as a linear combination of b^l as

$$g(x) \approx \sum_{l=1}^{L} \alpha^l b^l(x) \tag{4.6}$$

where L is the number of basis functions of our choice and α^l defines the weight (or coefficients) for corresponding b^l. For orthonormal bases, each coefficient is simply a projection of function g onto the corresponding basis as

$$\alpha^l = \int_{\mathcal{X}} g(x) b^l(x) \mathrm{d}x \approx \sum_{k}^{K} \frac{g(x_k) b^l(x_k)}{p(x_k)}. \tag{4.7}$$

Basis expansion has been used in various applications in rendering. Sloan et al. [24], for example, proposed to use a precomputed set of coefficients to represent light transport

under an environment map. Ng et al. [21] broke down the rendering equation for direct illumination into a triple product integral comprising the environment map, visibility, and cosine-weighted BSDF terms. This decomposition is akin to our decomposition of $g_i(x)$ and the resulting decomposition of the integral. A recent extension of this approach by Xu et al. [30] uses a neural network to train a more sophisticated set of basis functions for better approximation of each component.

Manzi et al. [20] also proposed incorporating basis expansion into gradient-domain rendering. Still, the contexts of basis expansion fundamentally differ between our method and those of Manzi et al. Our method defines basis functions to decompose the integrand and provide low-variance difference estimates. This process involves changes in both difference estimates and reconstruction. Manzi et al. consider a weighted sum of basis features (vectors) to regularize reconstruction only, which does not involve any change in difference estimates.

While those prior works are relevant in that they also use basis expansion, the contexts of how basis expansion helps are different and thus only remotely related to our work. For example, having higher-order basis expansion does not always help in our case because high-frequency signals tend to violate the similarity assumption as in Equation (2.37) when shift mapping slightly changes the sample location. In addition, unlike those prior uses of basis expansion, we would like to have an *exact* recovery of $g_i(x)$ for each pixel for consistent reconstruction. The example in Figure 3.1 is a case where the exact recovery is trivially available, but we would like to achieve this for more general cases. We thus propose slight modifications to this traditional basis expansion to align with our specific requirements.

4.2 Lossless Basis Expansion.

We need to choose $\{b_i^l\}$ such that the weighted sum perfectly recovers $g_i(x)$ while also satisfying the similarity assumption. Observing that approximation by basis expansion (e.g., spherical harmonics) effectively captures the general shape of g_i while simultaneously satisfying the similarity assumption (i.e., spherical harmonics are the same for all the pixels), we would like to still utilize such existing basis functions. To achieve this goal, we consider adding back the *residual* (i.e., approximation error) to the basis functions. Given a set of basis function $\{b^l(x)\}_l$ we define the residual as

$$r_i(x) = g_i(x) - \sum_{l=1}^{L} \alpha_i^l b^l(x). \tag{4.8}$$

To recover $g_i(x)$ exactly, we "distribute" $r_i(x)$ to all the given bases. We add $r_i(x)$ evenly based on either the absolute value or the squared value of the coefficient to each basis, which results in

$$\begin{aligned} g_i(x) &= \sum_{l=1}^{L} \alpha_i^l b^l(x) + r_i(x) = \sum_{l=1}^{L} \left(\alpha_i^l b^l(x) + \frac{\|\alpha_i^l\|^n}{\sum_{l'} \|\alpha_i^{l'}\|^n} r_i(x) \right) \\ &= \sum_{l=1}^{L} \alpha_i^l \left(b^l(x) + \frac{1}{\alpha_i^l} \frac{\|\alpha_i^l\|^n}{\sum_{l'} \|\alpha_i^{l'}\|^n} r_i(x) \right) \end{aligned} \tag{4.9}$$

where $n \in \{1, 2\}$. We can redefine our *lossless basis* for pixel i, namely $\{b_i^l(x)\}_l$, as

$$b_i^l(x) \stackrel{\text{def}}{=} b^l(x) + w_i^l r_i(x) \tag{4.10}$$

where $w_i^l = \frac{1}{\alpha_i^l} \frac{\|\alpha_i^l\|^n}{\sum_{l'} \|\alpha_i^{l'}\|^n}$. This process keeps the basis for every pixel as close as possible to the given original basis while still maintaining the equality constraint. We empirically did not observe a clear advantage of one weighting scheme over the other with $n = 1$ or $n = 2$. Consequently, in our implementation, we adhere to the $n = 2$, maintaining consistency for evaluation purposes. Figure 4.1 provides an overview of our method using lossless basis expansion.

This formulation is straightforward to implement with an arbitrary choice of the starting global basis. Now the difference of $b_i^l(x)$ between pixel i and j essentially depends on the residual, so selecting global bases that result in smaller residuals helps in general. We have not explored an optimal choice of such global basis functions and will leave it as future work. It is worth noting that using $g_i(x)$ as the basis itself, as well as the BSDF integration pre-factorization approach [32] are both special cases of our method. In the case of using $g_i(x)$, our method reduces to gradient-domain path tracing [18]. The case of BSDF integration pre-factorization approach [32] is equivalent to our method with a single constant global basis followed by per-pixel residual distribution, resulting in a unit cosine-weighted BSDF as a single lossless basis in our case. Our formulation thus generalizes those prior approaches. Figure 4.2 shows our method with different orders of spherical harmonics as global bases. General material variation, including albedo, can be effectively handled with more bases. It is worth iterating that a higher-order basis does not always mean better reconstruction in our method and the performance of higher-order bases is empirically comparable to lower-order ones in some scenes.

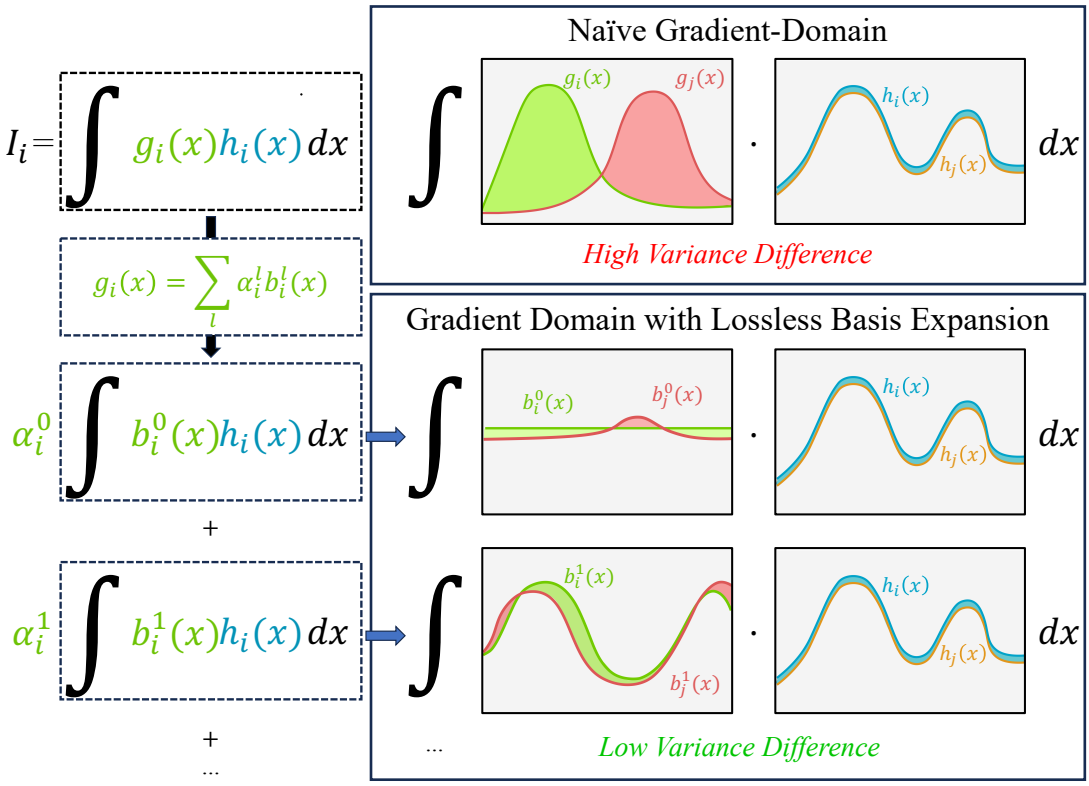


Figure 4.1: *Overview of our method. We propose to expand a part of the integrand $g(x)$ that significantly varies across pixels into a weighted sum of lossless basis functions. The resulting basis expansion is designed to satisfy the similarity assumption between neighboring pixels i, j, making difference estimates effective with shift mapping.*

4.3 Reconstruction

Once we decomposed g_i, we can estimate the difference estimator for $I_j^l - I_i^l \approx \langle \Delta_{ij} I^l \rangle$ as in Equation (2.35) using the same set of samples and the same shift mapping as in regular gradient-domain, except we replace $f_i(x)$ with $h_i(x)b_i^l(x)$. It is not necessary to have additional samples for multiple bases. We then reconstruct the contribution to the pixel coming from this basis integral I_i^l.

While directly reconstructing each I_i^l independently by following Equation (2.24) is straightforward, the results are rather undesirable because our end goal is to reconstruct I_i, not individual I_i^l. If the coefficient α_i^l of any particular basis l for pixel i is small, I_i^l is

not as important for our final estimate I_i because of Equation (4.5). As a result, we could give less weight to regularization related to I_i^l and allow the reconstruction to be more focused on other indices.

We introduce a new loss that takes the coefficients α into account. Under L^2 loss, our reconstruction is defined as

$$\begin{aligned}\left\{\left\langle I^l\right\rangle_{G'}\right\}_l = \min_{\{I^l\}_l} \sum_{i,j} \frac{1}{L} \sum_{l=1}^{L} \| I_j^l - I_i^l - \left\langle \Delta_{ij} I^l \right\rangle_{MC} \|^2 \\ + \lambda \sum_i \left\| \sum_{l=1}^{L} \alpha_i^l I_i^l - \langle I_i \rangle_{MC} \right\|^2 \end{aligned} . \tag{4.11}$$

where $I_i^l = \int_{\mathcal{X}_i} h_i(x) b_i^l(x) \mathrm{d}x$ is as defined in Equation (4.5). The subscript G' is to differentiate the estimator for I_i^l from the naive Poisson reconstruction. The first difference term is the same as in Equation (2.24) except now it also considers averaging the loss from all the bases. The second term, instead of regularizing each basis directly with the primal estimate, regularizes the reconstructed image with the primal image estimate. While another feasible option is to regularize the weighted basis with the weighted primal estimate, we use the current approach because it has only one λ parameter that empirically affects the results in the same manner as in regular gradient-domain rendering. The final integral estimator is simply $\langle I_i \rangle_{GB} = \sum_{l=1}^{L} \alpha_i^l \left\langle I_i^l \right\rangle_{G'}$. Note that the estimated $\left\{\left\langle I^l\right\rangle_{G'}\right\}_l$ are unbiased under L^2 because the expected values of $I_j^l - I_i^l - \left\langle \Delta_{ij} I^l \right\rangle_{MC}$ and $\sum_{l=1}^{L} \alpha_i^l I_i^l - \langle I_i \rangle_{MC}$ are both zeros by construction. Thus

$$\mathbf{E}\left[\langle I_i \rangle_{GB}\right] = \mathbf{E}\left[\sum_{l=1}^{L} \alpha_i^l \left\langle I_i^l \right\rangle_{G'}\right] = \sum_{l=1}^{L} \alpha_i^l \mathbf{E}\left[\left\langle I_i^l \right\rangle_{G'}\right] = \sum_{l=1}^{L} \alpha_i^l I_i^l = I_i \tag{4.12}$$

which implies that our estimator $\langle I_i \rangle_{GB}$ is unbiased under L^2 reconstruction.

4.4 Application to Rendering

We apply our framework to rendering based on the rendering equation [11] where we have

$$L_o(\mathbf{x}, \phi) = L_e(\mathbf{x}, \phi) + \int_\Omega \underbrace{L_i(\mathbf{x}, \omega)}_{h(\omega)} \underbrace{f_r(\mathbf{x}, \phi, \omega) |\mathbf{n} \cdot \omega|}_{g_\phi(\mathbf{x}, \omega)} \mathrm{d}\omega \tag{4.13}$$

where $\mathbf{x}$ is the position, ϕ, ω are the outgoing and incoming directions respectively. We assigned incident radiance L_i to h and the cosine weighted BRDF term $f_r(\mathbf{x}, \phi, \omega)|\mathbf{n} \cdot \omega|$ to g. This choice is justified by common scenarios where incident illumination varies smoothly (except for geometric discontinuities) and the BRDF term may vary rapidly over the image due to material variation. We ignore the emission term in our formulation for simplicity.

With a pinhole camera, we have a 2D array of integration problems for image I that involve the first hit vertices $\mathbf{x}_i$ for $i \in [1, W] \times [1, H]$:

$$\begin{aligned} I_i &= \int_\Omega L_i(\mathbf{x}_i, \omega) g_{\phi_i}(\mathbf{x}_i, \omega) \mathrm{d}\omega \\ &\stackrel{\text{def}}{=} \int_\Omega h_i(\omega) g_i(\omega) \mathrm{d}\omega \stackrel{\text{def}}{=} \int_\Omega f_i(\omega) \mathrm{d}\omega \end{aligned} \tag{4.14}$$

where we defined

$$h_i(\omega) \stackrel{\text{def}}{=} L_i(\mathbf{x}_i, \omega) \quad g_i(\omega) \stackrel{\text{def}}{=} g_{\phi_i}(\mathbf{x}_i, \omega) \quad f_i(\omega) \stackrel{\text{def}}{=} g_i(\omega) h_i(\omega) \tag{4.15}$$

to align with our notations h_i, g_i, and f_i. Anti-aliasing could be achieved by downsampling after rendering a higher-resolution image with this formulation.

Note that $h(\omega)$ is an integral over all possible paths $\bar{\mathbf{p}}$ traced from $\mathbf{x}$ towards direction ω. Shift mapping $T_{ij}(\omega)$ as defined in Equation (2.34) in a more complete context changes both ω and path $\bar{\mathbf{p}}$. Although the full path formulation is more complete, we focus on the formulation in the angular domain rather than the path integral domain for simplicity, treating $h_i(\omega)$ as a function that returns a stochastic value. As long as the similarity assumption can be designed to hold with path-space shift mapping (i.e., the value return from h_i and h_j satisfies $h_i(\omega) \approx h_j(T_{ij}(\omega))$ for $T_{ij}(\omega) \approx \omega$), our current formulation still satisfies the similarity assumption. There are several popular choices for shift function T_{ij} and we refer readers to the survey by Hua et al. [10] for more details.

The reconstruction step is as explained already. Our framework does not need much change over the existing pipeline of gradient-domain rendering, except that we need to incorporate the basis expansion. Our method generally works well for high-frequency albedo and normal maps. However, our difference estimators may still have high variance around geometric discontinuities where incoming radiance h_i changes rapidly, which invalidate our assumptions. Fortunately, because shift mapping already produces samples that can be used for regular-gradient domain rendering, we optionally mix the result of our estimator and regular gradient-domain estimator with inverse variance weight estimated with at least two batches of partial results. We found that this optional combination with G-PT empirically works well. Figure 4.3 shows the comparison of our method with and without

combining with G-PT. It is worth noting that in L^1 reconstruction, our method without such a combination still outperforms regular gradient-domain rendering because the propagation of error in L^1 is typically not as significant as in L^2.

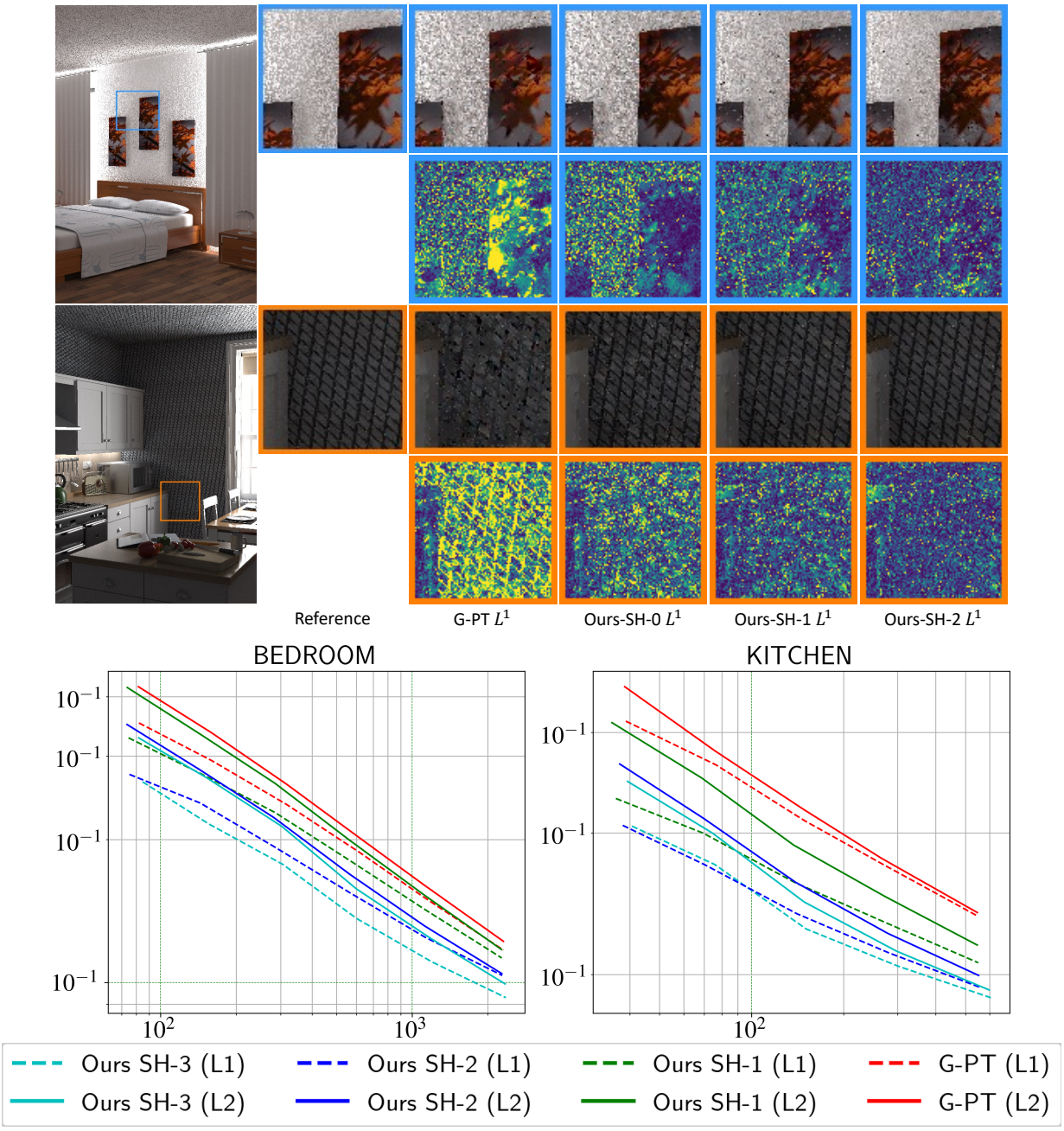


Figure 4.2: *Top: Comparison of our method with different orders of lossless basis alongside G-PT, with 128 samples per pixel under L^1 reconstruction. Bottom: Respective Convergence MAPE plot. Our method handles both the albedo and normal map better with higher-order bases.*

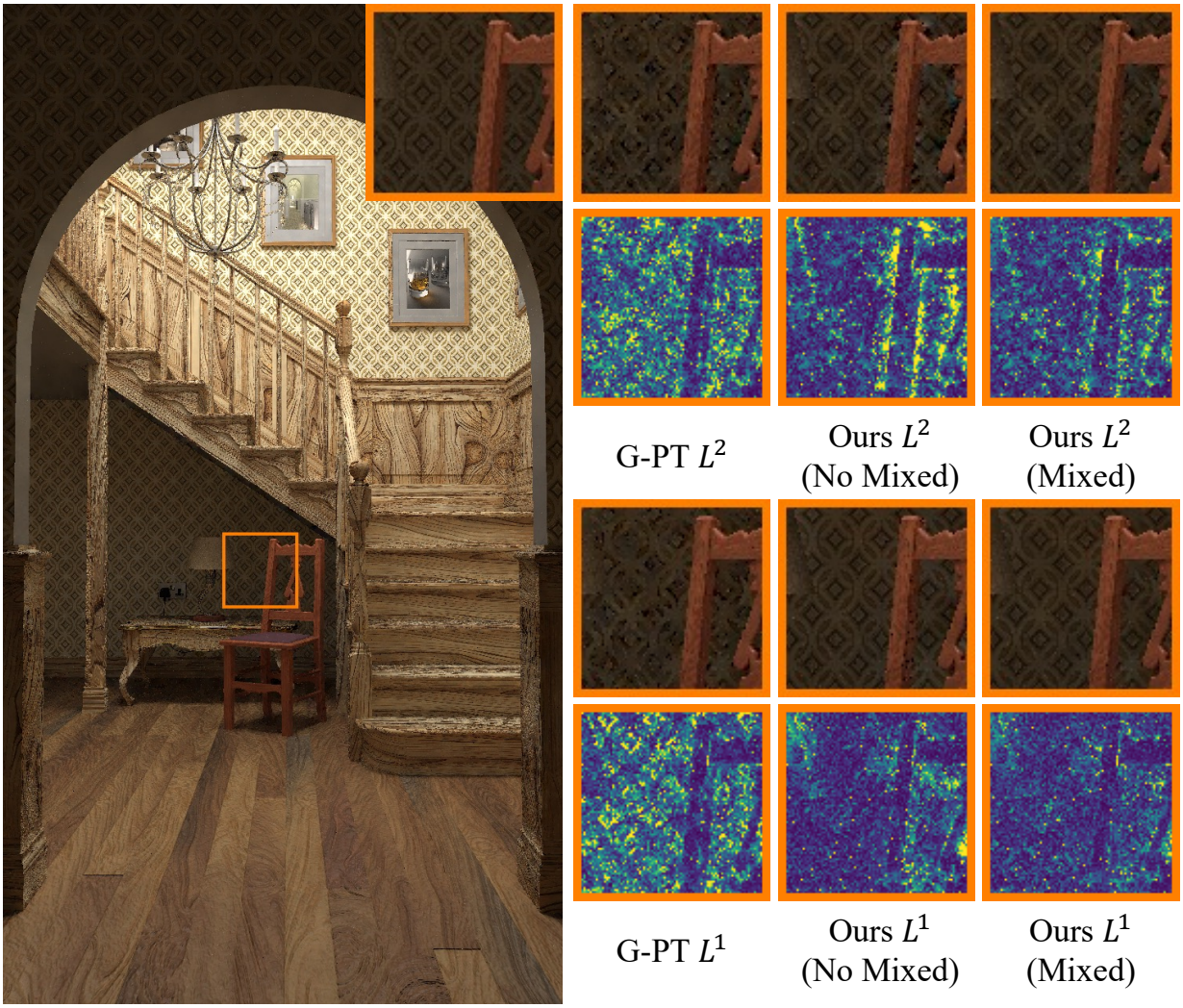


Figure 4.3: *Our result (128 spp) with/without mixing with G-PT. Our method works well for continuous surfaces with high-frequency texture maps. Artifacts that appear around geometry discontinuities could be effectively handled by combining our reconstruction with G-PT. Note that our method under L^1 reconstruction performs well even without such a combination.*

Chapter 5

Implementation

5.1 Baseline.

We implemented our own CPU gradient-domain path tracer (G-PT) [14] in LuisaRenderer [31] and our method on top of the same system. Our implementations are based on the method by [14] and we chose to omit next-event estimation for simplicity. We employed the reversible path-space shift mapping techniques to follow the suggested implementation by [14]. We only made slight modifications to regular gradient-domain path tracing to avoid some redundant calculations for our method: splitting the throughput into two parts and storing them: Basis/BSDF evaluation at the primary hit, and incoming throughput after the primary hit. When a path hits a light source, we only need to multiply the basis/BSDF evaluation of the first hit with the throughput after the hit. Primal estimates and difference estimates for each basis are stored in a buffer for the later reconstruction stage. We use the same sample to evaluate all lossless bases. The only overhead during the rendering stage compared to vanilla G-PT is basis evaluation, which is about 8% of the total rendering time with 9 basis functions.

5.2 Reconstruction.

We used PyTorch for both L^1 and L^2 reconstructions. Since we deal with convex optimization, we used the L-BFGS optimizer due to its efficiency. It has been observed that L^1 reconstruction often yields visually appealing results but may introduce bias in gradient-domain rendering. This bias is typically manifested as an overall darker image compared to

unbiased reconstruction or local blurring when high-frequency details are present. We have observed that employing L^1 loss for differences and L^2 loss for regularization addresses the brightness issue while still producing visually pleasant results. This is because the visual artifacts introduced in L^2 reconstruction are a consequence of over-penalization when noisy difference estimates are present. Therefore, we adapt L^2 regularization for both L^1 and L^2 reconstruction in both vanilla G-PT and our methods. We set $\lambda = 0.1$ for L^2 reconstruction and $\lambda = \sqrt{0.1}$ for L^1. Using the same λ has demonstrated near-optimal performance in our test scenes for both our method and G-PT, consistent with our conclusions in Section 4.3.

5.3 Mixture with G-PT.

To combine with a regular gradient-domain estimator using inverse variance weight, we divide samples into at least 2 batches and estimate the sample variance for $\langle I \rangle_{GB}$ and $\langle I \rangle_{G}$. In practice, we find that even 2 batches work effectively for high sample counts, so we set the batch number to $\max(2, \lfloor \frac{N}{64} \rfloor)$, with N being the number of samples used for rendering, demonstrating good performance in practice.

5.4 Choice of Basis.

To show that our method could easily adapt different basis functions, we test two different choices of bases: (1)2-degree spherical harmonics (SH) which has 9 bases and (2)9 independent box functions that evenly partition the spherical domain similar to the scaling function of the Haar wavelet. We used the Monte Carlo integration approach as in Equation (4.7) to solve for α_i^l with BSDF sampling as a precomputation. We set the number of samples $K = \max(150, 3N)$ with K defined in Equation (4.7) and N is the number of samples for rendering. This precomputation typically takes $2 - 3\%$ of our total rendering time. Once we have coefficients stored in buffers, one for each l, $b_i^l(\omega)$ can be efficiently calculated based on Equation (4.10).

Chapter 6

Results

All our experiments were run on a desktop with Intel(R) Core(TM) i7-5930K CPU @ 3.50GHZ with 12 threads and NVIDIA RTX 2080Ti GPU. GPU is used only for reconstruction as in the reference implementation by [14].

As our proposed method is designed to address scenarios involving high-frequency material maps, we initiate the experimentation by modifying the simple Cornell-Box scene. This experiment acts as a proof of concept and evaluates the ability of our method to handle changes in albedo, normal, and roughness with our chosen global basis. Our initial step involves generating a high-frequency checkerboard for albedo, creating a normal map by treating the checkerboard with Gaussian blur as the height map, and adjusting roughness by remapping the black and white range between 0.3 and 1. We conduct tests with only one property toggled at a time. The results are visually demonstrated in Figure 6.1, and the numerical result is shown with convergence plots in Figure 6.2. Our experiments under both L^1 and L^2 reveal that our method exhibits superior performance in handling changes in albedo and normal. While we observe a decreased performance compared to G-PT for high-frequency roughness, it is noteworthy that the error in this case is already low with G-PT itself compared to other scenarios.

As our current lossless bases focus on maintaining information in low frequency, the variance of the difference is dominated by the higher frequency "noise" defined in Section 4.2. Consequently, the "failure" case in Figure 6.1 where using constant albedo and normal with varying roughness only may not be conclusive, as high-frequency signals remain "similar" across different pixels. Therefore, we further explore how high-frequency noise affects the performance of our method. In the ablation study, we separate roughness from the choice of the checkerboard-patterned surface map, toggling both the albedo and

normal map and gradually setting roughness as a global surface variable. The results are demonstrated in Figure 6.3. We observe that with decreasing roughness, the performance of our method decreases as expected due to high approximation error in our current choices of bases.

To test our method in a more realistic setting, we modify scenes from the rendering resource [4] by applying different materials with varying albedo, normal, and roughness on six test scenes: Staircase, Bedroom, Kitchen, Living-Room, Bathroom, and additionally Lone-Monk. We demonstrate our results in Figure 6.4 with the same-time convergence plot in Figure 6.6. For Staircase, Bedroom, Kitchen, and Lone-Monk, our method is both numerically and visually better than G-PT for both L^1 and L^2, respectively. Upon closer inspection of the Staircase scene, areas under low light conditions, which originally suffered from over-blurring in L^1 reconstruction and displayed noisy artifacts in L^2, are effectively captured by our method. In the Kitchen scene, where we apply a varying grid pattern texture map with varying albedo and roughness, vanilla G-PT struggles with noisy gradients around feature discontinuities. Our method decouples these discontinuities and results in an overall much cleaner result. In the Bedroom scene with a high-frequency normal map, as anticipated, we achieve further variance reduction, particularly when high-frequency features dominate the screen. The improvement in the Lone-Monk scene is somewhat limited due to a lower frequency material map compared to previous scenes. Our method still preserves more details under L^1 and effectively removes artifacts around sharp features under L^2 (commonly termed as dipole artifacts).

We also include some failure cases. In Figure 6.5, the Living-Room scene, while qualitatively comparable to G-PT due to darkness in the texture and lack of image features, shows in the error image and the convergence plot that our method with both BOX and SH still outperforms G-PT. The bathroom scene in Figure 6.5 demonstrates a "failure" case, where varying features are far apart from each other in general. However, our method still performs visually competitively (notice the wooden pattern around the bathtub) and quantitatively better convergence than G-PT under the same-time comparison.

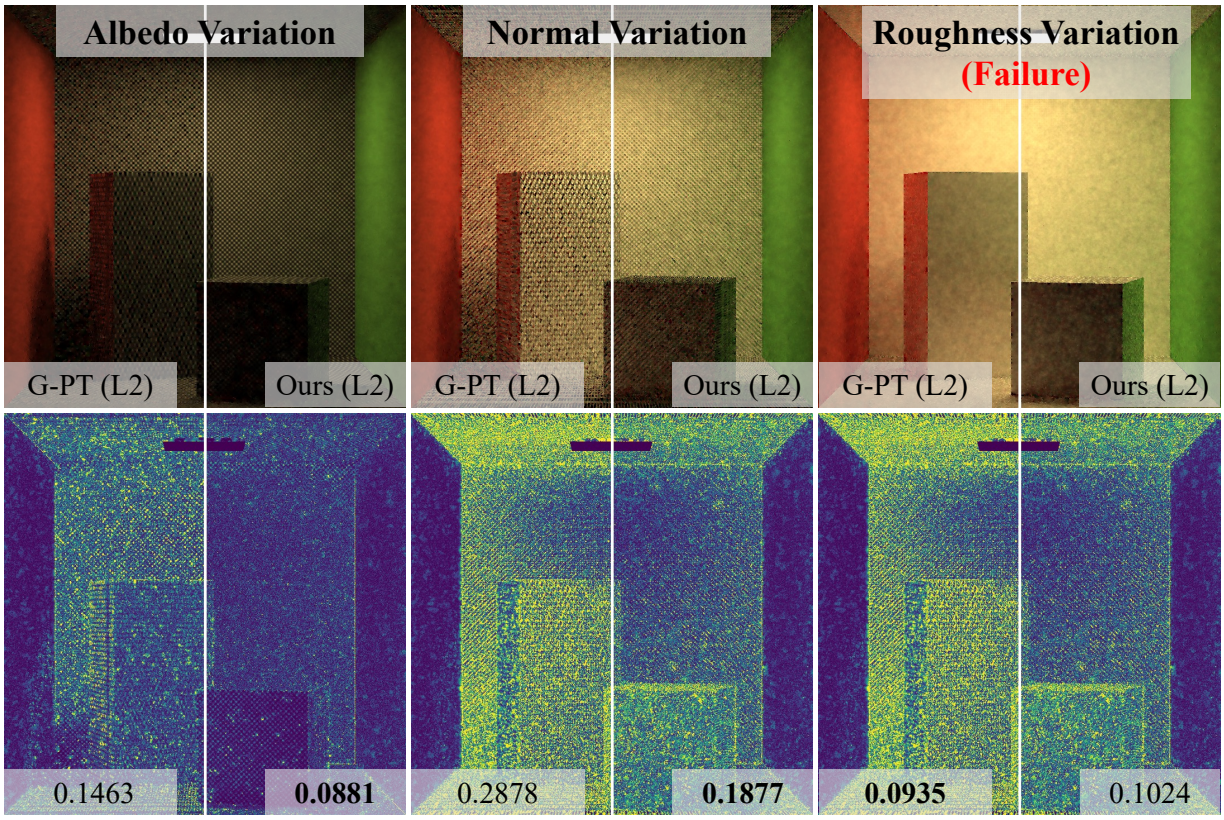


Figure 6.1: *Cornell-box test scenes with high-frequency albedo, normal, and roughness variation respectively. The top row shows the rendering result of G-PT and our method with spherical harmonics. The bottom row shows the corresponding MAPE error image. All scenes are rendered with 128 samples by our method and 147 samples by G-PT for same-time comparison. We see that with high-frequency albedo and normal map toggled, our method shows consistent improvement. When the roughness map is toggled our method performs comparable to G-PT, while the error is relatively low compared to the other two cases.*

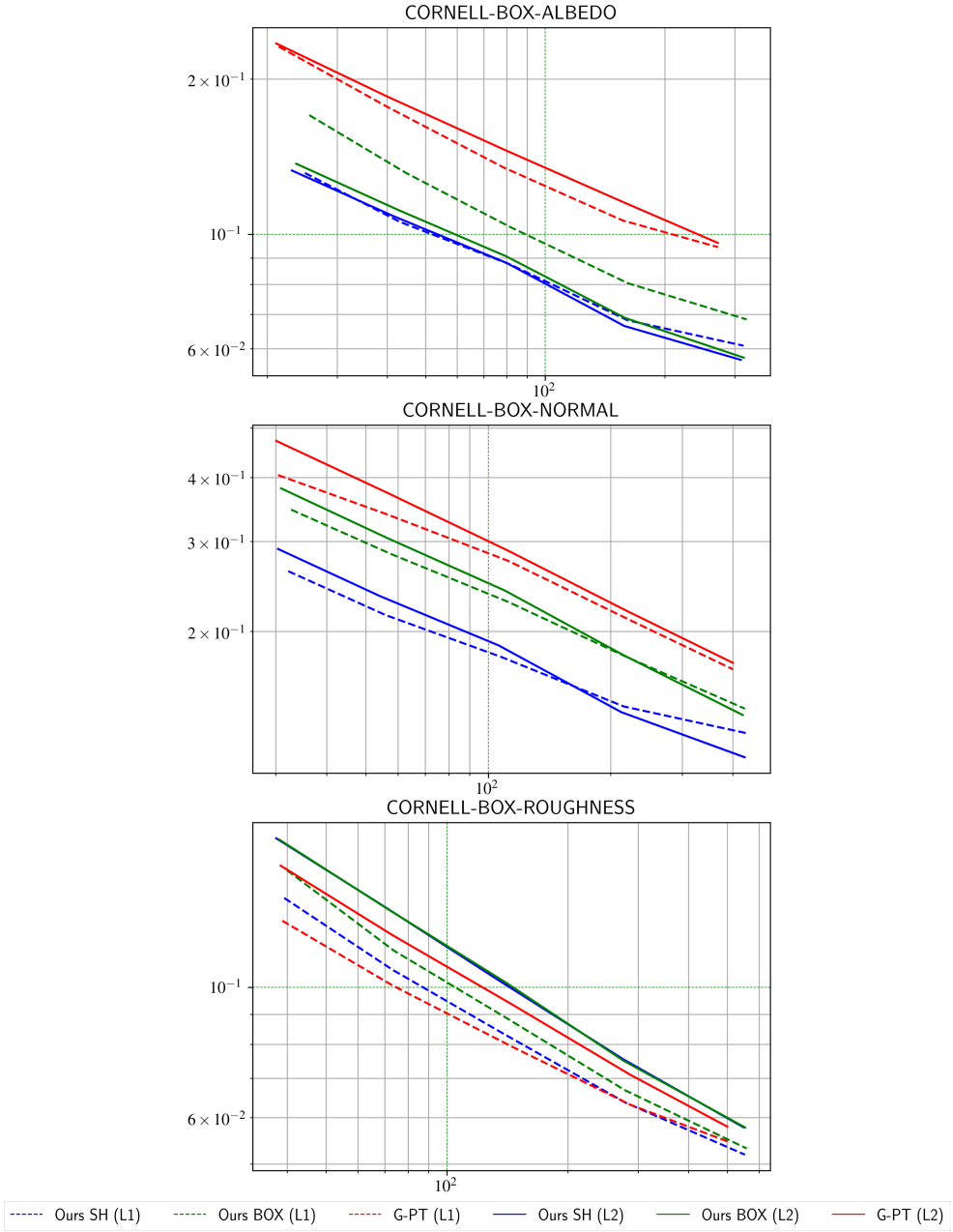


Figure 6.2: *Convergence plot (MAPE vs. Time) for high-frequency albedo, normal, and roughness toggled respectively with the Cornell box scene. The convergence plot with red represents G-PT, green for our method with box function, and blue for our method with spherical harmonics. Notice the error in the roughness toggled plot while G-PT performs better, the error is relatively low compared to the other cases.*

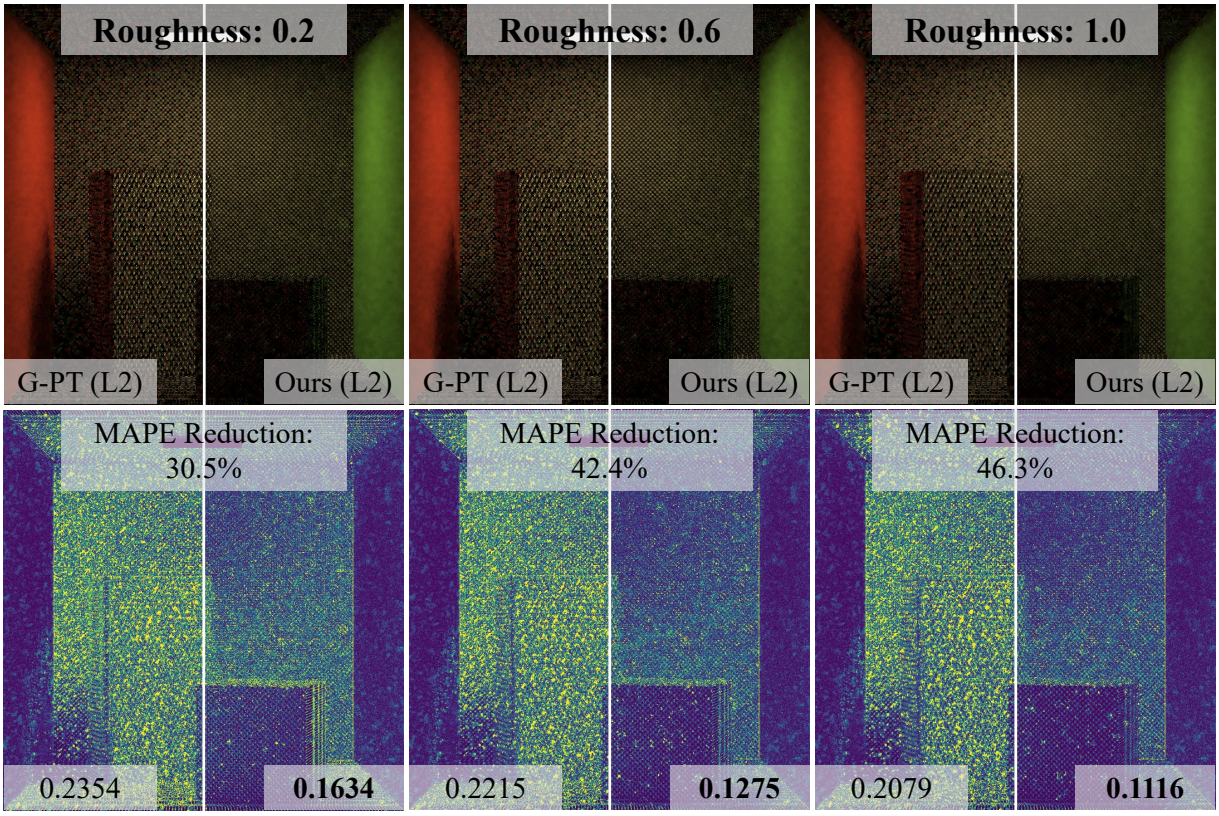


Figure 6.3: *Visualization of error images with G-PT and our reconstruction with gradually changing roughness from 0.2 to 1.0. Our method still shows variance reduction even in low roughness settings with high-frequency albedo and normal maps. While a higher roughness value allows more informative basis expansion which will result in better variance reduction overall.*

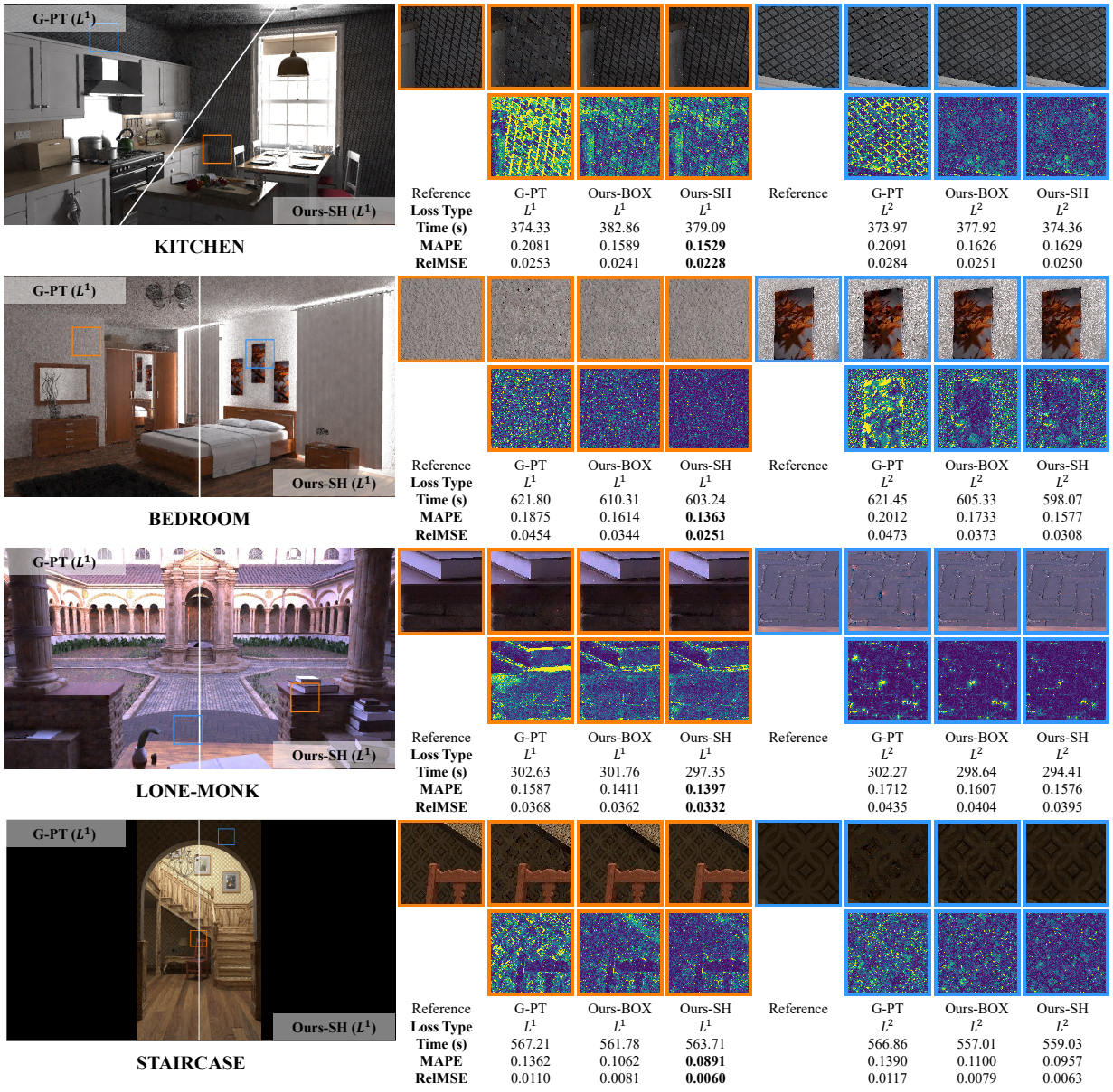


Figure 6.4: *Equal time visual comparison reconstruction result for G-PT, our method with spherical harmonics and box basis. We include rendered result and error image visualization with MAPE metric for both L^1 (red box) and L^2 (green box) reconstruction. corresponding MAPE and RelMSE for the full image are provided as numerical reference. In almost all cases our method with spherical harmonics performs best both visually and numerically. Note that in most of the cases box function still outperforms regular G-PT.*

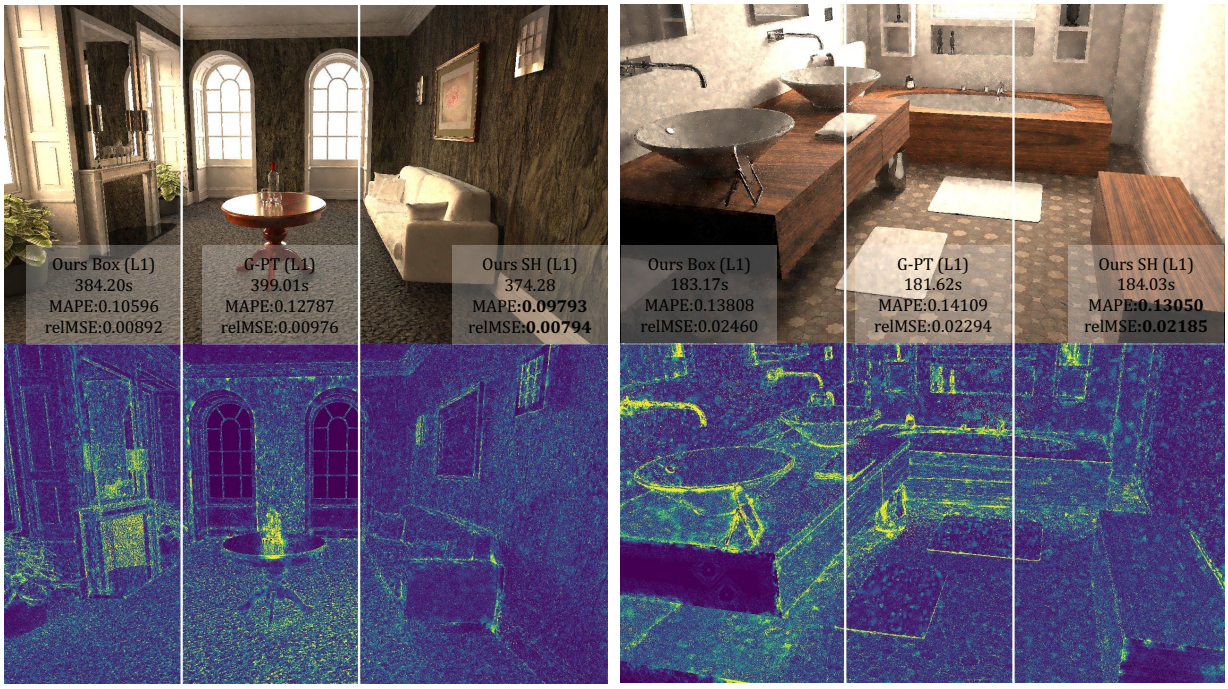


Figure 6.5: *The scene on the left (Living-Room) indeed contains high-frequency information. While the lack of visible features and dark texture limits the qualitative improvement, we can still visualize the quantitative improvement from the error image. The scene on the right (Bathroom) lacks high-frequency features. Although the tiles on the floor are patterned and have similar colors, they dominate many pixels, restricting the benefit of our algorithm. It is worth noting that our method with spherical harmonics (SH) still reduces error for the wooden pattern on the side of the bathtub, and overall, it still performs numerically better than the baseline G-PT.*

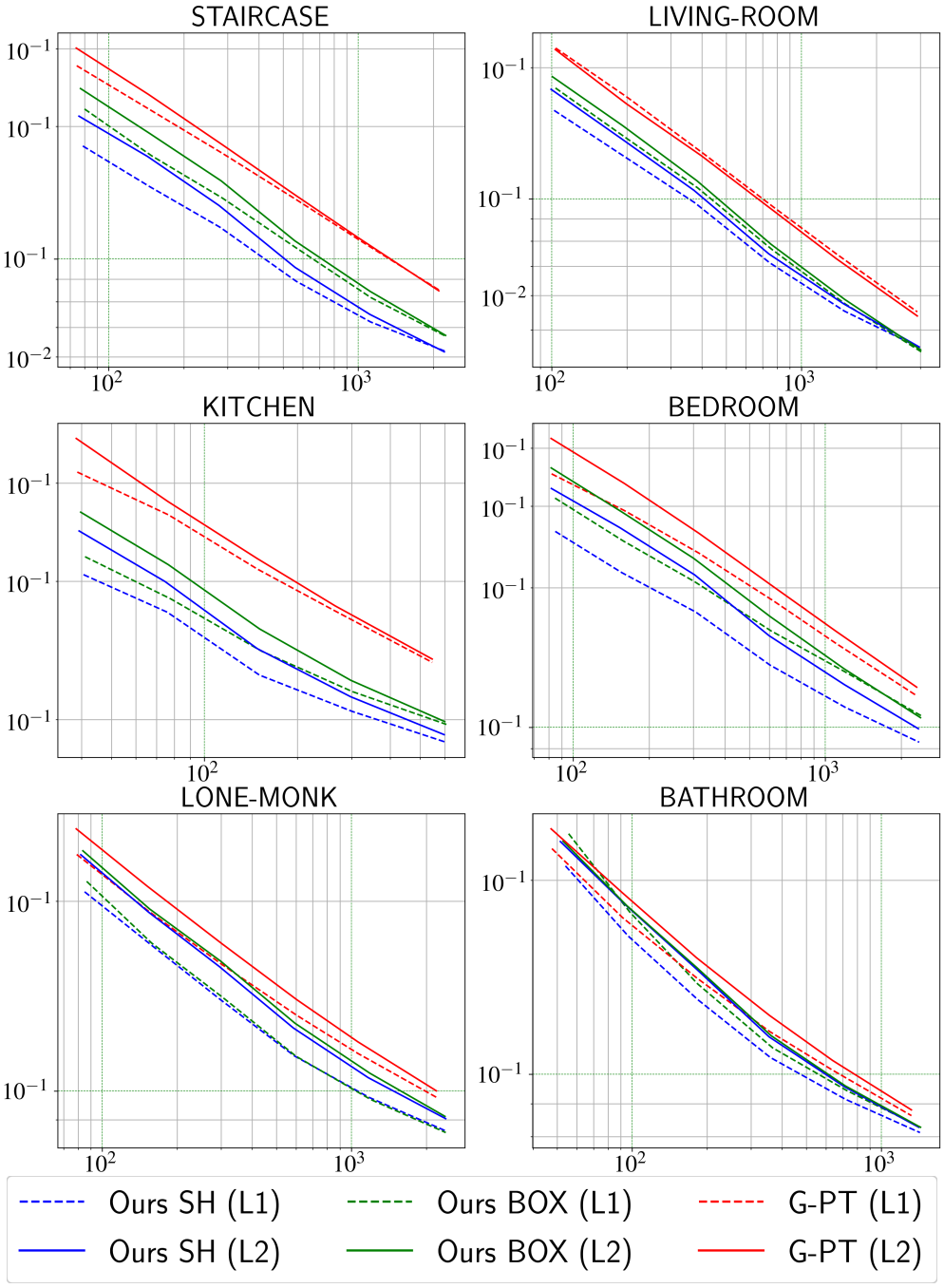


Figure 6.6: *The convergence plot for both L^1 and L^2 reconstructions is presented. Dashed lines represent L^1, and solid lines represent L^2. It is evident that under the same reconstruction loss, our method with both choices of basis consistently outperforms G-PT.*

Chapter 7

Limitations and future work

Similar to related work in gradient-domain rendering [32, 20, 13], our method currently does not support depth of field or motion blur effect. This issue is common to any technique that uses pixel-based information, such as per-pixel shading normal or albedos, so it is not unique to our method. For cases where we would like multiple subpixel samples, the computation for coefficients must be done for each sample, which introduces extra computation and memory costs. Using a common set of coefficients by varying residual distribution does not help due to our assumption that coefficients could vary significantly across pixels. One potential solution is to precompute the coefficients for arbitrary normal directions, but we leave it as future work. We want to mention that for antialiasing, rendering at a high resolution and downsampling is often the preferable approach because tone mapping should be done before antialiasing to properly deal with aliasing in HDR images. Our current method is compatible with this common practice.

Our method does not consider stochastic BSDF evaluation [8]. It could still help in variation reduction when conditional BSDF evaluation on the incoming direction has sufficiently low variance. Our current implementation does not handle specular reflection towards secondary hits and beyond. When encountered, it automatically falls back to regular gradient-domain rendering for such pixels. Although we could apply our method at the first hit with non-perfect specular BSDF, in general, such a scenario can only be handled properly with world space techniques, which we leave as future work.

Our numerical experiments show that spherical harmonics consistently outperform the box function. However, we have not identified any theoretical reason behind it. By designing a better choice of global bases under the same lossless basis construction, as discussed in Section 4.2, we could ideally achieve even further improvement. Additionally, while our

results experimentally show performance improvements, we currently lack theoretical understanding compared to G-PT. We ideally would like to have a provable improvement over G-PT since G-PT is a special case of our method. Although our combination with G-PT helps get rid of artifacts introduced by our method, a better reconstruction technique or a better design of pixel-wise basis could yield a more stable reconstruction step.

Inspired by prior work on advanced reconstruction in gradient-domain rendering [20, 13, 7], we believe that incorporating more advanced reconstruction techniques into our basis integral formulation could lead to immediate improvements. One can interpret our decomposition of gradient-domain rendering as a more effective path reuse among pixels, which may have applications in some recent work on spatial-temporal path reuse [17, 22, 5]. While our current method is designed for offline use, we envision that a more specialized lossless basis expansion focused on cosine-weighted BSDFs evaluation could potentially facilitate real-time applications in the future.

Another future work is in the realm of generalized denoising. Modern denoising methods rely on auxiliary image space data [2, 13, 33] to capture pixel-dependent information and to avoid over-blurring of sharp features. However, high-frequency variation, even if present in those auxiliary data does not necessarily indicate entirely different pixel intensities. For example, it is often the case that incident illumination is smooth even when albedo changes rapidly [29, 15]. Our basis expansion approach decouples this high-frequency per-pixel variation, and this concept can be useful to apply to denoising. Our framework itself should also be applicable to any problems where we need to solve multiple integrals that are correlated to each other.

References

[1] Image-space control variates for rendering.

[2] Attila T. Áfra. Intel® Open Image Denoise, 2023. https://www.openimagedenoise.org.

[3] Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. Gradient-domain path reusing. *ACM Transactions on Graphics (TOG)*, 36(6):1–9, 2017.

[4] Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/.

[5] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)*, 39(4):148–1, 2020.

[6] Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. Gradient-domain volumetric photon density estimation. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[7] Jie Guo, Mengtian Li, Quewei Li, Yuting Qiang, Bingyang Hu, Yanwen Guo, and Ling-Qi Yan. Gradnet: unsupervised deep screened poisson reconstruction for gradient-domain rendering. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

[8] Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. Multiple-scattering microfacet bsdfs with the smith model. *ACM Trans. Graph.*, 35(4), jul 2016.

[9] Binh-Son Hua, Adrien Gruson, Derek Nowrouzezahrai, and Toshiya Hachisuka. Gradient-domain photon density estimation. In *Computer Graphics Forum*, volume 36, pages 31–38. Wiley Online Library, 2017.

[10] Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. A survey on gradient-domain rendering. In *Computer Graphics Forum*, volume 38, pages 455–472. Wiley Online Library, 2019.

[11] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.

[12] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery.

[13] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. Deep convolutional reconstruction for gradient-domain rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[14] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Trans. Graph.*, 34(4), 2015.

[15] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005.

[16] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain metropolis light transport. *ACM Trans. Graph.*, 32(4), 2013.

[17] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. Generalized resampled importance sampling: foundations of restir. *ACM Transactions on Graphics (TOG)*, 41(4):1–23, 2022.

[18] Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain bidirectional path tracing. In *EGSR (EI&I)*, pages 65–74, 2015.

[19] Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. Temporal gradient-domain path tracing. *ACM Transactions on Graphics (TOG)*, 35(6):1–9, 2016.

[20] Marco Manzi, Delio Vicini, and Matthias Zwicker. Regularizing image reconstruction for gradient-domain rendering with feature patches. In *Computer graphics forum*, volume 35, pages 263–273. Wiley Online Library, 2016.

[21] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. In *ACM SIGGRAPH 2004 Papers*, pages 477–487. 2004.

[22] Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. Restir gi: Path resampling for real-time path tracing. In *Computer Graphics Forum*, volume 40, pages 17–29. Wiley Online Library, 2021.

[23] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM SIGGRAPH 2003 Papers*, 2003.

[24] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 339–348. 2023.

[25] Weilun Sun, Xin Sun, Nathan A Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. Gradient-domain vertex connection and merging. In *EGSR (EI&I)*, pages 83–92, 2017.

[26] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[27] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[28] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, 1997.

[29] Gregory J Ward, Francis M Rubinstein, and Robert D Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, 1988.

[30] Zilin Xu, Zheng Zeng, Lifan Wu, Lu Wang, and Ling-Qi Yan. Lightweight neural basis functions for all-frequency shading. In *SIGGRAPH Asia (Conference Track)*, November 2022.

[31] Shaokun Zheng, Zhiqian Zhou, Xin Chen, Difei Yan, Chuyan Zhang, Yuefeng Geng, Yan Gu, and Kun Xu. Luisarender: A high-performance rendering framework with layered and unified interfaces on stream architectures. *ACM Trans. Graph.*, 41(6), nov 2022.

[32] Tao Zhuang, Pengfei Shen, Beibei Wang, and Ligang Liu. Real-time denoising using brdf pre-integration factorization. In *Computer Graphics Forum*, volume 40, pages 173–180. Wiley Online Library, 2021.

[33] Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. In *Computer graphics forum*, volume 34, pages 667–681. Wiley Online Library, 2015.