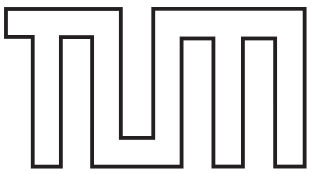


Technische Universität München
Chair of Media Technology
Prof. Dr.-Ing. Eckehard Steinbach

Master Thesis

Smooth Textured Surface Reconstruction from Point Cloud Rendered Images as well as Path Traced Images Using Variational Methods

Author:	Han Liang
Matriculation Number:	03737290
Address:	Arcisstr. 21
	80333 München
Advisor:	M. Sc. Josef Stumpfegger
	M. Sc. Christoph Neuhauser
	M. Sc. Annalena Ulschmid
	Prof. Dr. Rüdiger Westermann
	Prof. Dr. Michael Wimmer
Begin:	16.01.2023
End:	16.07.2023

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, July 17, 2023 _____

Place, Date Signature

München, July 17, 2023 _____

Place, Date Signature

Abstract

In this work, we propose total variation-based methods for smoothing textured surfaces in point-based rendering and reducing noise in Monte Carlo-rendered images. Initially, we survey the challenges and existing state-of-the-art methodologies in these two research domains. Subsequently, we delve into the details of our proposed total variational models, each aimed at smoothing point-rendered textured surfaces and reducing noise in Monte Carlo-rendered images, respectively. For smoothing textured surfaces in point-based rendering, our model incorporates geometric features and is then combined with an advanced Pull-Push method. This combined approach enables us to effectively fill gaps and smooth discontinuous surfaces. The models tailored for denoising Monte Carlo-rendered images leverage noise-free auxiliary features and noise estimation techniques. Our approach efficiently eliminates noise while preserving crucial image features. We conduct comprehensive comparison experiments against existing state-of-the-art techniques to evaluate the effectiveness of our methods. Although our implementations are currently offline, both the smoothing and denoising processes can be achieved within a few iterations. Given the simplicity of our approach's implementation, we foresee the potential for a GPU-based implementation, paving the way towards real-time applications.

Contents

Chapter 1

Introduction

Point-based rendering and Monte Carlo rendering are two essential techniques in the field of computer graphics. Both of these techniques are remarkable for their capacities. Point-based rendering is recognized for its ability to efficiently capture complex 3D geometry. However, it is often challenging to create smooth, continuous surfaces from point-sampled objects. On another front, Monte Carlo rendering methods are highly valued for their ability to produce realistic lighting and shading effects. However, one significant challenge associated with these methods is the presence of noise in the rendered images, especially when the number of samples per pixel is low. Addressing these challenges is the primary objective of this thesis.

In this thesis, we aim to smooth the reconstruction of textured surfaces by point-based rendering and reduce noise in Monte Carlo rendered low samples per pixel images. To maximize efficiency, we avoid performing pre-computations in the object space. All our processing steps occur in the image space. We delve into the field of image restoration, a critical process in image processing that involves retrieving an undistorted image from a distorted one. The process of image restoration often involves solving an inverse problem. Mathematically, given a distorted image u_0, the goal of an inverse problem is to find the original, undistorted image, represented by u, from this distorted version. Variational methods are a widely utilized approach for solving inverse problems. These methods strive to minimize a squared energy function with a regularity penalty $R(u)$, given by:

$$u = \arg\min_{u} \frac{1}{2}\|u - u_0\|^2 + R(u) \tag{1.1}$$

This penalty $R(u)$ encourages the solution to be smooth while reducing distortions. In solving such optimization problems, algorithms typically reformat the problem as a constrained optimization task, which enables them to iteratively find the optimal solution for u. A prominent example of a regularization term $R(u)$ is the total variation-based regularization $R_{TV}(u)$, well-known for its capability to smooth an image while preserving significant structural features.

In this research, we apply total variation-based methods as a post-processing step in image space to address the previously discussed challenges. We adapt the total variation-based techniques to effectively smooth discontinuous surfaces while preserving essential features from the original point-based rendering. To address the noise in low sample per pixel frames produced by Monte Carlo rendering, we propose a total variation-based method that integrates auxiliary features and noise estimation techniques. Our approach enables effective noise reduction while maintaining important image features.

This thesis is organized as follows: The Chapter 2 begins with a comprehensive explanation of the fundamental concepts behind total variation-based methods. We then delve into the challenges associated with obtaining continuous surfaces from point-based rendering and explore the difficulties in denoising Monte Carlo rendering. This chapter also presents an overview of the related approaches currently used to address these challenges. In Chapter 3, we detail our proposed total variation-based models, designed specifically to address the challenges of obtaining continuous surfaces from point-based rendering and denoising Monte Carlo rendering, respectively. Then, in Chapter 4, we proceed to evaluate the effectiveness of these models by conducting a thorough comparison of our results with those obtained using state-of-the-art methods. Finally, in Chapter 5, we present our conclusions, summarizing the key findings from our work. We also discuss future perspectives, suggesting possible directions for further study and development based on our research findings.

Chapter 2

Background

In this thesis, we address two primary research issues: the smoothing textured surface reconstruction from point cloud-rendered images and the denoising of path-traced images. We tackle these problems by formulating them as constrained optimization problems, which are based on the total variation principle. The initial part of this chapter begins by explaining the mathematical representation of a constrained optimization problem in the context of our research problems. Subsequently, we demonstrate the fundamental concept of total variational models, highlighting their effectiveness within the scope of our research issues. Lastly, we present recent work related to each research problem, discussing their respective strengths and weaknesses.

2.1 Mathematical Foundations of Total Variation-based Methods

Constrained optimization problems refer to optimization functions whose solutions are bound by constraints. A constrained optimization problem is solved by finding the best solution (either maximum or minimum) that satisfies certain conditions. In the context of image denoising and inpainting, the objective is to minimize the difference $f(u, u_0)$ between the reconstructed image u and the noisy or incomplete image u_0, subject to specific constraints $g(u)$ that ensure the solution's feasibility and optimality:

$$\begin{aligned} u &= \arg\min_{u} f(u, u_0) \\ &\text{subject to } g(u) = c \end{aligned}$$

To solve such problems, we utilize the Lagrange multiplier method, which introduces new variables for each constraint, resulting in a new objective function:

$$L(u, \lambda) = f(u, u_0) - \lambda(g(u) - c) \tag{2.1}$$

By solving Euler-Lagrange equations 2.2 and 2.3, we can iteratively find the optimal solution for u.

$$\frac{\partial L}{\partial u} = 0 \tag{2.2}$$

$$\frac{\partial L}{\partial \lambda} = 0 \tag{2.3}$$

Total Variation (TV) based methods are classic mathematically constrained optimization strategies for image denoising and inpainting tasks. Owing to their ability to effectively smooth images while preserving edges, TV models demonstrate considerable promise in resolving our research issues. As a foundation for further exploration, we will next outline the core concept of TV-based methods, focusing on the classic Rudin-Osher-Fatemi (ROF) model.

The TV model was initially introduced in [ROF92]. TV-based methods incorporate the TV term to construct the objective optimization function. In the context of image processing, the TV term is essentially the l1 norm of the gradient of the restored image u, represented as $R_{TV}(u) = |\nabla u|$. Drawing on the statistical observation that natural images exhibit local smoothness and that the pixel intensity transitions gradually across most regions, [ROF92] suggests that the l1 norm of the image gradient is particularly effective for image restoration tasks.

To provide a comprehensive understanding of how this method enhances image smoothness, we will next explain the ROF model contributed by [ROF92]. The optimization function for the ROF model in its discrete form is defined as:

$$u = \arg\min_{u} \frac{\lambda}{2}\|u - u_0\|^2 + |\nabla u| \tag{2.4}$$

where $\frac{\lambda}{2}\|u - u_0\|^2$ is the data fidelity term for preserving important features from the noisy images and $|\nabla u|$ is the TV term for promoting piecewise smoothness of the restored image. The strategic interaction of these two components enables the model to smooth images effectively while preserving essential structures. The two components are defined as:

$$\|u - u_0\|^2 = \sum_{i,j} (u_{i,j} - u_{0_{i,j}})^2 \tag{2.5}$$

$$|\nabla u| = \sum_{i,j} \sqrt{(\partial_x u)^2 + (\partial_y u)^2} \tag{2.6}$$

where $\partial_x u$ and $\partial_y u$ are the first-order differences of u along x and y directions, respectively.

TV-based methods had made significant success in the field of image denoising [FZFZ19] and image inpainting [CZL+21] owing to their ability to not only efficiently compute the optimal solution, but also maintain sharp edges in the image. Numerous algorithms have

been developed to solve optimization problems of TV models, including Primal-Dual methods [CP11], alternating direction method of multipliers (ADMM) method [EB92], and the Split Bregman method [GO09]. According to the work by [WT10], these optimization algorithms share fundamental similarities. No single algorithm universally outperforms the rest across all optimization problems. The choice of algorithm can significantly influence the problem-solving process. In this thesis, specific algorithms are selected for different models based on their suitability for the problem at hand and the clarity of their working principles.

In the field of computer graphics, there is a related work that employs a TV-based model to smooth fluid surfaces obtained via sphere rendering [RCSW14]. Their approach applies a screen-space post-smoothing filter to the depth buffer to enhance the image of surfaces obtained via sphere rendering. Remarkably, this filter integrates the ROF model with a locally adapted balance parameter. Our first research problem, filling and smoothing point rendered images, exhibits similarities with this application. Inspired by it, we designed a custom TV-based loss function and applied it to the depth map to infill holes in an edge-aware manner. Then the textured surface is smoothed, guided by the smoothed depth map. Furthermore, we extended our TV models to denoise Monte Carlo rendering, which also possesses geometric features that can contribute to custom objective function design. Nonetheless, the inpainting in point-rendered images and denoising tasks in Monte Carlo rendering differ from traditional image processing tasks. The specifics in data structure and noise properties are taken into consideration so that the custom TV models can reconstruct the final image as best as possible.

2.2 Reconstruction of Point-Rendered Surfaces

Point-based geometry is represented by 3D positions sampled from a continuous surface, often accompanied by associated attributes such as normal vectors, depth values, color, and other material properties. Points were initially proposed as rendering primitives in [LW85]. Given that topological consistency is not strictly necessary in rendering, point-sampled objects offer greater flexibility compared to other models. One popular method is one-pixel point rendering, where each point in the point cloud directly corresponds to a pixel in the rendered image. By interpreting each point as a pixel, this method can be highly efficient [SKW22]. However, one-pixel point rendering also has its challenges. The lack of a continuous surface representation often leads to noticeable surface discontinuities in the rendering. This issue becomes particularly challenging when dealing with massive, unstructured raw point cloud data. To mitigate these issues, numerous studies have been conducted [KB04] [KMŽ+22] to fill the holes and gaps between samples. In the following section, we will explore and discuss recent related work to tackle this research problem.

2.2.1 Related Work

Traditional methods generally fall into two broad categories: image-space techniques and object-space techniques. Besides these, deep learning-based methods have been introduced for both categories.

- **Object-space reconstruction techniques.** Some methods aim to reconstruct the entire point cloud into a renderable surface representation, such as a mesh. The mesh can be described based on an implicit surface that is locally fitted to the data [KBH06][KH13][KCRH20], or it can be approximated by local higher-order polynomials [ABCO+03]. However, rendering a locally fitted surface requires dynamic adjustment of the sampling rate to match the current viewing parameters for each frame, making these approaches computationally demanding. Moreover, these methods tend to be inefficient when only part of the scene is viewed. They typically perform slower than approaches that employ surface elements like splats. In splat-rendering, points are interpreted as object-space disks or ellipses, accompanied by normal vectors and radius. These overlapped splats are rendered instead of points to bridge the gaps between neighboring point samples. The high-quality Elliptical Weighted Average (EWA) splatting framework [ZPVBG01], which employs object-space reconstruction kernels and image-space filters, results in hole-free, high-quality rendering. Nevertheless, such implementation can be computationally expensive when dealing with highly complex models.
- **Neural rendering approaches.** Deep learning-based approaches have been proposed to generate high-quality [EGO+20], photo-realistic [BLMD18] point renderings from low-resolution point clouds. However, their complex network structures limit them to offline applications. In [RFS22], a deep learning method is proposed for real-time point rendering, which uses calibrated camera images and estimated point clouds as inputs. However, the limitation of their neural network's architecture is that it can only fill in gaps up to a specific size. This results in non-continuous surfaces when the point cloud is sparse or when the camera approaches an object too closely.
- **Image-space operators.** As the quantity of surface points grows, it becomes more beneficial to transfer surface approximation algorithms to image space. In [RL08], an iterative hole-filling technique is employed for datasets of moderate size. Efficient pyramid Pull-Push algorithms are utilized in [PGA11] and [BDG18]. A noted drawback of these Pull-Push based methods is the introduction of blocking artifacts.

In this research, we go back to the basics of rendering points directly. Using a TV-based method as a post-process step, we fill in the holes in a geometry-aware manner in the image space.

2.3 Denoising Monte Carlo Rendering

Path tracing takes a leading role in graphics rendering research, attributed to its capability to produce physically-based illumination effects. Monte Carlo methods offer an efficient approach to compute rendering results through stochastic sampling of light paths. As the quantity of sampled paths increases, the estimated image converges toward an accurate solution. Monte Carlo methods universally encounter noise (variance) in the estimated pixel values, particularly when the sampling rate is low. Traditional denoising techniques in computer vision typically make assumptions that the noise is evenly spread across the image or follows a specific statistical distribution. However, these assumptions may not hold for Monte Carlo noise. Monte Carlo noise carries unique characteristics. Firstly, noise resulting from Monte Carlo rendering is unevenly distributed within the image, with complex lighting areas typically possessing more noise. Secondly, the noise can correlate with the scene's geometry, while standard denoising methods assume noise is independent of image content. Furthermore, Monte Carlo noise often presents as high-frequency noise which impacts fine details within the image. Its removal can subsequently lead to the loss of these details. Therefore, denoising techniques for Monte Carlo rendering often need to take into account the specific properties of the noise. The related work addressing the denoising of Monte Carlo rendering will be introduced next.

2.3.1 Related Work

This section primarily focuses on post-processing techniques. These methods are based on statistical analyses of sample sets obtained from Monte Carlo rendering, along with additional feature information.

Alongside noisy images, auxiliary first hit image features such as per-pixel normal, depth, and albedo colors can also be acquired from the path tracer. According to [RMZ13] and [MIGMM17], the outcomes of the restoration process can be enhanced by utilizing so obtained feature buffers. These buffers provide a wealth of noise-free data about image structures. In [McC99], an offline method based on anisotropic diffusion guided by normal and positions is proposed to blur noisy images iteratively. Locally adaptive filters can be derived from auxiliary features to prevent blurring samples across geometric edges. The À trous filter, proposed in [DSHL10], leverages wavelets for real-time filtering and integrates a customized edge-aware stop function. [KIM+19] introduces a real-time block-wise reconstruction pipeline that also utilizes auxiliary features to design a multi-order regression-based model. Moreover, reprojecting and accumulating samples across multiple frames can yield a larger effective spp count to reduce noise and enhance temporal stability. In [SKW+17] and its subsequent work [BWP+20], a hierarchical wavelet local filter guided by spatiotemporal luminance variance shows promising results in denoising for temporally accumulated 1 spp frames. Deep learning-based approaches also represent a compelling direction for achieving real-time Monte Carlo denoising [VRM+18], [MZV+20]. For example, the OptiX Neural Network Denoiser [CKS+17] uses recurrent connections in each neural

network layer to address temporal noise.

Most real-time denoising strategies rely on image space filtering due to its simplicity and efficiency. As suggested in [KS13], we can borrow locally adaptive denoising strategies from the field of image processing to effectively deal with Monte Carlo noise. Drawing inspiration from this, we adaptively modify the aggressiveness of denoising parameters in the TV model, thus addressing the characteristics of the noise in different image areas. Building upon prior work, we utilize auxiliary features to isolate fine details and incorporate temporal information to ensure both temporal stability and to acquire more valid samples.

Chapter 3

Proposed Methods

In this chapter, we provide an in-depth description of our TV-based methodologies for inpainting and denoising tasks, in Section 3.1 and Section 3.2, respectively. In Section 3.1, we integrate gradually new features into the fundamental TV model, eventually allowing us to smooth the textured surface from point rendered images efficiently. This section also explains in detail the algorithms we used to solve the corresponding optimization problems. In Section 3.2, we utilize auxiliary features and noise estimation techniques to adapt TV models for Monte Carlo denoising tasks.

3.1 TV-based Methods for Smoothing Point-Rendered Images

Figure 3.1 shows an example of a point-rendered texture image, paired with its corresponding depth map (where each pixel value signifies the distance from the viewpoint, as illustrated in Figure 3.3b) and normal map (which represents surface orientations, as illustrated in Figure 3.1b). Our goal is to infill the discontinuous surface, especially the discontinuous surfaces near the camera, as marked regions in Figure 3.1. These auxiliary features, such as depth and normal maps, can offer valuable information for surface reconstruction. We propose to employ a TV-based method to bridge gaps in the depth map. Using the infilled depth map, we proceed to reconstruct both the normal map and the texture image. The reasons that we first smooth the depth map and then use it to guide the reconstruction of textured surface and normal map are as follows:

1. A smoothed depth map can provide basic and consistent information for the reconstruction of other maps. It can serve as a more accurate and stable guide for the reconstruction of the normal map and texture images, yielding higher quality results.

2. TV-based methods are known for their edge-preserving properties. They can effectively smooth within regions while preserving sharp discontinuities at the boundaries. Given

(a) Point-Rendered Albedo (b) Point-Rendered Normal (c) Point-Rendered Depth

Figure 3.1: Point-Rendered Images

that depth maps represent more directly the scene geometry compared to normal maps and texture images, using the TV model to smooth it first is a more straightforward approach to restoring the underlying geometry.

We begin by applying the Beltrami framework as the baseline model to the raw depth map. The objective function of this model is solved using an efficient primal-dual algorithm proposed in [ZB14]. Observations from the results by this initial model allow us to gradually add new features to enhance the reconstruction performance. Each improvement will be detailed in the subsequent sections.

3.1.1 Beltrami Framework

The Beltrami framework can effectively balance the preservation of important features and the reduction of staircasing artifacts in the reconstructed image [ZB14]. The discrete form of the Beltrami framework for inpainting is defined as :

$$\min_u \left\{ \left(\sqrt{1+\beta^2\|\nabla u\|^2} + \frac{\boldsymbol{\lambda}_e}{2}\|u-u_0\|^2 \right) \right\} \tag{3.1}$$

where $u \in R^{M\times N}$ represents the unbiased image and $u_0 \in R^{M\times N}$ denotes the raw, distorted image. The components of this optimization problem are defined as:

$$\|\nabla u\|^2 = \sum_{i,j} \left((\partial_x u_{i,j})^2 + (\partial_y u_{i,j})^2 \right) \tag{3.2}$$

$$\|u-u_0\|^2 = \sum_{i,j} (u_{i,j} - u_{0_{i,j}})^2 \tag{3.3}$$

$\boldsymbol{\lambda}_e = \lambda \boldsymbol{\chi}_D$ is the weighted binary mask, where $\boldsymbol{\chi}_D$ represents a binary foreground mask with 0 representing holes and 1 representing foreground pixels:

$$\boldsymbol{\chi}_{Di,j} = \begin{cases} 0, & \text{hole} \\ 1, & \text{foreground pixel} \end{cases}$$

And the $\nabla u = (\partial_x^+, \partial_y^+)$ satisfies the Neumann boundary conditions through a forward difference scheme:

$$\partial_x^+ u_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j}, & \text{if } 1 \le i \le M,\ 1 \le j < N \\ 0, & \text{if } 1 \le i \le M,\ j = N \end{cases} \tag{3.4}$$

$$\partial_y^+ u_{i,j} = \begin{cases} u_{i,j} - u_{i-1,j}, & \text{if } 1 < i \le M,\ 1 \le j \le N \\ 0, & \text{if } i = M,\ 1 \le j \le N \end{cases} \tag{3.5}$$

In Equation 3.1, β and λ serve as balancing parameters that mediate between the preservation of important features and the avoidance of staircasing artifacts. Staircasing is a common issue introduced by TV models since the TV term $|\nabla u|$ tends to create artificial edges that divide the image into piecewise constant patches. The Beltrami regularization term $\sqrt{1+\beta^2\|\nabla u\|^2}$ reduces this issue by allowing soft gradients to remain smooth. The functional with this term can be efficiently solved using a primal-dual projected gradient algorithm [ZB14].

Solving Algorithm

To solve the minimization problem, we can reformat Equation 3.1 into two separate subproblems:

$$F(Ku) := \sqrt{1+\beta^2\|\nabla u\|^2} \tag{3.6}$$

$$G(u) := \frac{\boldsymbol{\lambda}_e}{2}\|u - u_0\|^2 \tag{3.7}$$

Here, K is a linear operator, which in this context represents the gradient operator ∇. By using the concept of the Legendre-Fenchel transform [HNAD11], a primal problem can be transformed into a primal-dual form:

$$\underbrace{\min_{u \in R^{M\times N}} (F(Ku) + G(u))}_{\text{Primal}} = \underbrace{\min_{u \in R^{M\times N}} \max_{\phi \in (R^{M\times N})^2} (< Ku, \phi > - F^*(\phi) + G(u))}_{\text{Primal-Dual}} \tag{3.8}$$

where ϕ is a dual variable with $\phi \in (R^{M\times N})^2$, F^* is the convex conjugate of F. So, we can rewrite the objective functional 3.1 in this equivalent primal-dual form:

$$\min_{u \in R^{M\times N}} \max_{\phi \in (R^{M\times N})^2} \left(\sum_{i,j} < \nabla u, \phi > + \frac{\sqrt{\beta^2 - \|\phi\|^2}}{\beta} + \frac{\boldsymbol{\lambda}_e}{2}\|u - u_0\|^2 \right) \tag{3.9}$$

After transforming the Beltrami inpainting problem into a primal-dual form, we can iteratively update ϕ and u using the following algorithm:

(a) Raw Depth Map (b) Result by Beltrami (c) Reference by Splatting

Figure 3.2: Analysis of Results by the Baseline Model

Algorithm 1 Primal-dual for Beltrami

Initialize $k = 0$, k_{max}, θ_1, θ_2, λ, β, $u^k = u_0$, $\phi^k = \mathbf{0}$ with u, $u_0 \in R^{M\times N}$, and ϕ, $\bar{\phi}^{k+1} \in (R^{M\times N})^2$
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
$\bar{\phi}^{k+1} \leftarrow (1-\theta_1)\phi^k + \beta\theta_1\nabla u^k / \sqrt{\beta^2 - \|\phi^k\|^2}$
$\phi^{k+1} \leftarrow \beta\bar{\phi}^{k+1} / \max(|\bar{\phi}^{k+1}|, \beta)$
$u^{k+1} \leftarrow (1 - \boldsymbol{\lambda}_e\theta_2)u^k + \theta_2(\mathrm{div}\phi^{k+1} + \boldsymbol{\lambda}_e u_0)$
until convergence of u

Figure 3.2 provides an example of a depth map smoothed using the Beltrami model and a splat-rendered depth map as reference. Upon observing the result, we encounter several issues:

1. There are discontinuous surfaces near the camera. This typically appears when one zooms in closer to the object, leading to a scenario where the projected distances between adjacent points significantly exceed the pixel size. In cases where two consecutive surface layers are projected onto the same image space, points on the back surface can be visible through the gaps or holes in the front surface and ultimately contribute to the final rendered image. During the smoothing process, these points are misinterpreted as being part of the front surface, leading to inaccurate reconstruction.
2. We can see that the staircasing artifacts still remain a challenge.
3. Due to the sparseness of the foreground points, it takes a relatively long time to archive pleasing smoothness on the front surfaces. This leads to the background features being oversmoothed. Therefore, finding a balance between undersmoothing and oversmoothing poses a significant challenge.

To address the misinterpretation of the front surface, we propose two solutions:

a. Pruning the occluded background pixels before iteratively smoothing.

b. During the smoothing process, detect the pixels that cause distortion from the background, and apply adaptive weights for the data fidelity term to effectively ignore these distortion-causing pixels.

To tackle the issue of staircasing artifacts, we propose exploring:

a. Models that incorporate a curvature term.

b. Models that incorporate a normal-related term.

Finally, to balance between undersmoothing and oversmoothing, we propose two solutions:

a. Blending the results from undersmoothed and oversmoothed models.

b. Adopting an adaptive hybrid model.

In the following subsections, we will delve into each of these solutions and discuss how they can help us overcome the challenges we face in reconstructing smooth textured surfaces.

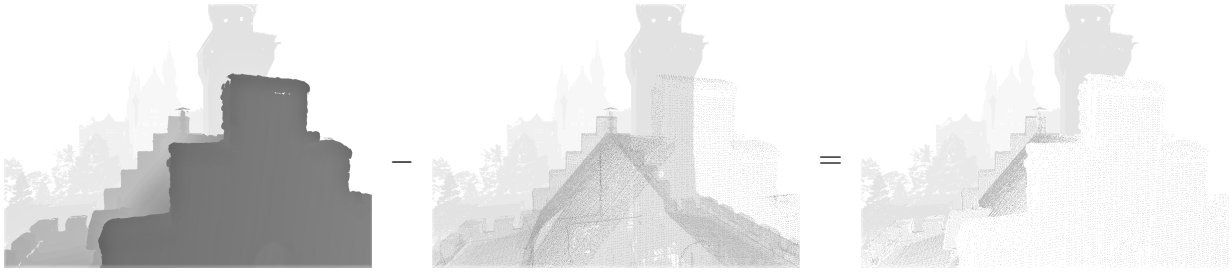
3.1.2 Techniques Against Occluded Points

To address the misinterpretation of the front surface, we can detect these occluded pixels and prune or ignore them. There are two ideas to do this: splatting based method and utilization of an adaptive data fidelity term.

Pruning via Splat-Rendered Depth Comparison

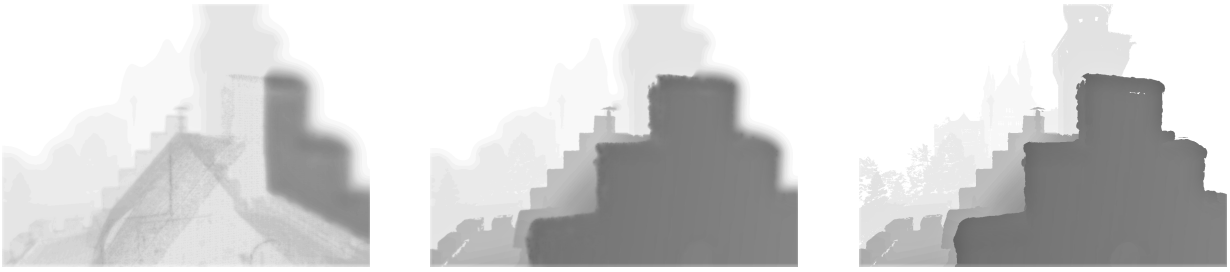
By comparing the raw depth points (Figure 3.3b) with the splat-rendered depth map (Figure 3.3d), points located behind the front surface originally have larger depth values than their pixel value in the splat-rendered depth map. By identifying and removing these occluded pixels, we can prevent them from distorting the smoothing result.

(a) Depth Map by Splatting (b) Raw Depth Map (c) Pruned Depth Map



(d) Depth Map by Splatting (e) Raw Depth Map (f) Pruned Depth Map

Figure 3.3: Pruning via Splat-Rendered Depth Map



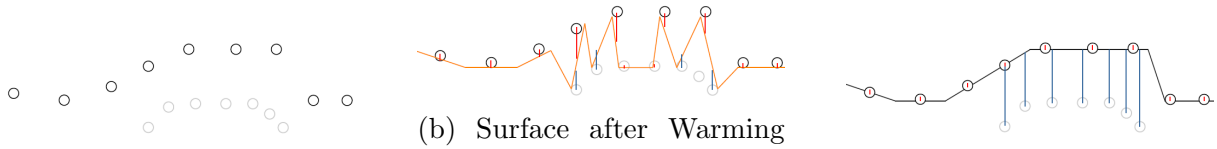
(a) Results without Pruning (b) Results with Pruning (c) Results by Splatting

Figure 3.4: Results with Pruning by Splat-Rendered Depth Map

After pruning, we are left with additional holes to close, as illustrated in Figure 3.3f. The pruned depth map is then smoothed using the TV model as per usual. Although this will take longer, the surface will be smoothed correctly, leading to better reconstruction results, as shown in Figure 3.4b.

Adaptive Weights for Data Fidelity Term

Another solution, to address the issue of sparse points near the camera not covering the surface behind them properly, is using adaptive weights for data fidelity terms. Without any adjustments, after several smoothing iterations, we obtain a fitted surface as shown in the Figure 3.5b.



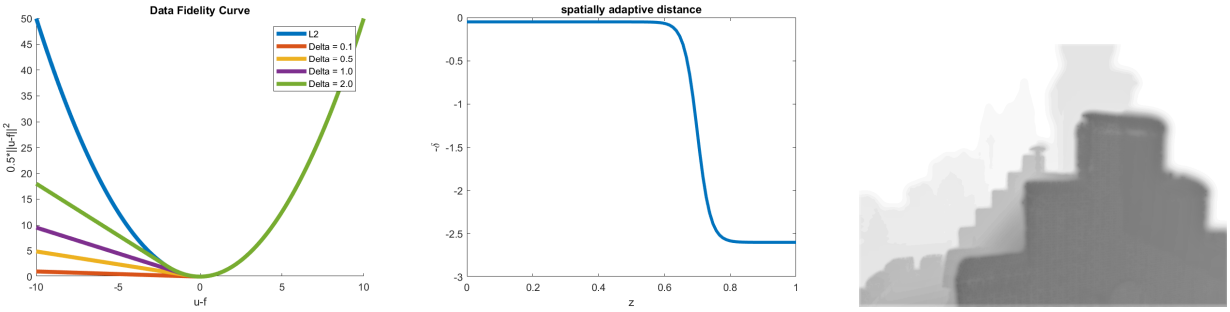
(a) Raw Depth Points (b) Surface after Warming Up (c) Adaptively Fitted Surface

Figure 3.5: Illustration of the Process for Adaptively Weighting the Data Fidelity Term(I)

The orange line represents the fitted surface, black circles are the near points, and gray circles represent the back pixels that cause distortion. We can detect these distortion-causing pixels by comparing their original depth values with the depth value on the fitted surface since they should be behind the surface. Through the comparison, the adaptive data fidelity term $G(u_{i,j})$ is then updated as follows:

$$G(u_{i,j}) = \begin{cases} \frac{\lambda}{2}(u_{i,j} - u_{0_{i,j}})^2, & |u_{i,j} - u_{0_{i,j}}| > -\delta \\ \lambda\delta(|u_{i,j} - u_{0_{i,j}}| - \frac{1}{2}\delta), & \text{else} \end{cases}$$

We assign a very small weight, δ, to the data fidelity term of such pixels. This increases the influence of the variational term on these pixels, allowing them to be more heavily smoothed by their neighboring front pixels. These neighboring pixels are assigned a weight



(a) Data Fidelity Curve (b) Spatially Adaptive δ (c) Adaptively Fitted Surface

Figure 3.6: Illustration of the Process for Adaptively Weighting the Data Fidelity Term(II)

of the data fidelity term that is close to the initial weight. Figure 3.6a illustrates how the data fidelity term changes as the weight varies.

Through this adaptive approach, we can achieve a more accurately reconstructed surface. However, as the smoothing iterations progress, neighboring pixels on another back surface may be influenced by the adaptive weights of the data fidelity term. These pixels tend to smooth out and approach the depth value of the pixels on the front surface edges. This pulling forward of pixels results in poor preservation of edges. Such pulling forward of pixels is particularly pronounced when a small δ value is set. Conversely, a large δ value yields higher runtimes for the process of infilling the foreground surface. To mitigate these issues, a spatially varying δ value is necessary. We can define a spatially varying delta value as:

$$-\delta(z) = -\delta_{min} + (-\delta_{max} - \delta_{min})\frac{1}{1 + e^{-s(z-threshold)}} \tag{3.10}$$

where z represents the pixel value in the depth map, s controls the sharpness of the curve and the *threshold* is a parameter depending on the specific dataset. It is also shown in Figure 3.6b, in which the horizontal axis represents the depth map's pixel value z, while the vertical axis denotes the spatially varying $-\delta(z)$. As points draw closer to the camera (i.e., $z \to 0$), the value of $-\delta$ approaches zero, enabling rapid convergence of closed surfaces. Conversely, when $z \to 1$, $-\delta \to -(\delta_{max})$, preserving the edges in the middle. By leveraging a spatially varying $\delta(z)$, we can achieve a front surface that converges while avoiding the oversmoothness of edges. Figure 3.6c provides an illustrative example of these outcomes.

3.1.3 Curvature-based Regularization Model

To mitigate the staircasing issue, the High-order TV model with a squared l2 data fidelity term (HTVL2 model) serves as our first solution, which employs a curvature regularizer, represented by $|\nabla^2 u|$. The model is defined as:

$$u = \arg\min_u \frac{\boldsymbol{\lambda}_e}{2}\|u - u_0\|^2 + \alpha|\nabla u| + \beta|\nabla^2 u| \tag{3.11}$$

where,

$$|\nabla^2 u| = \sum_{i,j} \sqrt{(\partial_x \partial_x u_{i,j})^2 + (\partial_y \partial_x u_{i,j})^2 + (\partial_x \partial_y u_{i,j})^2 + (\partial_y \partial_y u_{i,j})^2}$$

The curvature term involves the second derivative of the image, meaning that it penalizes changes in the surface curvature across neighboring points, rather than simply smoothing the surface piecewise. While the first-order term favors locally constant patches, the higher-order term allows the model to better capture the curvature of the surface. This strikes a balance between smoothing out the surface and preserving the original geometry, leading to a reduction in staircasing artifacts. Nevertheless, the introduction of the curvature regularizer results in a non-convex optimization problem. Such problems are typically more complex as they often possess multiple local minima, complicating the search for the global minimum or the optimal solution. Moreover, non-convex optimization usually demands more computational resources. For example, the Euler-Lagrange equations associated with the curvature regularizer are fourth-order nonlinear partial differential equations (PDEs). These equations present significant challenges in terms of discretization and computational cost. To solve a high-order variational optimization problem, we employ the Split Bregman method. Originally proposed in [GO09], the Split Bregman algorithm is designed to efficiently solve l1-regularized optimization problems. Notably, it consistently provides relatively optimal solutions for non-convex problems [LDQ+16]. By leveraging the Fast Fourier Transform (FFT) and shrinkage operators, this method is equipped to handle high-order variational models with improved efficiency.

In the subsequent sections, we first provide a brief introduction to the Split Bregman algorithm. Then, we describe the steps involved in utilizing the Split Bregman algorithm to solve the optimization problem for the current HTVL2 model.

Split Bregman Algorithm

The Split Bregman algorithm is employed to convert the complex minimization problem into several subproblems. In the context of solving 11-regularized optimization problems, these subproblems can be efficiently addressed by the FFT, the analytical soft shrinkage equation, and the projection formula without requiring iterative methods. The general form of l1-regularized optimization problems can be expressed as:

$$\min_u |\Phi(u)| + H(u) \tag{3.12}$$

In this equation, $|\Phi(u)|$ denotes the 11 regularizers, and both $H(u)$ and $\Phi(u)$ are assumed to be convex. For instance, in the context of the ROF model (defined by Equation 2.4), $H(u)$ represents the data fidelity term $\frac{\lambda}{2}|u - u_0|^2$ and $|\Phi(u)|$ stands for the first-order variational term $|\nabla u|$. For each l1 norm regularizer, an auxiliary splitting vector variable $\mathbf{w}$ and a Bregman iterative parameter $\mathbf{b}$ are introduced to reformulate the original problem to:

$$(u^{k+1}, \mathbf{w}^{k+1}) = \arg\min_{u,\mathbf{w}} |\mathbf{w}| + H(u) + \frac{\theta}{2}\|\mathbf{w} - \Phi(u) - \mathbf{b}^k\|_2^2 \tag{3.13}$$

In the case of ROF model, we have $u \in \mathbf{R}^{M\times N}$, $\nabla u \in (\mathbf{R}^{M\times N})^2$, $\mathbf{w} \in (\mathbf{R}^{M\times N})^2$ and $\mathbf{b} \in (\mathbf{R}^{M\times N})^2$ to update with the update rate θ. The Equation 3.13 allows us to decouple the l1 and l2 components of the functional 3.12 and then minimize u and $\mathbf{w}$ separately using the following steps:

$$u^{k+1} = \arg\min_u H(u) + \frac{\theta}{2}\|\mathbf{w}^k - \Phi(u) - \mathbf{b}^k\|_2^2 \tag{3.14}$$

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} |\mathbf{w}| + \frac{\theta}{2}\|\mathbf{w} - \Phi(u^{k+1}) - \mathbf{b}^k\|_2^2 \tag{3.15}$$

Lastly, we just need to update b for finishing a complete update iteration with the following step:

$$b^{k+1} = b^k + (\Phi(u^{k+1}) - \mathbf{w}^{k+1}) \tag{3.16}$$

The effectiveness of the Split Bregman method in solving an objective function relies heavily on the efficiency of solving subproblems 3.14 and 3.15. For the differential subproblem 3.14, within the context of TV-based models, we can leverage the FFT to accelerate this sub-optimization process. This concept will later be illustrated using solving subproblem of u for the HTVL2 model as an example. For the subproblem 3.15, since this function is a typical l1-l2 problem and $\mathbf{w}$ has no other coupling elements, we can compute the optimal value of w via soft shrinkage operators. The soft shrinkage operator is defined as follows:

$$shrink(x, \gamma) = \frac{x}{|x|} * \max(|x| - \gamma, 0) \tag{3.17}$$

The shrinkage operator essentially reduces the values of x by γ, but it does not allow values to go below zero. Hence, w can be calculated with the following formula:

$$\mathbf{w}^{k+1} = shrink(\Phi(u) + \mathbf{b}^k, \frac{1}{\theta}) \tag{3.18}$$

Summarizing the above, the Split Bregman algorithm for the optimization problem with a l1 regularizer is presented in Algorithm 2.

Algorithm 2 Split Bregman

Initialize $k = 0$, k_{max}, $\mathbf{w}^k = \mathbf{0}$, $\mathbf{b}^k = \mathbf{0}$, tol and θ
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
$u^{k+1} \leftarrow \arg\min_u H(u) + \frac{\theta}{2}\|\mathbf{w}^k - \Phi(u) - \mathbf{b}^k\|_2^2$
$\mathbf{w}^{k+1} \leftarrow shrink(\Phi(u^{k+1}) + \mathbf{b}^k, \frac{1}{\theta})$
$\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + (\Phi(u^{k+1}) - \mathbf{w}^{k+1})$
$k \leftarrow k + 1$
until convergence of u

In preparation for outlining the algorithms being used to solve subproblems 3.14, a few definitions are necessary. Primarily, it's critical that the definitions of the gradient and Hessian matrix satisfy periodic boundary conditions. By choosing these boundary conditions, each discrete differential operators can be interpreted as a circular convolution of u. This choice allows us to utilize the FFT in the Split Bregman algorithm, greatly enhancing its efficiency in solving subproblems. For more detailed information on this technique, refer to [WT10]. To meet the periodic boundary conditions, we can discretize the gradient and Hessian matrix as follows:

$$(\nabla u) = \left(\partial_x^+ u, \partial_y^+ u\right), \tag{3.19}$$

$$(\nabla^2 u) = \begin{pmatrix} \partial_x^- \partial_x^+ u & \partial_y^+ \partial_x^+ u \\ \partial_x^+ \partial_y^+ u & \partial_y^- \partial_y^+ u \end{pmatrix}, \tag{3.20}$$

The first-order forward differences of u at point (i, j) along x and y directions are defined as:

$$\partial_x^+ u_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j}, & \text{if } 1 \le i \le M,\ 1 \le j < N \\ u_{i,1} - u_{i,j}, & \text{if } 1 \le i \le M,\ j = N \end{cases} \tag{3.21}$$

$$\partial_y^+ u_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j}, & \text{if } 1 \le i < M,\ 1 \le j \le N \\ u_{1,j} - u_{i,j}, & \text{if } i = M,\ 1 \le j \le N \end{cases} \tag{3.22}$$

The first-order backward differences at point (i, j) are given by:

$$\partial_x^- u_{i,j} = \begin{cases} u_{i,j} - u_{i,j-1}, & \text{if } 1 \le i \le M,\ 1 < j \le N \\ u_{i,j} - u_{i,N}, & \text{if } 1 \le i \le M,\ j = 1 \end{cases} \tag{3.23}$$

$$\partial_y^- u_{i,j} = \begin{cases} u_{i,j} - u_{i-1,j}, & \text{if } 1 < i \le M,\ 1 \le j \le N \\ u_{i,j} - u_{M,j}, & \text{if } i = 1,\ 1 \le j \le N \end{cases} \tag{3.24}$$

Deriving from the first-order derivatives, the discrete second-order derivates at point (i, j) are as follows:

$$\partial_x^+ \partial_x^- u_{i,j} = \partial_x^- \partial_x^+ u_{i,j} = \begin{cases} u_{i,N} - 2u_{i,j} + u_{i,j+1}, & \text{if } 1 \le i \le M,\ j = 1 \\ u_{i,j-1} - 2u_{i,j} + u_{i,j+1}, & \text{if } 1 \le i \le M,\ 1 < j < N \\ u_{i,j-1} - 2u_{i,j} + u_{i,1}, & \text{if } 1 \le i \le M,\ j = N \end{cases} \tag{3.25}$$

$$\partial_y^+ \partial_y^- u_{i,j} = \partial_y^- \partial_y^+ u_{i,j} = \begin{cases} u_{M,j} - 2u_{i,j} + u_{i+1,j}, & \text{if } i = 1,\ 1 \le j \le N \\ u_{i-1,j} - 2u_{i,j} + u_{i+1,j}, & \text{if } 1 < i < M,\ 1 \le j \le N \\ u_{i-1,j} - 2u_{i,j} + u_{1,j}, & \text{if } i = M,\ 1 \le j \le N \end{cases} \tag{3.26}$$

$$\partial_x^+ \partial_y^+ u_{i,j} = \partial_y^+ \partial_x^+ u_{i,j} = \begin{cases} u_{i,j} - 2u_{i,j} + u_{i+1,j}, & \text{if } i = 1,\ 1 \le j \le N \\ u_{i-1,j} - 2u_{i,j} + u_{i+1,j}, & \text{if } 1 < i < M,\ 1 \le j \le N \\ u_{i-1,j} - 2u_{i,j} + u_{1,j}, & \text{if } i = M,\ 1 \le j \le N \end{cases} \tag{3.27}$$

Solving Algorithm

In this section, we apply the Split Bregman algorithm (as detailed in Algorithm 2) to solve the objective function for the HTVL2 model. An explanation of how the FFT is incorporated into the solution process is also provided. First, by using Equation 3.13, the objective function 3.11 of the HTVL2 model can be transformed into the following form:

$$E(u,w,v;b,d)=\frac{\boldsymbol{\lambda_e}}{2}\|u-u_0\|^2+\alpha|\mathbf{w}|+\frac{\theta_1}{2}\|\mathbf{w}-\nabla u-\mathbf{b}\|^2+\beta|\mathbf{v}|+\frac{\theta_2}{2}\|\mathbf{v}-\nabla^2 u-\mathbf{d}\|^2 \quad (3.28)$$

In the HTVL2 model, there are two l1 regularizers: $\nabla u \in (\mathbf{R}^{M\times N})^2$ and $\nabla^2 u \in (\mathbf{R}^{M\times N})^4$. Therefore, we need two auxiliary parameters: $\mathbf{w} \in (\mathbf{R}^{M\times N})^2$ and $\mathbf{v} \in (\mathbf{R}^{M\times N})^4$, and two corresponding Bregman iterative parameters: $\mathbf{b} \in (\mathbf{R}^{M\times N})^2$ and $\mathbf{d} \in (\mathbf{R}^{M\times N})^4$. The update rates are controlled by θ_1 and θ_2, respectively. With $(\mathbf{w},\mathbf{v},\mathbf{b},\mathbf{d})$ fixed, we have a differential minimization functional for u as described by Equation 3.14, denoted as $E(u)$. Through $\partial E(u)/\partial u = 0$, its Euler-Lagrange equation can be derived:

$$\boldsymbol{\lambda}_e u+\theta_1\nabla^\intercal\nabla u+\theta_2\nabla^{2^\intercal}\nabla^2 u=\boldsymbol{\lambda}_e u_0+\theta_1\nabla^\intercal\big(\mathbf{w}-\mathbf{b}\big)+\theta_2\nabla^{2^\intercal}\big(\mathbf{v}-\mathbf{d}\big) \quad (3.29)$$

Recognizing that discrete differential operators behave like circular convolutions, we can apply 2-dimensional discrete FFT for efficient resolution of the u subproblem. The equations need to be restructured to make them suitable for FFT processing. The operators on the left side of Equation 3.29 can be explicitly expressed in these forms (Proof provided in Proof 3.1.3):

$$\nabla^\intercal\nabla u=-\mathrm{div}(\nabla u) \;=-(\partial_x^-\partial_x^++\partial_y^-\partial_y^+)u \quad (3.30)$$

$$\nabla^{2^\intercal}\nabla^2 u=\mathrm{div}^2(\nabla^2 u) \;=(\partial_x^-\partial_x^+\partial_x^-\partial_x^+ +2\partial_x^-\partial_x^+\partial_y^-\partial_y^+ +\partial_y^-\partial_y^+\partial_y^-\partial_y^+)u \quad (3.31)$$

$$\mathcal{F}(\partial_x^+(s,r))=\left(\cos\left(\frac{2\pi s}{N}\right)-1+i\sin\left(\frac{2\pi s}{N}\right)\right) \quad (3.32)$$

$$\mathcal{F}(\partial_x^-(s,r))=\left(-\cos\left(\frac{2\pi s}{N}\right)+1+i\sin\left(\frac{2\pi s}{N}\right)\right) \quad (3.33)$$

$$\mathcal{F}(\partial_y^+(s,r))=\left(\cos\left(\frac{2\pi r}{M}\right)-1+i\sin\left(\frac{2\pi r}{M}\right)\right) \quad (3.34)$$

$$\mathcal{F}(\partial_y^-(s,r))=\left(-\cos\left(\frac{2\pi r}{M}\right)+1+i\sin\left(\frac{2\pi r}{M}\right)\right) \quad (3.35)$$

Here, N and M represent the number of columns and rows, respectively. Instead of using i and j (to avoid confusion with i, which represents the imaginary part of a complex number), we use s and r to represent pixel indices. They also denote the frequencies in the discrete frequency domain with $s \in [0,N)$ and $r \in [0,M)$.

Proof. Through the application of the difference mapping property of DFT, we can derive the DFT definition for the forward difference operator as follows:

$\mathcal{F}(\partial_x^+ u(k,l)) = \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} \partial_x^+ u_{m,p} e^{-2\pi i(km+lp)/n}$

$= \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} (u_{m+1,p} - u_{m,p}) e^{-2\pi i(km+lp)/n}$

$= \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} u_{m+1,p} e^{-2\pi i(km+lp)/n} - \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} u_{m,p} e^{-2\pi i(km+lp)/n}$

$= \sum_{m=1}^{n}\sum_{p=0}^{n-1} u_{m,p} e^{-2\pi ik(m-1)/n} e^{-2\pi ilp/n} - \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} u_{m,p} e^{-2\pi i(km+lp)/n}$

$= \left(\cos\left(\frac{2\pi k}{n}\right) + i\sin\left(\frac{2\pi k}{n}\right)\right) \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} u_{m,p} e^{-2\pi i(km+lp)/n} - \sum_{m=0}^{n-1}\sum_{p=0}^{n-1} u_{m,p} e^{-2\pi i(km+lp)/n}$

$= \left(\cos\left(\frac{2\pi k}{n}\right) + i\sin\left(\frac{2\pi k}{n}\right)\right) \mathcal{F}(u(k,l)) - \mathcal{F}(u(k,l))$

This leads us to obtain:

$$\mathcal{F}(\partial_x^+(k,l)) = \left(\cos\left(\frac{2\pi k}{n}\right) - 1 + i\sin\left(\frac{2\pi k}{n}\right)\right)$$

Similarly, we obtain:

$$\mathcal{F}(\partial_x^-(k,l)) = \left(-\cos\left(\frac{2\pi k}{n}\right) + 1 + i\sin\left(\frac{2\pi k}{n}\right)\right)$$
$$\mathcal{F}(\partial_y^+(k,l)) = \left(\cos\left(\frac{2\pi l}{m}\right) - 1 + i\sin\left(\frac{2\pi l}{m}\right)\right)$$
$$\mathcal{F}(\partial_y^-(k,l)) = \left(-\cos\left(\frac{2\pi l}{m}\right) + 1 + i\sin\left(\frac{2\pi l}{m}\right)\right)$$

where n represents the number of columns and k serves as the horizontal index, m denotes the number of rows, and l acts as the vertical index. □

By substituting equations 3.32, 3.33, 3.34 and 3.35 into $\mathcal{F}(\nabla^\intercal\nabla)$ and $\mathcal{F}(\nabla^{2^\intercal}\nabla^2)$, we obtain:

$$\mathcal{F}(\nabla^\intercal\nabla) = -2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) \tag{3.36}$$
$$\mathcal{F}(\nabla^{2^\intercal}\nabla^2) = 4(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2 \tag{3.37}$$

We cannot directly apply FFT to both sides of Equation 3.29 since $\boldsymbol{\lambda}_e$ is not a scalar factor and $\mathcal{F}(\boldsymbol{\lambda}_e u) \neq \boldsymbol{\lambda}_e \circ \mathcal{F}(u)$ [PSS13]. To use FFT for this optimization problem, we introduce another auxiliary parameter $\tilde{u}$ and another Bregman iterative parameter $\tilde{b}$. The equivalent

objective function of Equation 3.28 is then defined as:

$$\begin{aligned} E(u,\tilde{u},w,v;\tilde{b},b,d) = &\frac{\boldsymbol{\lambda}_e}{2}\|u-u_0\|^2 + \frac{\theta_0}{2}\|u-\tilde{u}-\tilde{b}\|^2 \\ &+ \alpha|\mathbf{w}| + \frac{\theta_1}{2}\|\mathbf{w}-\nabla\tilde{u}-\mathbf{b}\|^2 \\ &+ \beta|\mathbf{v}| + \frac{\theta_2}{2}\|\mathbf{v}-\nabla^2\tilde{u}-\mathbf{d}\|^2 \end{aligned} \tag{3.38}$$

The subproblem of u that we aim to solve is now:

$$u = \arg\min_u \frac{\boldsymbol{\lambda}_e}{2}\|u-u_0\|^2 + \frac{\theta_0}{2}\|u-\tilde{u}-\tilde{b}\|^2 \tag{3.39}$$

with the solution of u:

$$u = \frac{2\boldsymbol{\lambda}_e}{2\boldsymbol{\lambda}_e+\theta_0}u_0 + \frac{\theta_0}{2\boldsymbol{\lambda}_e+\theta_0}(\tilde{u}+\tilde{b}) \tag{3.40}$$

With $(u,\mathbf{w},\mathbf{v},\mathbf{b},\mathbf{d})$ fixed, we have a differential minimization function for $\tilde{u}$, $E(\tilde{u})$, as described by Equation 3.38. From this, we derive the Euler-Lagrange equation $\partial E(\tilde{u})/\partial\tilde{u} = 0$. We then apply the FFT to both sides of this Euler-Lagrange equation, yielding:

$$\mathcal{F}\big(\theta_0+\theta_1\nabla^\intercal\nabla+\theta_2\nabla^{2^\intercal}\nabla^2\big)\circ\mathcal{F}(\tilde{u}) = \mathcal{F}\big(\theta_0(u-\tilde{b})+\theta_1\nabla^\intercal(\mathbf{w}-\mathbf{b})+\theta_2\nabla^{2^\intercal}(\mathbf{v}-\mathbf{d})\big) \tag{3.41}$$

where $\circ$ represents an element-wise multiplication. The operators on the right side of Equation 3.41 are explicitly rewritten as:

$$\nabla^\intercal = -\mathrm{div} = -(\partial_x^- + \partial_y^-) \tag{3.42}$$

$$\nabla^{2^\intercal} = \mathrm{div}^2 = \partial_x^-\partial_x^+ + 2\partial_x^-\partial_y^- + \partial_y^-\partial_y^+ \tag{3.43}$$

The solution of $\tilde{u}$ with the current $\mathbf{w}$, $\mathbf{v}$, $\mathbf{b}$, and $\mathbf{d}$, is given by:

$$\tilde{u} = \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big) \tag{3.44}$$

with,

$$\begin{aligned} G &= \theta_0(u-\tilde{b}) + \theta_1\nabla^\intercal(\mathbf{w}-\mathbf{b}) + \theta_2\nabla^{2^\intercal}(\mathbf{v}-\mathbf{d}) \\ \xi &= \theta_0 - 2\theta_1(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2 \end{aligned}$$

where $\mathcal{F}^{-1}$ represents the discrete inverse Fourier transform. $\Re$ is the real part of a complex number. $-$ in Equation 3.44 stands for pointwise division of matrices.

Following this, the updates for $\mathbf{w}$, $\mathbf{v}$, $\mathbf{b}$, and $\mathbf{d}$ are analogous to Equation 3.16 and Equation 3.18. The algorithm for solving the object function for HTVL2 model, including the

Algorithm 3 Split Bregman for HTVL2 Model

Initialize $k = 0$, $\tilde{u}^k = u_0$, $\mathbf{w}^k = \mathbf{0}$, $\mathbf{v}^k = \mathbf{0}$, $\tilde{b}^k = \mathbf{0}$, $\mathbf{b}^k = \mathbf{0}$, $\mathbf{d}^k = \mathbf{0}$, tol, k_{max}, θ_0, θ_1, θ_2
$\xi \leftarrow \theta_0 - 2\theta_1(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2$
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
$u^{k+1} \leftarrow \frac{2\boldsymbol{\lambda}_e^k}{2\boldsymbol{\lambda}_e^k + \theta_0} u_0 + \frac{\theta_0}{2\boldsymbol{\lambda}_e^k + \theta_0}(\tilde{u}^k + \tilde{b}^k)$
$\boldsymbol{\lambda}_e^{k+1} \leftarrow \boldsymbol{\lambda}_e^k$ according to Section 3.1.2
$G \leftarrow \theta_0(u^{k+1} - \tilde{b}^k) + \theta_1 \nabla^\intercal(\mathbf{w}^k - \mathbf{b}^k) + \theta_2 \nabla^{2\intercal}(\mathbf{v}^k - \mathbf{d}^k)$
$\tilde{u}^{k+1} \leftarrow \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big)$
$\mathbf{w}^{k+1} \leftarrow \max\big(|\nabla\tilde{u}^{k+1} + \mathbf{b}^k| - \frac{\alpha}{\theta_1}, 0\big)\frac{\nabla\tilde{u}^{k+1} + \mathbf{b}^k}{|\nabla\tilde{u}^{k+1} + \mathbf{b}^k|}$
$\mathbf{v}^{k+1} \leftarrow \max\big(|\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k| - \frac{\beta}{\theta_2}, 0\big)\frac{\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k}{|\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k|}$
$\tilde{b}^{k+1} \leftarrow \tilde{b}^k + \tilde{u}^{k+1} - u^{k+1}$
$\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + \nabla\tilde{u}^{k+1} - \mathbf{w}^{k+1}$
$\mathbf{d}^{k+1} \leftarrow \mathbf{d}^k + \nabla^2\tilde{u}^{k+1} - \mathbf{v}^{k+1}$
$k \leftarrow k + 1$
until convergence

adaptively reweighting data fidelity method introduced in Section 3.1.2, is summarized in Algorithm 3:

In Figure 3.7, we display an example of smoothing results by the ROF model and the HTVL2 model. Both models incorporate adaptively weighted data fidelity terms. Since we lack a mesh-rendered depth map for this dataset, we use the splats-rendered depth map as the reference for comparison. The example images show that the curvature term can reduce staircasing artifacts while achieving a similar smoothness level on the front surface.

(a) Result by ROF Model (b) Result by HTVL2 model (c) Reference by Splatting

Figure 3.7: Impacts of the Curvature Regularizer

3.1.4 Normal-based Regularization Model

Another approach to reducing staircasing artifacts is to employ a model that incorporates a normal-related regularizer. Alongside the HTVL2 model, the model is defined as:

$$u = \arg\min_{u} \frac{\boldsymbol{\lambda}_e}{2}\|u - u_0\|^2 + \alpha|\nabla u| + \beta|\nabla^2 u| + \gamma|\nabla u - \sigma(N_x, N_y)| \tag{3.45}$$

The additional term, $\gamma|\nabla u - \sigma(N_x, N_y)|$, represents the normal related regularizer, where N_x and N_y denote the x and y component of point normal, respectively. Since the gradient of the depth map represents the rate of change in depth values along the x and y directions, it should maintain consistency with the x and y components of the normal map to ensure that the reconstructed surface properly follows the surface's orientation and structure. This additional term can help prevent staircasing, better preserve edges and detail features, and speed up the smoothing process.

Solving Algorithm

The function defined by Equation 3.45 now incorporates three l1 regularizers $\alpha|\nabla u|$, $\beta|\nabla^2 u|$ and $\gamma|\nabla u - \sigma(N_x, N_y)|$. In addition to the variables $\tilde{u}$ and $\tilde{b}$, we introduce three auxiliary parameters $\mathbf{w}$, $\mathbf{v}$, $\mathbf{t}$, and the corresponding three Bregman iterative parameters $\mathbf{b}$, $\mathbf{d}$, and $\mathbf{e}$, with $\mathbf{w}, \mathbf{t}, \mathbf{b}, \mathbf{e} \in (\mathbf{R}^{M\times N})^2$, and $\mathbf{v}, \mathbf{d} \in (\mathbf{R}^{M\times N})^4$. The step of solving the subproblem of u is the same as shown in Equation 3.40. The Euler-Lagrange equation derived to solve $\tilde{u}$ is:

$$\theta_0\tilde{u}+(\theta_1+\theta_3)\nabla^\intercal\nabla\tilde{u}+\theta_2\nabla^{2^\intercal}\nabla^2\tilde{u} = \theta_0(u-\tilde{b})+\theta_1\nabla^\intercal\big(\mathbf{w}-\mathbf{b}\big)+\theta_2\nabla^{2^\intercal}\big(\mathbf{v}-\mathbf{d}\big)+\theta_3\nabla^\intercal\big(\sigma\mathbf{N}+\mathbf{t}-\mathbf{e}\big) \tag{3.46}$$

And the solution of u with fixed $\tilde{u}$, $\mathbf{w}$, $\mathbf{v}$, $\mathbf{t}$, $\tilde{b}$, $\mathbf{b}$, $\mathbf{d}$ and $\mathbf{e}$, is given as:

$$u = \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big) \tag{3.47}$$

with:

$$\begin{aligned}
\mathbf{N} &= (N_x, N_y) \\
G &= \theta_0(u - \tilde{b}) + \theta_1\nabla^\intercal\big(\mathbf{w} - \mathbf{b}\big) + \theta_2\nabla^{2^\intercal}\big(\mathbf{v} - \mathbf{d}\big) + \theta_3\nabla^\intercal\big(\sigma\mathbf{N} + \mathbf{t} - \mathbf{e}\big) \\
\xi &= \mathcal{F}\big(\theta_0 + (\theta_1 + \theta_3)\nabla^\intercal\nabla + \theta_2\nabla^{2^\intercal}\nabla^2\big) \\
&= \theta_0 - 2(\theta_1 + \theta_3)(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2
\end{aligned}$$

The updates for $\mathbf{w}$, $\mathbf{v}$, $\mathbf{t}$, $\mathbf{b}$, $\mathbf{d}$, and $\mathbf{e}$ are analogous to steps described in Algorithm 3. The algorithm for solving the optimization function of the HTVL2 model with the normal-based regularizer (HTVL2-N model) is summarized in Algorithm 4.

Figure 3.8 presents examples of outcomes generated by different models, comparing those incorporating a normal-related term versus those without it. The model with the normal-related term achieves a similar smoothness level to the model without it but with fewer iterations, less staircasing artifacts, and better preserved edge features.

Algorithm 4 Split Bregman for HTVL2-N Model

Initialize $k = 0$, $\tilde{u}^k = u_0$, $\mathbf{w}^k = \mathbf{0}$, $\mathbf{v}^k = \mathbf{0}$, $\mathbf{t}^k = \mathbf{0}$, $\tilde{b}^k = \mathbf{0}$, $\mathbf{b}^k = \mathbf{0}$, $\mathbf{d}^k = \mathbf{0}$, $\mathbf{e}^k = \mathbf{0}$, tol, θ_0, θ_1, θ_2, θ_3, k_{max}
$\xi \leftarrow \theta_0 - 2(\theta_1 + \theta_3)(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2$
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
$u^{k+1} \leftarrow \frac{2\boldsymbol{\lambda}_e^k}{2\boldsymbol{\lambda}_e^k + \theta_0} u_0 + \frac{\theta_0}{2\boldsymbol{\lambda}_e^k + \theta_0}(\tilde{u}^k + \tilde{b}^k)$
$\boldsymbol{\lambda}_e^{k+1} \leftarrow \boldsymbol{\lambda}_e^k$ according to Section 3.1.2
$G \leftarrow \theta_0(u^{k+1} - \tilde{b}^k) + \theta_1 \nabla^\intercal\big(\mathbf{w}^k - \mathbf{b}^k\big) + \theta_2 \nabla^{2\intercal}\big(\mathbf{v}^k - \mathbf{d}^k\big) + \theta_3 \nabla^\intercal\big(\sigma\mathbf{N} + \mathbf{t}^k - \mathbf{e}^k\big)$
$\tilde{u}^{k+1} \leftarrow \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big)$
$\mathbf{w}^{k+1} \leftarrow \max\big(|\nabla\tilde{u}^{k+1} + \mathbf{b}^k| - \frac{\alpha}{\theta_1}, 0\big)\frac{\nabla\tilde{u}^{k+1} + \mathbf{b}^k}{|\nabla\tilde{u}^{k+1} + \mathbf{b}^k|}$
$\mathbf{v}^{k+1} \leftarrow \max\big(|\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k| - \frac{\beta}{\theta_2}, 0\big)\frac{\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k}{|\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k|}$
$\mathbf{t}^{k+1} \leftarrow \max\big(|\nabla\tilde{u}^{k+1} - \sigma\mathbf{N} + \mathbf{e}^k| - \frac{\gamma}{\theta_3}, 0\big)\frac{\nabla\tilde{u}^{k+1} - \sigma\mathbf{N} + \mathbf{e}^k}{|\nabla\tilde{u}^{k+1} - \sigma\mathbf{N} + \mathbf{e}^k|}$
$\tilde{b}^{k+1} \leftarrow \tilde{b}^k + \tilde{u}^{k+1} - u^{k+1}$
$\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + \nabla\tilde{u}^{k+1} - \mathbf{w}^{k+1}$
$\mathbf{d}^{k+1} \leftarrow \mathbf{d}^k + \nabla^2\tilde{u}^{k+1} - \mathbf{v}^{k+1}$
$\mathbf{e}^{k+1} \leftarrow \mathbf{e}^k + \nabla\tilde{u}^{k+1} - \sigma\mathbf{N} - \mathbf{t}^{k+1}$
$k \leftarrow k + 1$
until convergence of u

(a) Result by HTVL2 Model (b) Result by HTVL2-N (c) Reference by Splatting

Figure 3.8: Impacts of the Normal-based Regularizer

3.1.5 Undersmoothing and Oversmoothing Issues

This section explores two approaches to mediate the issues of undersmoothing and oversmoothing by the infilling process. Firstly, we propose a technique that blends the outcomes of two models: the HTVL1 model with the normal-based regularizer (HTVL1-N model) known for its capability to maintain sharp characteristics and edges, and the HTVL2-N model recognized for its fast convergence of the foreground surface. Subsequently, we introduce an adaptive hybrid HTV-N model as a second method to navigate the trade-off.

Blending

The model with a squared l2 data fidelity can indeed lead to fast convergence, but it may oversmooth detail features, especially pixels that are far from the camera in a depth map. This oversmoothing can result in background fusion, where fine details and texture in the background are lost or blended together. Previous studies in the field of denoising have shown that the TV model with l1 norm for data fidelity term offers some potential advantages over TV model with squared l2 norm for data fidelity term [CP11]. The TV model with a l1 data fidelity term (TV-L1 model) is particularly good at eliminating noise containing strong outliers. The TV model with a squared l2 data fidelity term (TV-L2 model) often leads to over-regularized outcomes, whereas the TV-L1 model successfully removes outliers while maintaining minor details. However, the solution of the TV-L1 model is in general not unique.

There are existing studies that utilize high-order variational models with an l1 data fidelity term for image processing tasks [LHL14]. In the context of handling point-rendered depth map inpainting, our empirical results suggest that using an l1 norm for data fidelity better preserves fine details in the image. However, one trade-off is that models using l1 data fidelity term tend to converge relatively slowly. We propose a blending method that integrates the results from two models. The first model uses a squared l2 data fidelity term, recognized for smoothing foreground surfaces, albeit at the risk of oversmoothing background surfaces. The second model, deploying an l1 data fidelity term, is excellent at preserving detailed features but may undersmooth the foreground surfaces within a reasonable number of iterations. By blending the outcomes of these two models, we aim to balance their respective strengths and mitigate their weaknesses, leading to improved overall results.

The energy function for the HTVL1-N model, as a variant of the HTVL2-N model, is defined as follows:

$$u = \arg\min_{u} \boldsymbol{\lambda}_e|u - u_0| + \alpha|\nabla u| + \beta|\nabla^2 u| + \gamma|\nabla u - \sigma(N_x, N_y)| \tag{3.48}$$

The object function of the HTVL1-N model remains in the l1 regularization class and can be resolved using the Split Bregman algorithm. In this scenario, four auxiliary splitting vector variables, z, $\mathbf{w}$, $\mathbf{v}$, and $\mathbf{t}$, and four Bregman iterative parameters, $\mathbf{b}$, $\mathbf{d}$, $\mathbf{e}$, and c, are introduced. With these variables and parameters fixed, the Euler-Lagrange equation is derived from Equation 3.48 as follows:

$$(\theta_1+\theta_3)\nabla^\intercal\nabla u+\theta_2\nabla^{2^\intercal}\nabla^2 u+\theta_4 u = \theta_1\nabla^\intercal\big(\mathbf{w}-\mathbf{b}\big)+\theta_2\nabla^{2^\intercal}\big(\mathbf{v}-\mathbf{d}\big)+\theta_3\nabla^\intercal\big(\sigma\mathbf{N}+\mathbf{t}-\mathbf{e}\big)+\theta_4(u_0+z-c) \tag{3.49}$$

And the solution of u with fixed $\mathbf{w}$, $\mathbf{v}$, $\mathbf{b}$ and $\mathbf{d}$, derived by Equation 3.49, is given as:

$$u = \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big) \tag{3.50}$$

with:

$$
\begin{aligned}
G &= \theta_1 \nabla^{\intercal}\big(\mathbf{w}-\mathbf{b}\big) + \theta_2 \nabla^{2^{\intercal}}\big(\mathbf{v}-\mathbf{d}\big) + \theta_3 \nabla^{\intercal}\big(\sigma\mathbf{N}+\mathbf{t}-\mathbf{e}\big) + \theta_4(u_0+z-c) \\
\xi &= \mathcal{F}\big((\theta_1+\theta_3)\nabla^{\intercal}\nabla + \theta_2 \nabla^{2^{\intercal}}\nabla^2 + \theta_4\big) \\
&= -2(\theta_1+\theta_3)(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2 - \theta_4
\end{aligned}
$$

The Algorithm to resolve the objective function of the HTVL1-N model is detailed in Algorithm 5:

Algorithm 5 Split Bregman for HTVL1-N Model

Initialize $k=0$, $\mathbf{w}=\mathbf{0}$, $\mathbf{v}=\mathbf{0}$, $\mathbf{t}=\mathbf{0}$, $z=\mathbf{0}$, $\mathbf{b}=\mathbf{0}$, $\mathbf{d}=\mathbf{0}$, $\mathbf{e}=\mathbf{0}$, $c=\mathbf{0}$, tol, θ_1, θ_2, θ_3, k_{max}
$\xi \leftarrow -2(\theta_1+\theta_3)(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2 - \theta_4$
while $|(u^{k+1}-u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
$G \leftarrow \theta_1 \nabla^{\intercal}\big(\mathbf{w}^k-\mathbf{b}^k\big) + \theta_2 \nabla^{2^{\intercal}}\big(\mathbf{v}^k-\mathbf{d}^k\big) + \theta_3 \nabla^{\intercal}\big(\sigma\mathbf{N}+\mathbf{t}^k-\mathbf{e}^k\big) + \theta_4(u_0+z^k-c^k)$
$u^{k+1} \leftarrow \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big)$
$\boldsymbol{\lambda}_e^{k+1} \leftarrow \boldsymbol{\lambda}_e^k$ according to Section 3.1.2
$\mathbf{w}^{k+1} \leftarrow \max\big(|\nabla u^{k+1} + b^k| - \frac{\alpha}{\theta_1}, 0\big)\frac{\nabla u^{k+1}+\mathbf{b}^k}{|\nabla u^{k+1}+\mathbf{b}^k|}$
$\mathbf{v}^{k+1} \leftarrow \max\big(|\nabla^2 u^{k+1} + \mathbf{d}^k| - \frac{\beta}{\theta_2}, 0\big)\frac{\nabla^2 u^{k+1}+\mathbf{d}}{|\nabla^2 u^{k+1}+\mathbf{d}^k|}$
$\mathbf{t}^{k+1} \leftarrow \max\big(|\nabla u^{k+1} - \sigma N + \mathbf{e}^k| - \frac{\gamma}{\theta_3}, 0\big)\frac{\nabla u^{k+1}-\sigma N+\mathbf{e}^k}{|\nabla u^{k+1}-\sigma N+\mathbf{e}^k|}$
$z^{k+1} \leftarrow \max\big(|u^{k+1} - u_0 + c^k| - \frac{\boldsymbol{\lambda}^{k+1}}{\theta_4}, 0\big)\frac{u^{k+1}-u_0+c^k}{|u^{k+1}-u_0+c^k|}$
$\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + \nabla u^{k+1} - \mathbf{w}^{k+1}$
$\mathbf{d}^{k+1} \leftarrow \mathbf{d}^k + \nabla^2 u^{k+1} - \mathbf{v}^{k+1}$
$\mathbf{e}^{k+1} \leftarrow \mathbf{e}^k + \nabla u^{k+1} - \sigma N - t^{k+1}$
$c^{k+1} \leftarrow c^k + u^{k+1} - u_0 - z^{k+1}$
$k \leftarrow k+1$
until convergence of u

Upon solving the outcome of the HTVL1-N model through the Algorithm 5, we subsequently compute a normalized local variance map, denoted as Var_{norm}, by applying the Equation 3.53 to each pixel located at coordinates (x, y). This calculation of Var_{norm} is

made within a predefined local window size n.

$$\mu(x,y) = \frac{1}{n^2}\sum_{i=-n/2}^{n/2}\sum_{j=-n/2}^{n/2} u(x+i,y+j) \tag{3.51}$$

$$Var(x,y) = \left(\frac{1}{n^2}\sum_{i=-n/2}^{n/2}\sum_{j=-n/2}^{n/2} (u(x+i,y+j)-\mu(x,y))^2\right) \tag{3.52}$$

$$Var_{norm} = \frac{Var - Var_{min}}{Var_{max} - Var_{min}} \tag{3.53}$$

Following its calculation, the normalized local variance map, Var_{norm}, is then smoothed using a filter to prevent abrupt transitions on the blended surfaces. Through our tests, we found that using a guided filter, directed by the infilled depth image, generates a satisfyingly smoothed variance map. The Var_{norm} is then employed to derive a weights map $\mathbf{W}$ using the equation as follows:

$$\mathbf{W}^{'} = \frac{1}{\mathbf{1} + \exp\big(-(Var_{norm} - threshold) \times \text{sharp\_level}\big)} \tag{3.54}$$

$$\mathbf{W} = (\mathbf{W}^{'})^{p} \tag{3.55}$$

Our blending process is designed such that pixels in regions with high variance should be contributed more by results from the HTVL2-N model. To achieve our objective, we manipulate the variance values by raising them to a power, denoted as p. This operation amplifies the difference between weights of high and low variance areas in the image. Consequently, regions with high variance become more pronounced, areas of low variance are deemphasized. To mitigate the impact of sharp features, which also present relatively high variance values but should ideally be resolved by the HTVL1-N model, we set a *threshold* to filter out pixels representing detailed features. This weights map subsequently is used in blending the results produced by the HTVL1-N model, represented as u_1, and those by the HTVL2-N model, denoted as u_2, as defined below:

$$u = (\mathbf{1} - \mathbf{W})u_1 + \mathbf{W}u_2 \tag{3.56}$$

This blending strategy is straightforward and effective. However, it's essential to carefully select the parameters to control the weights map. This simple approach can result in abrupt transitions in u. This can be seen in the example shown in Figure 3.9.

Adaptive Hybrid Model

Drawing inspiration from the work [LZ22], we propose an adaptive hybrid model in this section. The model seeks to utilize both the l1 and l2 norms of data fidelity. Specifically, we capitalize on the l1 norm for the preservation of crucial details, while employing the l2 norm for the efficient filling of surface holes. This adaptive hybrid model is defined as:

$$u = \arg\min_{u} \boldsymbol{\lambda}_{e1}|u-u_0| + \frac{\boldsymbol{\lambda}_{e2}}{2}\|u-u_0\|^2 + \alpha|\nabla u| + \beta|\nabla^2 u| + \gamma|\nabla u - \sigma(N_x,N_y)| \tag{3.57}$$

(a) Result by HTVL1-N (b) Result by HTVL2-N (c) Blended Result

Figure 3.9: Illustration of the Blending Process

(a) Result by HTVL1-N (b) Result by HTVL2-N (c) Result by Hybrid Model

Figure 3.10: Impacts of the Hybrid Terms

After a few initial warming-up steps (in this case, we set the number of warming-up steps to three) using the l1 norm data fidelity term weighted by the original $\boldsymbol{\lambda}_{e1}$, we then reweight this term with the weights map derived from Equation 3.55. The algorithm to solve the optimization function of the adaptive hybrid model is described in Algorithm 6:

Figure 3.10 shows an example of the resulting output from the hybrid model. In this case, the smoothed depth map doesn't exhibit the abrupt transition typically seen in the blending method.

3.1.6 A Fast Pipeline for Real Time Application

Due to the numerous holes to be infilled, the iterative optimization process requires between 100 and 150 iterations per model to generate desired outcomes. However, this process is not realistic for real-time applications. To accelerate the process, we propose to add a step before the TV-based smoothing process so that most holes are filled, allowing the TV models to refine the result thereafter.

Here, we employ the advanced Pull-Push algorithm implemented by [Sch23] as the initialization step. This technique conceptualizes the rendered points as 3D splats. These splats, when projected onto the screen, cover an area that depends on several factors: the splat's radius, the distance between the splat and the camera, the splat's orientation, and the camera settings. The pull phase involves recording the nearest projection of a 3D splat in

Algorithm 6 Split Bregman for Hybrid Model

Initialize $k = 0$, $\tilde{u}^k = u_0$, $\mathbf{w}^k = \mathbf{0}$, $\mathbf{v}^k = \mathbf{0}$, $\mathbf{t}^k = \mathbf{0}$, $\mathbf{b}^k = \mathbf{0}$, $\mathbf{d}^k = \mathbf{0}$, $\mathbf{e}^k = \mathbf{0}$, tol, θ_0, θ_1, θ_2, θ_3, θ_4, k_{warmup}, k_{max}
$\xi \leftarrow \theta_0 - 2(\theta_1 + \theta_3)(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2) + 4\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)^2 - \theta_4$
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
  if $k == k_{warmup} + 1$ **then**
    Compute W according to Equation 3.55
    $\boldsymbol{\lambda}_{e1} \leftarrow \lambda_1(1 - \mathbf{W})$
  else
    $u^{k+1} \leftarrow \frac{2\boldsymbol{\lambda}_2^k}{2\boldsymbol{\lambda}_2^k + \theta_0} \cdot u_0 + \frac{\theta_0}{2\boldsymbol{\lambda}_{e2}^k + \theta_0}(\tilde{u}^k + \tilde{b}^k)$
    $\boldsymbol{\lambda}_{e1}^{k+1} \leftarrow \boldsymbol{\lambda}_{e1}^k$ according to Section 3.1.2
    $\boldsymbol{\lambda}_{e2}^{k+1} \leftarrow \boldsymbol{\lambda}_{e2}^k$ according to Section 3.1.2
    $G \leftarrow \theta_0(u^{k+1} - \tilde{b}^k) + \theta_1\nabla^\intercal(\mathbf{w}^k - \mathbf{b}^k) + \theta_2\nabla^{2\intercal}(\mathbf{v}^k - \mathbf{d}^k) + \theta_3\nabla^\intercal(\sigma\mathbf{N} + \mathbf{t}^k - \mathbf{e}^k) + \theta_4(u_0 + z^k - c^k)$
    $\tilde{u}^{k+1} \leftarrow \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big)$
    $\mathbf{w}^{k+1} \leftarrow \max\big(|\nabla\tilde{u}^{k+1} + \mathbf{b}^k| - \frac{\alpha}{\theta_1}, 0\big)\frac{\nabla\tilde{u}^{k+1} + \mathbf{b}^k}{|\nabla\tilde{u}^{k+1} + \mathbf{b}^k|}$
    $\mathbf{v}^{k+1} \leftarrow \max\big(|\nabla^2\tilde{u}^{k+1} + \mathbf{d}^k| - \frac{\beta}{\theta_2}, 0\big)\frac{\nabla^2\tilde{u}^{k+1} + \mathbf{d}}{|\nabla^2\tilde{u}^{k+1} + \mathbf{d}|}$
    $\mathbf{t}^{k+1} \leftarrow \max\big(|\nabla u^{k+1} - \sigma\mathbf{N} + \mathbf{e}^k| - \frac{\gamma}{\theta_3}, 0\big)\frac{\nabla\tilde{u}^{k+1} - \sigma\mathbf{N} + \mathbf{e}}{|\nabla\tilde{u}^{k+1} - \sigma\mathbf{N} + \mathbf{e}^k|}$
    $z^{k+1} \leftarrow \max\big(|\tilde{u}^{k+1} - u_0 + c^k| - \frac{\boldsymbol{\lambda}_{e1}}{\theta_4}, 0\big)\frac{\tilde{u}^{k+1} - u_0 + c^k}{|\tilde{u}^{k+1} - u_0 + c^k|}$
    $\tilde{b}^{k+1} \leftarrow \tilde{b}^k + \tilde{u}^{k+1} - u^{k+1}$
    $\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + \nabla\tilde{u}^{k+1} - \mathbf{w}^{k+1}$
    $\mathbf{d}^{k+1} \leftarrow \mathbf{d}^k + \nabla^2\tilde{u}^{k+1} - \mathbf{v}^{k+1}$
    $\mathbf{e}^{k+1} \leftarrow \mathbf{e}^k + \nabla\tilde{u}^{k+1} - \sigma\mathbf{N} - \mathbf{t}^{k+1}$
    $c^{k+1} \leftarrow c^k + \tilde{u}^{k+1} - u_0 - z^{k+1}$
  end if
  $k \leftarrow k + 1$
until convergence of u

screen space at each level. If multiple splats cover the same area, the one closest to the camera is retained. The subsequent push phase fills in the gaps where values are missing and performs a depth test to replace occluded points. This methodology effectively takes into account the surface orientation while filling in the gaps, thereby resolving point visibility issues. The computational cost of this Pull-Push Algorithm depends solely on the resolution of the input texture, making it a very fast solution. The advanced Pull-Push method results in a depth map with noticeable blocking artifacts on the boundary when the splat size is sufficiently large to fill surface gaps. Its efficiency allows us to fill most surface holes using a smaller splat size, which prevents the formation of blocking artifacts on the boundary.

We continue to use blending techniques to strike a balance between surface smoothness and feature preservation. Unlike the raw point-rendered depth map, this method feeds now different data structures after the advanced Pull-Push process to the TV model. Through experimental analysis, we have discovered that in this case, blending results by the HTVL2-N models with varying balance parameters can yield good reconstructed surfaces with preserved details for real-time inpainting. We use the HTVL2-N model with different balance parameters to smooth the original depth map and post-processed depth map by advanced Pull-Push method, respectively. These balance parameters differ in their level of aggressiveness for each model. The result using the original depth map as input retains the most detailed features but undersmooths foreground surfaces. Conversely, the result using the HTVL2-N model combined with the Pull-Push method can fill surface holes near the camera within 10 optimization iterations. Applying the same blending methodology described in Section 3.1.5, we blend these two outputs to generate the final reconstructed depth map. Compared to our previous method, this combined approach is significantly faster, delivering a smoothed depth map within 20 iterations. Although the real-time reconstructed depth maps may not be as visually appealing as the results demonstrated in previous sections, they are still quite satisfactory for real-time applications. Figure 3.11 exhibits the fast pipeline, complemented by an example of the results generated by this pipeline and instances of intermediate productions within the pipeline.

3.1.7 Reconstruction of Textured Surfaces

In Section 3.1.2, we utilized splat-rendered depth maps to prune occluded points, which helped us prevent incorrect surface interpretations. Having obtained the reconstructed depth map, we can now utilize it to prune the raw normal map and albedo image. Following the pruning, we implement depth map D guided interpolation for both the normal map and albedo image. This allows us to correctly close most holes on the pruned surface. Consider a pixel, denoted by $I(x, y)$, within the image I, for which interpolation is required. If it holds a valid depth value $D(x, y)$, we search for valid neighboring pixels within its local window Ω of size n. A neighboring pixel is marked as valid if it carries a valid value in $I(i, j)$ and has a depth value similar to the original pixel. This similarity is assessed based on the condition $|D(x, y) - D(i, j)| < threshold$. Upon verification, this neighboring

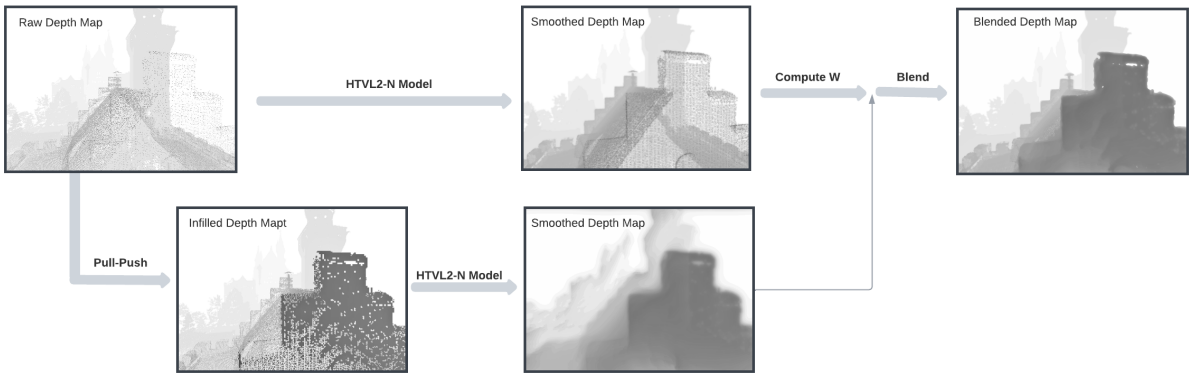
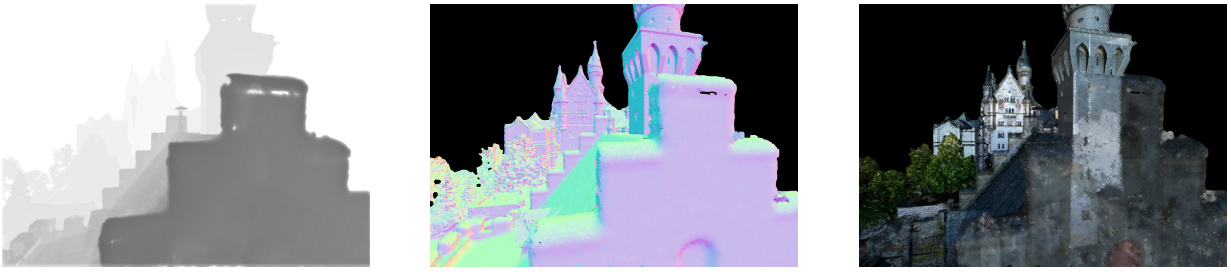


Figure 3.11: The flowchart illustrates our proposed fast pipeline for smoothing a point-rendered depth map. Initially, we use the HTVL2 model with different balance parameters to smooth the original depth map and the depth map that has been post-processed with the advanced Pull-Push method, respectively. These balance parameters differ in their level of aggressiveness for each model. Subsequently, we blend the outputs from both these stages to generate the final, reconstructed depth map.

pixel is added to the valid neighboring set Ω_v. Additionally, we assign it a weight $w_{i,j} = 1/(|D(x,y) - D(i,j)| + eps)$. The weights of all valid neighboring pixels are then normalized. The final interpolated pixel value is then computed with: $I(x,y) = \frac{1}{\sum_{\Omega_v} w_{i,j}} \sum_{\Omega_v} w_{i,j} I(i,j)$ The algorithm used for interpolation is presented in Algorithm 7.

Using the interpolated normal map or albedo image as input for the next smoothing step, the final reconstructed texture images are obtained by applying the TV model for each color channel of the interpolated images separately. Here, we show an example of the interpolated normal map and textured image guided by the smoothed depth map in Figure 3.12.



(a) Smoothed Depth Map (b) Interpolated Normal Map (c) Interpolated Albedo

Figure 3.12: Infilled Textured Surfaces

We summarize the pipeline for smoothing point rendered textured surfaces: We start with a raw depth map, generated by point-based rendering. This map is smoothed and infilled.

Algorithm 7 Depth Map Guided Interpolation

Inputs: I, D, $window\_size$, $threshold$
Output: $I_{interpolated}$
for $I(x, y)$ in I **do**
  if $I(x, y)$ invalid $\wedge$ $D(x, y)$ valid **then**
    $w \leftarrow 0$
    Define Ω by $window\_size$
    Initial $I_{interpolated}(x, y, :) \leftarrow 0$
    for $I(i, j)$ in Ω **do**
      if $D(i, j)$ valid $\wedge$ $|D(x, y) - D(i, j)| < threshold$ **then**
        Compute $w_{i,j} \leftarrow 1/(|D(x, y) - D(i, j)| + eps)$
        Compute $w \leftarrow w + w_{i,j}$
        Compute $I_{interpolated}(x, y, :) \leftarrow I_{interpolated}(x, y, :) + w_{i,j} I(i, j, :)$
      end if
    end for
    Compute $I_{interpolated}(x, y, :) \leftarrow I_{interpolated}(x, y, :)/w$
  else
    Compute $I_{interpolated}(x, y, :) \leftarrow I(x, y, :)$
  end if
end for

This step may use a hybrid model or blend results from HTVL1-N and HTVL2-N models, both of which strive to smooth discontinuous surfaces while preserving detailed features. Alternatively, a combination of TV models and the advanced Pull-Push method can be used for fast smoothing. Subsequently, we use the smoothed depth map to prune the raw normal map and albedo image. The pruned map and image are then interpolated, under the guidance of the smoothed depth map, to fill in most of the remaining gaps. Typically, the interpolated normal map has minimal holes after this process. Finally, we apply TV model smoothing to the interpolated albedo images, handling each color channel separately. This results in the final, smoothed image output. The summary involves three different approaches to smoothing raw depth maps, the whole pipeline with all of which is illuminated in Figure 3.13.

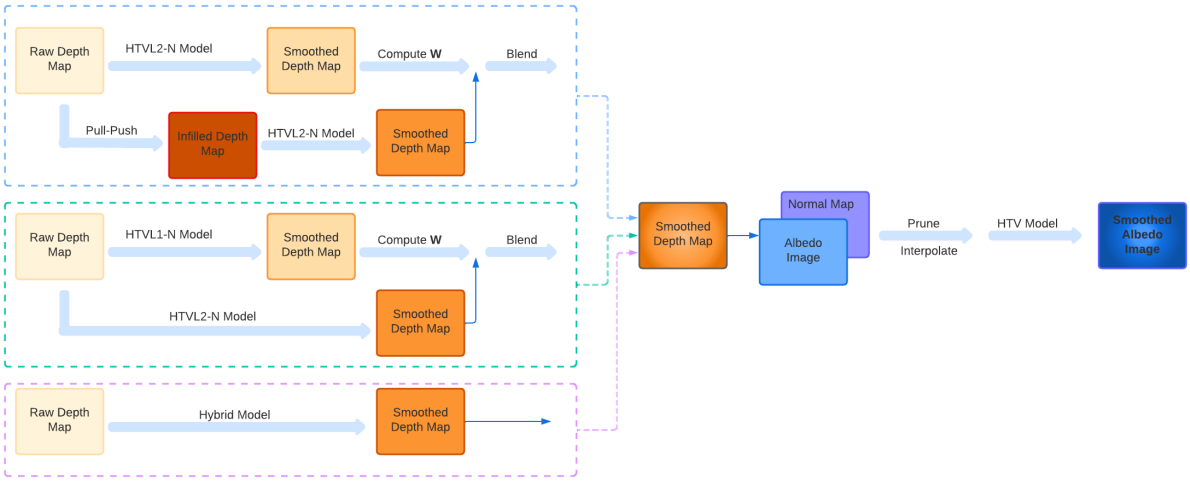


Figure 3.13: This flowchart illustrates the proposed pipeline for smoothing point-rendered textured surfaces. It outlines three alternate strategies for smoothing raw depth maps.

3.2 TV-based Methods for Denoising Monte Carlo Rendering

In this section, we repurpose our proposed models for denoising Monte Carlo rendering. As done in the state of the art work, we initiate by demodulating the noisy rendered images and then implement temporal accumulation through the reprojection method. Subsequently, we employ our proposed TV model to denoise the preprocessed images. Two critical aspects of this approach involve utilizing auxiliary features and concerning Monte Carlo noise to adaptively adjust the balance parameters of TV model. The following sections briefly introduce the demodulation and temporal accumulation steps. The methodology for setting adaptive parameters is explained in detail.

3.2.1 Demodulation

In the context of Monte Carlo denoising, the presence of high-frequency components within the first-hit albedo often poses significant challenges for the denoising process. These high frequencies, primarily due to detailed texture or fine geometry, can introduce noise artifacts, complicate noise estimation, and disrupt the efficiency and accuracy of the denoising algorithms. Demodulation is introduced as a preprocessing step to address this issue, wherein the pixel color is divided by the albedo. This operation seeks to separate the image content into a high-frequency albedo component and a smooth lighting component. Conceptually, this is denoted as:

$$\text{Light} = \frac{\text{Pixel Color}}{\text{Albedo}} \tag{3.58}$$

The above equation is derived from a simplified representation of the rendering equation, which is given by:

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_\Omega A(p)F(\omega_i, \omega_o, n)L_i(p, \omega_i)(\omega_i \cdot n)d\omega_i \tag{3.59}$$

In this equation, $L_o(p, \omega_o)$ represents the outgoing radiance at a point p in direction ω_o, $L_e(p, \omega_o)$ is the emitted radiance at the point p in direction ω_o, and $F(\omega_i, \omega_o, n)$ is a function that depends on the light, view directions and the surface normal. In this simplified rendering equation 3.59, the albedo map corresponds to the albedo $A(p)$, and the normal map corresponds to the surface normal n. Then, all the denoising steps are performed on light components. Finally, the remodulation step reintroduces the albedo information into the denoised result.

3.2.2 Temporal Accumulation

The inherent stochastic nature of Monte Carlo rendering, which involves random sampling of the scene's light paths, leads to the generation of noise. Temporal accumulation is a strategy employed to minimize this noise by accumulating samples across multiple frames. Because the average of several independent, identically-distributed random samples tends to converge toward the expected value of these samples, temporal accumulation can reduce noise effectively. By accumulating samples across multiple frames for the noisy 1 spp data, the effective spp can be greater than 1. This helps reduce the variance and subsequently, the noise, enhancing temporal stability.

The process of temporal accumulation via reprojection in this work is outlined as follows:

1. Each pixel from the current frame is projected into world space using image space coordinates, the depth map, and camera projection, rotation, and position matrices.
2. These 3D coordinates are then reprojected into the image space of the previous frame using its camera matrices. This operation uses the constraints of the normal map and world-space positions to identify the corresponding pixel from the previous frame.
3. The noise reduction is achieved by calculating a weighted sum of pixel information from both frames.

3.2.3 Utilization of Auxiliary Features

Auxiliary features provide better geometric information that can be leveraged to preserve edges and prevent over-smoothing in the images. We compute a coherence map for the depth map, normal map, and albedo texture, respectively. For each pixel, we identify the maximum value across these maps. Subsequently, we employ the exponential function to

map this value into the range of $[0, 1]$. This forms the final coherence map, denoted as $\mathbf{W}$:

$$\bar{\mathbf{W}} = \max\{\mathbf{W}_{albedo}, \mathbf{W}_{depth}, \mathbf{W}_{normal}\} \tag{3.60}$$

$$\mathbf{W} = \exp(-s_A \bar{\mathbf{W}}) \tag{3.61}$$

Each coherence map is derived by taking the normalized second-order image gradient for each color channel so that important features are assigned larger values compared to pixels on smooth surfaces. The scalar factor s_A dictates the sensitivity of the weight α with respect to the values in the coherence map $\mathbf{W}$. A larger s_A implies that regions with high coherence will have an aggressive influence on the TV term, resulting in more intense preservation of these regions. In contrast, regions with low coherence will contribute less influence on the variational term, allowing more changes during the denoising process. However, an excessively large s_A may lead to edge artifacts. On the other hand, a smaller s_A makes the influence of the coherence map W less pronounced. This might result in a more uniformly denoised image but also means that important details in high-coherence areas are not preserved. The final coherence map is then used to weight the variational term in a TV model. For instance, adapting the ROF model, the optimization function is now defined as:

$$f(u) = \frac{\lambda}{2}\|u - u_0\|^2 + \boldsymbol{\alpha}|\nabla u| \tag{3.62}$$

where $\boldsymbol{\alpha}$ now varies spatially, with:

$$\boldsymbol{\alpha} = \alpha_0 \mathbf{W} \tag{3.63}$$

3.2.4 Hybrid Denoising Model

Similar as described in the inpainting, the type of data fidelity terms holds a significant impact on the denoising results. In the Monte Carlo denoising tasks, both l1 and squared l2 data fidelity terms come with their respective strengths and limitations. The squared l2 data fidelity is particularly effective in minimizing large errors. However, due to its inherent sensitivity to outliers and its quadratic penalty on large errors, it tends to excessively reduce the magnitude of errors. As a consequence, the inherent smoothing effect of the squared l2 norm can lead to a blurring of sharp features in the denoised images. On the other hand, the l1 data fidelity penalizes the absolute difference between the denoised and original pixel values linearly. This makes it more robust against outliers compared to the squared l2 norm. Edges or features like shadows and other high-frequency features that are often viewed as "outliers" in an image can be better preserved by using the l1 norm than the squared l2 norm. The squared l2 norm over-penalizes outliers due to its quadratic error penalty and consequently causes loss of details. Conversely, the l1 norm doesn't over-penalize outliers, maintaining a balance between noise removal and feature preservation. However, in terms of noise suppression efficiency, the l1 norm may not always perform as well as the squared l2 norm

Furthermore, in Monte Carlo rendering, noise often manifests as darker pixels in dark areas. If the dark noise pixels significantly outnumber the correctly exposed pixels, then

the median pixel value will be darker. In such a scenario, an model with l1 data fidelity term could indeed result in darker output in these areas.

In light of these observations, we propose a hybrid model that take the strengths of both forms of data fidelity, similar to the model proposed in Section 3.1.5. The hybrid denoising model is defined as:

$$u = \arg\min_{u} \lambda_1 |u - u_0| + \frac{\lambda_2}{2}\|u - u_0\|^2 + \boldsymbol{\alpha}|\nabla u| \tag{3.64}$$

where $\boldsymbol{\alpha}$ is computed as described in Section 3.2.3, λ_1 and λ_2 are scale balance parameters. Since λ_2 is a constant scale, no additional auxiliary parameter is needed to be introduced to solve the optimization problem for the denoising model. The algorithm to solve the optimization problem defined by Equation 3.64 is summarized in Algorithm 8.

Algorithm 8 Split Bregman for Adaptive Hybrid Denoising Model

Initialize $k = 0$, $\mathbf{w}^k = \mathbf{0}$, $\mathbf{b}^k = \mathbf{0}$, $z^k = 0$, $c^k = 0$, tol, θ_1, θ_2, k_{max}, λ_1, λ_2, α_0
$\mathbf{W} \leftarrow$ According to Equation 3.60
$\boldsymbol{\alpha} \leftarrow \alpha_0 \exp(-s\mathbf{W})$
$\xi \leftarrow \lambda_2 - \theta_1 - 2\theta_2(\cos\frac{2\pi s}{N} + \cos\frac{2\pi r}{M} - 2)$
while $|(u^{k+1} - u^k)/u^k| < tol$ && $k < k_{max}$ **do**
repeat
  $G \leftarrow \lambda_2 u^{k+1} + \theta_1(u_0 + z^k - c^k) + \theta_2 \nabla^\intercal(\mathbf{w}^k - \mathbf{b}^k)$
  $u^{k+1} \leftarrow \Re\big(\mathcal{F}^{-1}\big(\frac{\mathcal{F}(G)}{\xi}\big)\big)$
  $z^{k+1} \leftarrow \max\big(|u^{k+1} - u_0 + c^k| - \frac{\lambda_1}{\theta_1}, 0\big)\frac{u^{k+1} - u_0 + c^k}{|u^{k+1} - u_0 + c^k|}$
  $\mathbf{w}^{k+1} \leftarrow \max\big(|\nabla u^{k+1} + \mathbf{b}^k| - \frac{\boldsymbol{\alpha}}{\theta_2}, 0\big)\frac{\nabla u^{k+1} + \mathbf{b}^k}{|\nabla u^{k+1} + \mathbf{b}^k|}$
  $c^{k+1} \leftarrow c^k + u^{k+1} - u_0 - z^{k+1}$
  $\mathbf{b}^{k+1} \leftarrow \mathbf{b}^k + \nabla u^{k+1} - \mathbf{w}^{k+1}$
  $k \leftarrow k + 1$
until convergence of u

3.2.5 Noise Estimation

A noise estimation matrix $\mathbf{W}_{Noise}$ is a quantitative representation of the level of noise present in an image, with higher values in the matrix indicating a higher likelihood of noise. In this section, we derive the noise estimation matrix either by Median Absolute Deviation (MAD) or spatiotemporal variance adapted from [SKW+17]. The noise estimation matrix, combined with auxiliary features, is then used to compute a new and improved coherence map, which is employed to adaptively weight the TV regularization terms, as given by:

$$\mathbf{W} = \max\{\mathbf{W}_{Auxillary}, \mathbf{W}_{Noise}\} \tag{3.65}$$

where,

$$\mathbf{W}_{Noise} = 1 - \exp(-s_E \bar{W}_N) \tag{3.66}$$

$\mathbf{W}_{Auxillary}$ is computed as described in Section 3.2.3. $\mathbf{W}_{Noise}$ and $\bar{W}_N$ are matrices with each entry (i, j) corresponding to a pixel in the image. $\bar{W}_N(i, j)$ is the value of MAD or spatiotemporal variance, and $\mathbf{W}_{Noise}(i, j)$ lies in the interval $[0, 1]$, A higher value of $\bar{W}_N(i, j)$ indicates that the pixel is more likely to be noisy. By applying these weights, we can enforce the TV regularization to have a stronger influence in areas of the image where the noise level is estimated to be higher and a weaker influence in areas where the noise level is estimated to be lower. This approach allows us to more effectively reduce noise, particularly in noisy regions with sharp geometric characteristics. The following sections explain the MAD and spatiotemporal variance used in our method in detail.

MAD

The MAD [DJ94] operates under the assumption that noise remains stationary within a small pixel window, denoted as w, surrounding the observed pixel. The mathematical expression for MAD is given as:

$$\sigma_w(p) = \frac{median(|D_0^1|)}{0.6745} \tag{3.67}$$

In this equation, $|D_0^1|$ corresponds to the diagonal detail coefficients at the highest level of the wavelet transformation. The wavelet transform enables the decomposition of an image or signals into different frequency sub-bands and scales. These diagonal detail coefficients at the finest level of the wavelet transform represent the smallest-scale, highest-frequency details in the image, which are typically representative of noise. The constant 0.6745 is a scaling factor that has been empirically determined to obtain optimal results. $\sigma_w(p)$ represents the estimated noise standard deviation for the window centered on the pixel p.

Spatiotemporal Variance

In this section, we determine the variance of color luminance for each pixel, which then serves as a guide to locally adjust the weights of TV terms. We employ temporally estimated variance, leveraging information from different temporal samples, as a more reliable noise estimator compared to spatial variance. A high value of spatiotemporal variance generally indicates a greater likelihood of noise in pixel values.

Firstly, the first raw moment, μ_{1_i} is calculated using the samples from a small window W around pixel i as follows:

$$\mu_{1_i} = \frac{1}{|W|} \sum_{j \in W} L_j \tag{3.68}$$

where L_j represents the luminance of the j^{th} pixel in the window W and $|W|$ is the number of pixels in this window. The second raw moment, μ_{2_i}, is similarly calculated as:

$$\mu_{2_i} = \frac{1}{|W|} \sum_{j \in W} (L_j)^2 \tag{3.69}$$

Here, $|W|$ represents the number of pixels in the window, and the summation is performed over the luminance values of all pixels in this window. These moments are then temporally accumulated to generate integrated moments μ'_{1_i} and μ'_{2_i}, using $\mu'_{1_i} = \alpha\mu'_{1_{i-1}} + (1-\alpha)\mu_{1_i}$ and $\mu'_{2_i} = \alpha\mu'_{2_{i-1}} + (1-\alpha)\mu_{2_i}$. However, when the geometric constraints discussed in Section 3.2.2 are not met, we opt for raw moments in instead of integrated ones. The temporal variance is then determined from the integrated moments with $\sigma_i'^2 = \mu'_{2_i} - \mu'^{\,2}_{1_i}$. In scenarios where temporal history is scarce, we alternatively estimate the variance $\sigma_i'^2$ spatially. It should be noted that a low variance doesn't necessarily imply an absence of noise; it could merely suggest consistent noise across space and time.

Lastly, we summarize the denoising pipeline: The process begins with demodulation, which is employed to eliminate the object's inherent color, thereby isolating the lighting information. Subsequently, we utilize reprojection to temporally accumulate samples. Here, we also apply a logarithmic operation to the image's light components to aid our smoother in efficiently dealing with the noisy components saved in High Dynamic Range (HDR) format. After these preprocessing steps, we utilize auxiliary features and a noise estimation matrix, calculated either through the MAD or the spatiotemporal variance, to adapt the weights of the variational terms of the TV model. With these steps completed, we feed the noisy light components into our TV model for denoising. The final step of the process involves performing an inverse logarithmic operation and remodulation to reintroduce the original texture into the denoised results.

Chapter 4

Experiments and Evaluation

In this chapter, the results by our proposed methods for two research problems are evaluated. We compare our results to results by the state of the art methods. A discussion and interpretation of these results in the context of our research questions are also made. In this chapter, we present a comprehensive evaluation of the results obtained using our proposed methods for the task of smoothing textured surfaces rendered by point-based methods (Section 4.1) and the task of reducing noise in Monte Carlo rendered images (Section 4.2). To validate the effectiveness of our approaches, we compare our results against those obtained by state-of-the-art methods. For the denoising task we conducted ablation studies to assess the impact of our innovative contributions. Throughout the chapter, we discuss and interpret these results within the context of our research objectives.

4.1 Smoothing Point-Rendered Images

In Section 3.1, we proposed three alternatives for smoothing textured surfaces: the blending method (see Section 3.1.5), hybrid model (see Section 3.1.5), and a fast pipeline combined with the Pull-Push algorithm (see Section 3.1.6). We now turn our attention to a comparative analysis of our proposed methods against established point cloud rendering techniques. Specifically, we compare our results to the High-Quality-Splatting (HQS) method and the Point-to-Pixel (PP) rendering method, the latter of which incorporates the advanced Pull-Push Algorithm as a crucial post-processing step to fill in the holes in its raw output. Both of these methods have been implemented by [Sch23]. The comparison focuses primarily on the subjective visual quality produced by each approach.

Figures 4.1, 4.2, 4.3, and 4.4 display comparisons of our results with those from different rendering methods across four datasets: Baker and the Bridge, Walkman, Stanford Dragon, and Neuschwanstein. The renderings of these four datasets, produced by the other methods, are sourced from [Sch23]. The first three datasets, originally in mesh format, are sampled into point clouds and rendered for evaluation. Their mesh-rendered images are

served as references. The Neuschwanstein dataset is large raw point cloud data, and therefore mesh-rendered images are not provided for comparison. We compare the depth map, normal map, and albedo images filled in by our methods with images rendered by HQS and PP methods. To better assess our methods, we present reconstructed images from both distant and close-up views of the point cloud. For images rendered from a distant perspective, which display fewer discontinuous surfaces, we employ our blending method and hybrid model. For surfaces with less non-smoothness, these strategies achieve satisfactory results within 50 iterations. On the other hand, images rendered from a close-up view are smoothed by our fast pipeline. This method is particularly good at filling a large number of holes on surfaces near to the camera quickly with preserved geometric features, typically within just 20 iterations.

Primarily, we successfully infilled most holes. For images rendered from a distant view, we achieved exceptionally smooth surfaces while preserving detailed features, often resulting in outcomes comparable to those qualitative results from HQS. For images rendered from a very close perspective, our fast pipeline managed to smooth even highly discontinuous surfaces. However, this success is heavily dependent on the output of the preceding Pull-Push step. Secondly, while both HQS and the advanced Pull-Push Algorithm possibly introduce artifacts along boundaries, our method exhibits no such significant boundary artifacts. This advantage is particularly evident when dealing with data of simple geometry (like the Walkman and Stanford Dragon datasets) or when images are rendered from a close perspective. Additionally, Our method, while currently implemented in MATLAB as an offline approach, shows promise for real-time applications. The straightforward nature of the implementation, combined with satisfactory results achieved within just 20 iterations using our fast pipeline, suggests that a GPU-based implementation could yield processing times close to 10 ms per frame. This outcome would fulfill the needs of real-time point-based rendering applications.

However, owing to the characteristics of the TV term, edges in a depth map are prone to blend into the background. The blending technique we proposed mitigates this over-blurring of edges to a certain extent, the infilled depth map may still exhibit blurred boundaries. These artifacts are consequently carried over into the reconstructed normal map and albedo images since they are initially interpolated with guidance from the infilled depth map. Furthermore, due to the nature of a TV-based model, where its energy function doesn't focus on preserving total energy, a consistent outcome is a reduction in image contrast following the smoothing process.

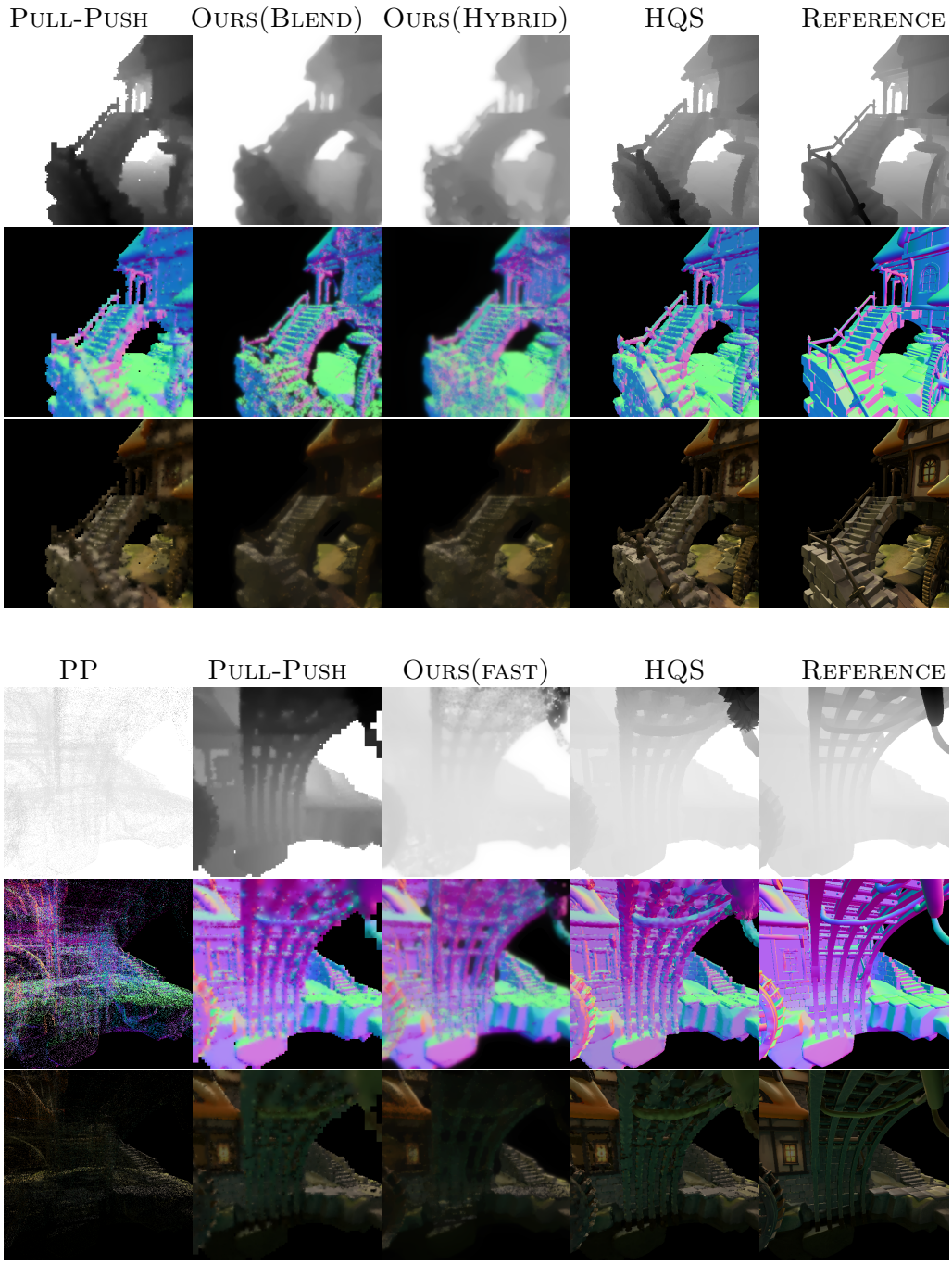


Figure 4.1: This figure displays the comparison for the Baker-Bridges dataset. Two images have been selected, one rendered from a distant view and the other from a close view. The first three rows of images from top to bottom represent the depth map, normal map, and albedo image of the distant view. From left to right, the columns depict results by PP rendering with Pull-Push, our blending method, our hybrid method, HQS rendering, and the mesh-rendered reference. The next three rows show the depth map, normal map, and albedo image from top to bottom of the close view. From left to right, the columns represent results by PP rendering, PP rendering with Pull-Push, our fast pipeline method, HQS rendering, and the mesh-rendered reference

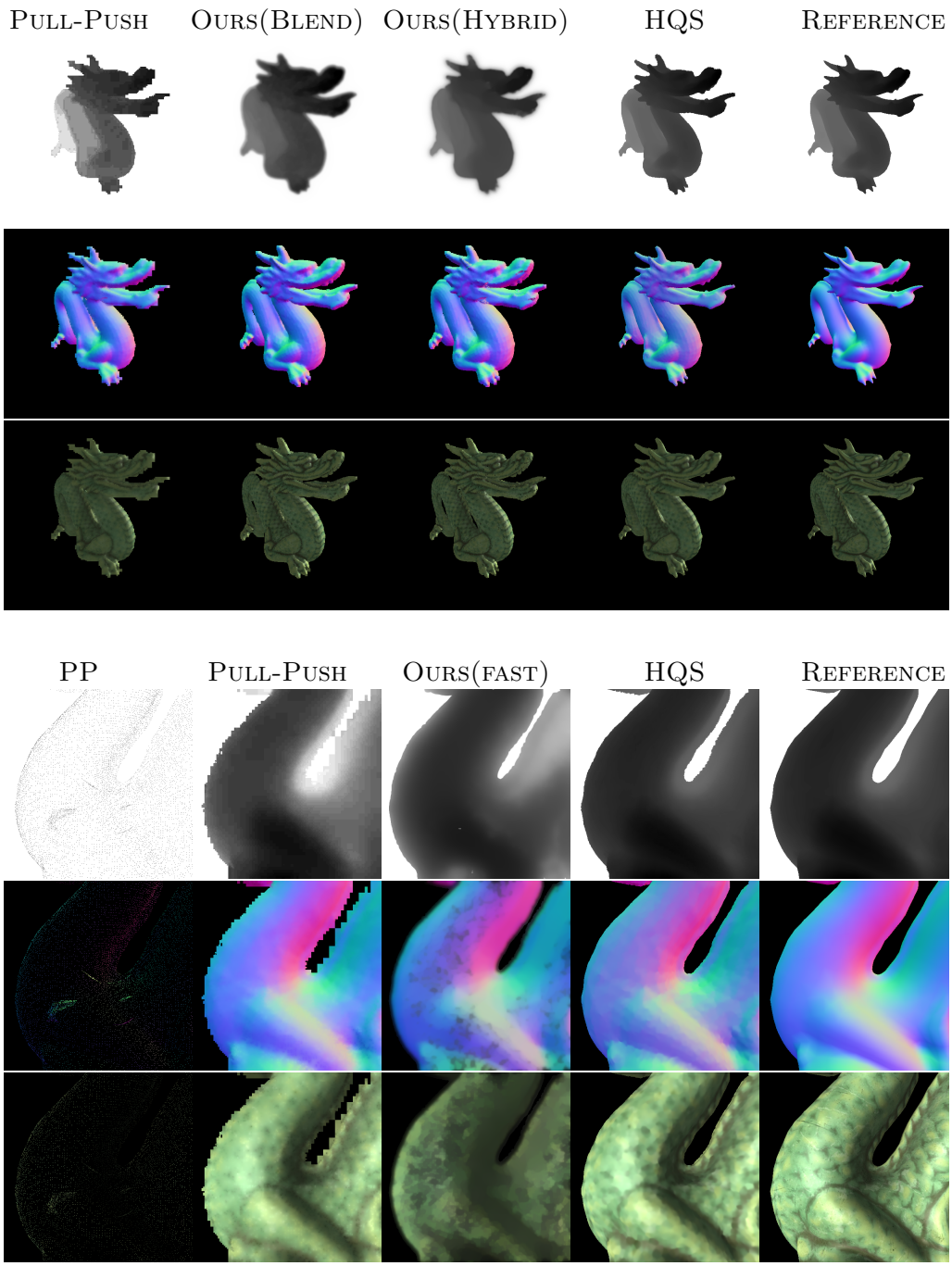


Figure 4.3: This figure displays the comparison for the Stanford Dragon dataset. Two images have been selected, one rendered from a distant view and the other from a close view. The first three rows of images from top to bottom represent the depth map, normal map, and albedo image of the distant view. From left to right, the columns depict results by PP rendering with Pull-Push, our blending method, our hybrid method, HQS rendering, and the mesh-rendered reference. The next three rows show the depth map, normal map, and albedo image from top to bottom of the close view. From left to right, the columns represent results by PP rendering, PP rendering with Pull-Push, our fast pipeline method, HQS rendering, and the mesh-rendered reference

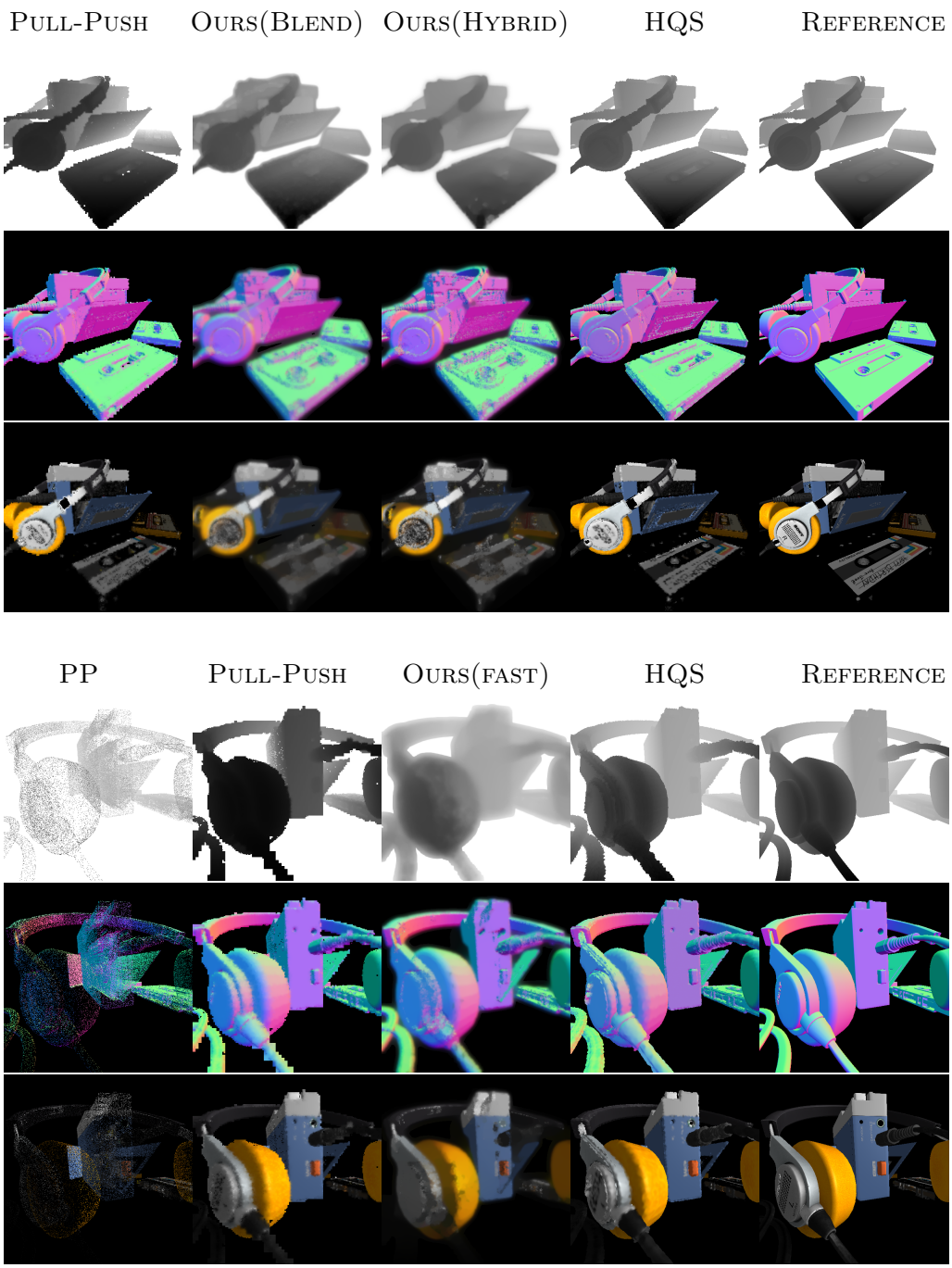


Figure 4.2: This figure displays the comparison for the Walkman dataset. Two images have been selected, one rendered from a distant view and the other from a close view. The first three rows of images from top to bottom represent the depth map, normal map, and albedo image of the distant view. From left to right, the columns depict results by PP rendering with Pull-Push, our blending method, our hybrid method, HQS rendering, and the mesh-rendered reference. The next three rows show the depth map, normal map, and albedo image from top to bottom of the close view. From left to right, the columns represent results by PP rendering, PP rendering with Pull-Push, our fast pipeline method, HQS rendering, and the mesh-rendered reference

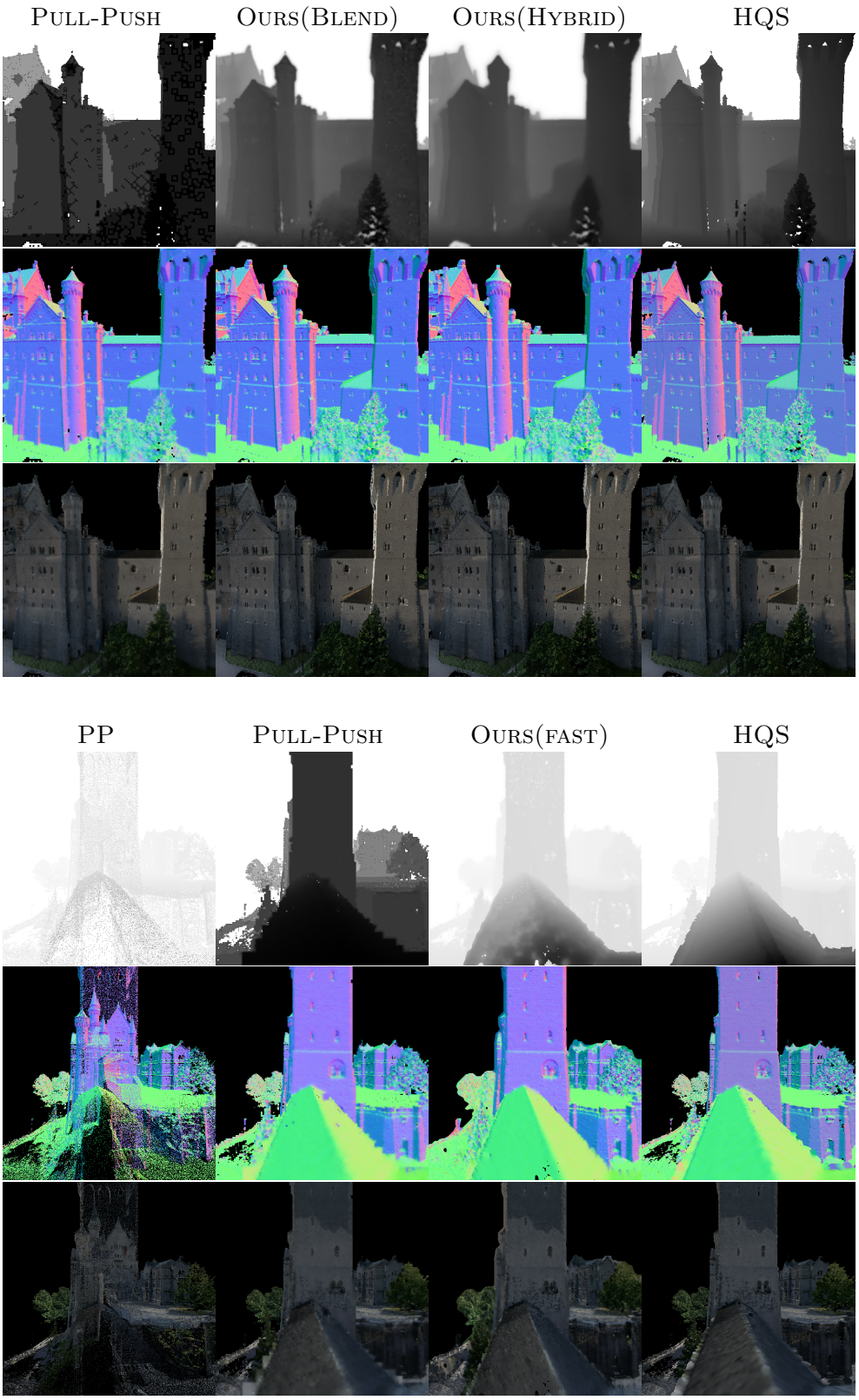


Figure 4.4: This figure displays the comparison for the Neuschwanstein dataset. Two images have been selected, one rendered from a distant view and the other from a close view. The first three rows of images from top to bottom represent the depth map, normal map, and albedo image of the distant view. From left to right, the columns depict results by PP rendering with Pull-Push, our blending method, our hybrid method and HQS rendering. The next three rows show the depth map, normal map, and albedo image from top to bottom of the close view. From left to right, the columns represent results by PP rendering, PP rendering with Pull-Push, our fast pipeline method and HQS rendering.

4.2 Denoising Monte Carlo Rendering

In Section 3.2, except the de/remodulation and temporal accumulation, we introduced three enhancements aimed at improving the results produced by the base ROF model for Monte Carlo denoising. In this section, our focus on the empirical validation of these proposed enhancements through a series of comparative and ablation studies. Our goal is to evaluate whether these adjustments can lead to noticeable improvements, thereby justifying their integration into the model. Following this, we will evaluate the performance of our updated method in relation to the current state of the art methods in the field. The experiments are performed on the datasets provided by [MZV+20], which include 5 scenes: Classroom, Sponza, Sponza glossy and Living room, and San Miguel. All the subsequent experiments incorporate demodulation and temporal accumulation steps as part of the processing pipelinel. All results displayed in this section are post gamma correction, using a gamma value of 2.2. We use the Root Mean Square Error (RMSE) and the Structural Similarity Index Measure (SSIM) to evaluate the quality of denoised images.

4.2.1 Impact of Auxiliary Features

This section is dedicated to analyzing the effect of auxiliary features on the denoising results produced by TV models. We establish the ROF model, defined by Equation 2.4, as our baseline. Subsequently, we adapt this model by incorporating auxiliary features to reweight its TV term, as defined by Equation 3.62. The results obtained from both models are then compared, enabling us to evaluate the influence of these auxiliary features.

For the same scene, all parameters remain consistent. The only difference lies in the variation term, which is either weighted or unweighted by auxiliary features. Visual comparisons of the quality differences are displayed in Figure 4.5. The results are as expected. Given that the coherence map values, computed by Equation 3.60, fall within the range $[0, 1]$, the geometry-aware model applies similar penalties to smooth surface pixels as the model without geometric feature guidance. Conversely, pixels representing sharp features, such as edges, are assigned to less aggressive variation terms, increasing their preservation likelihood. For instance, in the denoised images of the Classroom scene produced by the standard ROF model, the edges of the chairs are noticeably oversmoothed. The nearby background pixels, which were originally dark, appear brighter than intended. In contrast, the geometry-aware model applies adaptive TV terms to edge and non-edge pixels, preventing the appearance of unnatural bright transitions. Additionally, the incorporation of albedo texture also enhances detail preservation, as evident in the Spnoza scene where the green and red curtains exhibit well-preserved textures. The parameter s_A in Equation 3.63 controls the degree to which features are preserved. A larger s_A value results in better-preserved features and nearby shadows but can also potentially retain artifacts in these areas. This is evident in the unremoved noise on the artificial flowers in the Living Room image denoised by the ROF model guided by auxiliary features. We also computed the SSIM and RMSE metrics to statistically assess the effect of utilizing auxiliary features. The

Auxillary Features	Sponza		Sponza glossy		Classroom		Living room		San Miguel	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
without	0.050	0.915	0.076	0.825	0.069	0.865	0.064	0.878	0.087	0.666
with	0.041	0.929	0.070	0.833	0.035	0.920	0.047	0.901	0.080	0.699

Table 4.1: This table delivers a statistical comparison between the base model and the model enhanced with auxiliary features, using SSIM and RMSE as performance metrics. Lower RMSE values and higher SSIM scores indicate better performance.

findings, as presented in Table 4.1, indicate that the model guided by auxiliary features tends to yield results with reduced RMSE and increased SSIM. This quantitative analysis aligns with our visual observations.

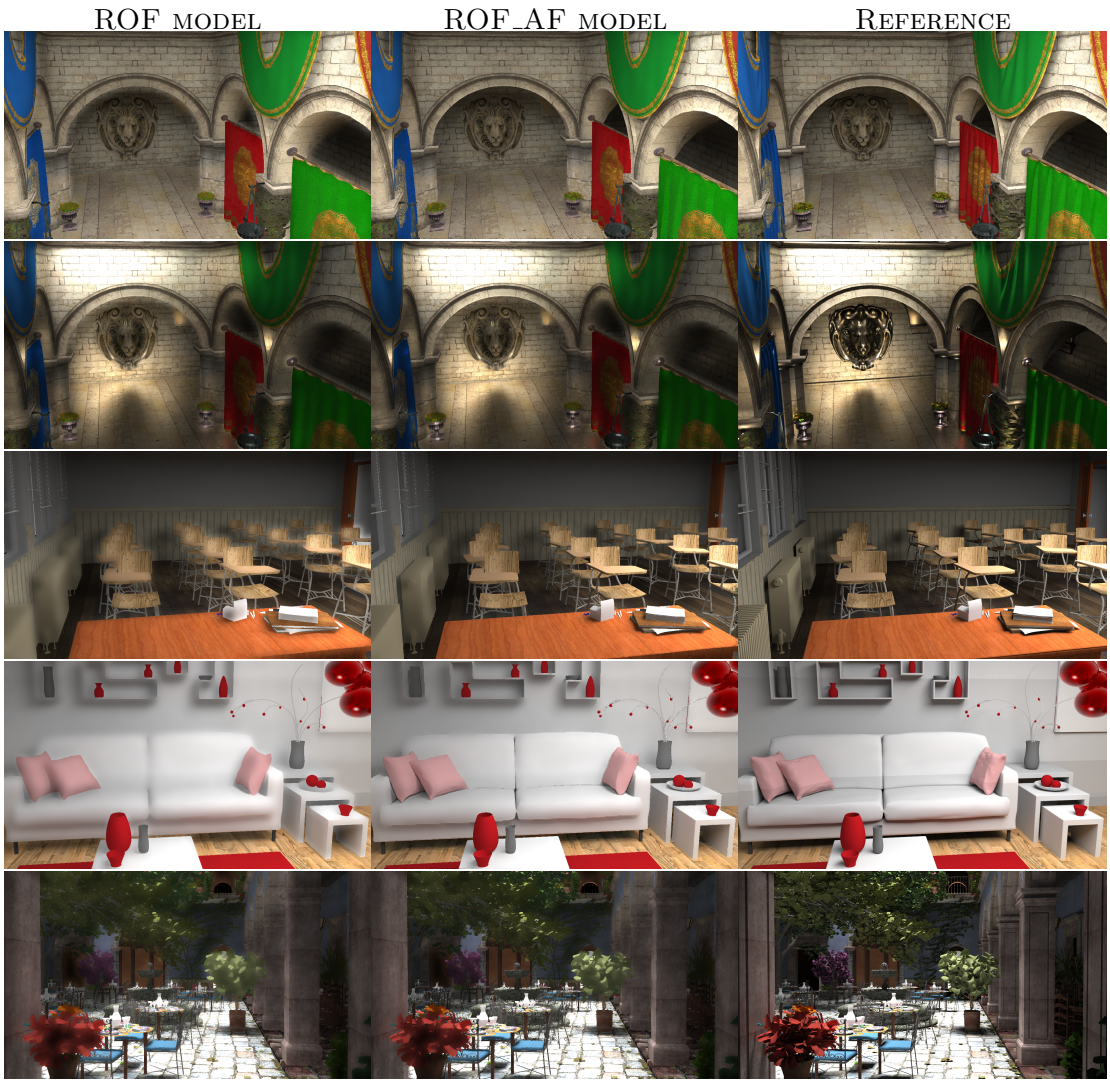


Figure 4.5: This figure presents a series of comparisons between outputs generated by the standard ROF model and the ROF model guided by auxiliary features (termed as ROF_AF model). The scenes shown, from top to bottom, represent examples from the Sponza, Sponza-Glossy, Classroom, and San-Miguel scenes, respectively. Columns from left to right display the results produced by the ROF model, the ROF_AF model, and the reference, respectively. The incorporation of Auxiliary Features allows for the preservation of finer details. However, it is worth noting that this occasionally results in residual noise within areas of sharp detail.

4.2.2 Comparative Analysis of Data Fidelity Terms: L1 Norm vs. Squared L2 Norm

In this section, we delve into a comparative analysis between results produced by the TV model with 11 norm of data fidelity term and the squared l2 norm of data fidelity

term, respectively. The advantages and limitations of both types of data fidelity terms were outlined in Section 3.2.4. In this section, we visually illustrate these characteristics through examples. Figures 4.6 and 4.7 respectively showcase the results produced by the TV-L1 and TV-L2 models. Moreover, we also present results from the proposed hybrid model, which incorporates both types of data fidelity. The TV-L1 and TV-L2 models used for these tests are defined as:

$$\lambda_1|u - u_0| + \boldsymbol{\alpha} \cdot |\nabla u| \tag{4.1}$$

$$\lambda_2\|u - u_0\|^2 + \boldsymbol{\alpha} \cdot |\nabla u| \tag{4.2}$$

Here, the $\boldsymbol{\alpha}$ is computed as described in Section 3.2.3.

As stated in Section 3.2.4, the TV-L1 model tends to outperform the TV-L2 model in preserving sharp features and shadows. This property is noticeable in the denoised images of the Sponza scene (Figure 4.6) and the Classroom scene (Figure 4.7), where the blue insets show better preserved sharp features by the TV-L1 model compared to the denoised results by the TV-L2 model. The marked patches in the images denoised by the TV-L2 model demonstrate oversmoothed edges and shadows. Similarly, the denoised results of the Living room scene by the TV-L1 model (Figure 4.7) show better preserved shadows. This behavior is also evident when we denoise images with reflections. The l1 norm data fidelity, owing to its robustness to outliers, manages the highlights, which are often treated as outliers by the model, in a manner that minimizes their influence on neighboring pixels. This results in neighboring pixels retaining more of their original values, thereby leading to a more effectively preserved sharp contrast between the highlight and its surroundings. On the other hand, the squared l2 norm of data fidelity minimizes the sum of squared differences between the original and denoised images. This tends to amplify larger differences (outliers) so that a bright highlight in a darker region could heavily influence its neighboring pixels during the denoising process. As the model tries to minimize the overall error, it may distribute the intensity of the highlight spot across its neighboring pixels, causing the highlight to appear less sharp or intense. This phenomenon can be observed through the blue insets in the example of the Sponza glossy scene in Figure 4.6.

Furthermore, we observe differences in the illumination of darker areas within a scene when comparing the results of the TV-L1 and TV-L2 models. Typically, results produced by the TV-L1 model tend to be darker compared to the reference image, while the TV-L2 model tends to result in brighter images. These can be observed through the red insets from the Sponza and Classroom scenes in Figures 4.6 and 4.7. In the context of Monte Carlo rendering, noise often manifests as darker pixels in areas of the scene with less illumination. This is due to these areas receiving less light and, consequently, fewer samples contributing to the final pixel color. When the number of samples is relatively low, the noise can often appear darker than the true color of these pixels. In the region where dark noise pixels considerably outnumber the correctly exposed pixels, the median pixel value will be darker. Since the TV-L1 model would converge towards the median, this may result in a darker output in these dark areas with less correctly exposed pixels. Conversely, the TV-L2 model,

which works to minimize the mean squared difference, may yield a brighter result. This is due to the mean, which the TV-L2 model converges towards, being more influenced by the brighter pixels. Consequently, the resulting image might be slightly brighter overall, particularly in dark regions of the images. The darker pixels in areas with less illumination, denoised by the TV-L1 model, can be seen in the red insets of the Sponza and Classroom scenes in Figures 4.6 and 4.7.

In response to these observations, we propose the use of a Hybrid model with both types of data fidelity terms as a balanced solution. From the examples provided, it's evident that this model effectively preserves sharp features while successfully reduce the overly dark results that the TV-L1 model often produces in darker areas.

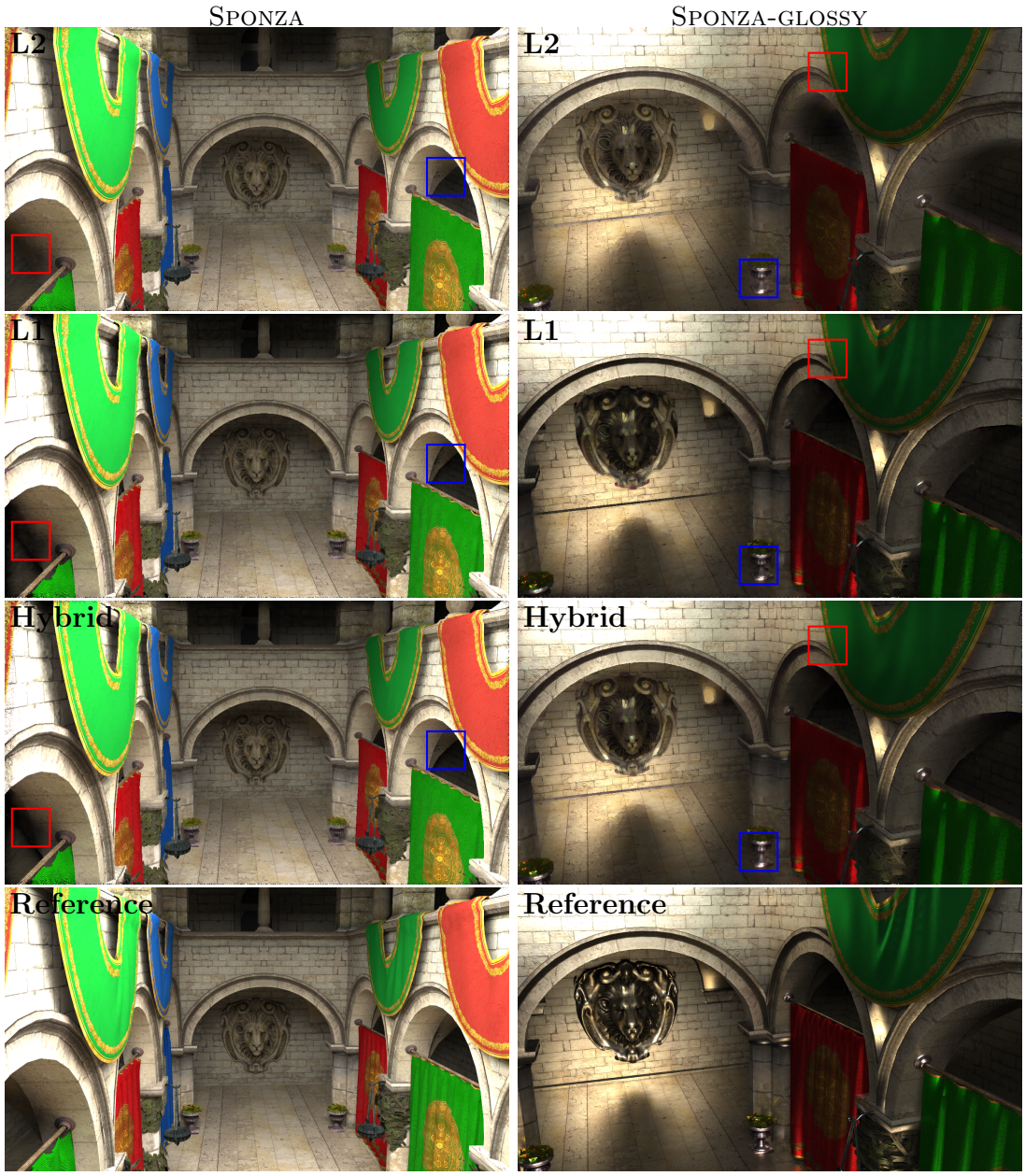


Figure 4.6: This figure presents a comparison of the denoising results of the Sponza scene and Sponza-glossy scene produced using the TV-L1 model, TV-L2 model and hybrid model.

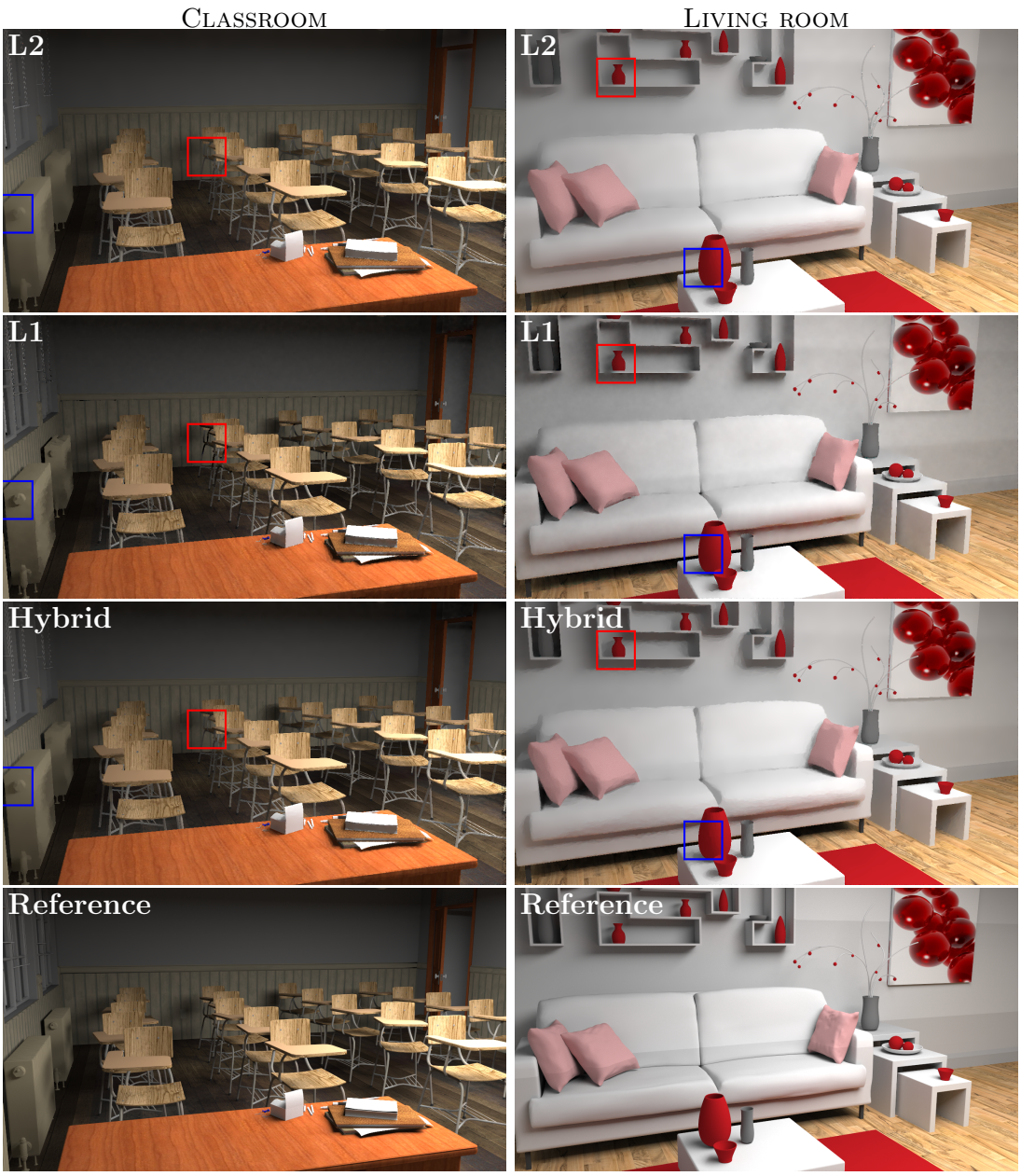


Figure 4.7: This figure presents a comparison of the denoising results of the Classroom scene and Living room scene produced using the TV-L1 model, TV-L2 model and hybrid model.

4.2.3 Influence of Noise Estimation Techniques

In this section, we investigate the role of noise estimation techniques in noise reduction. We use the hybrid geometry-aware model defined by Equation 3.64 as a baseline to denoise images. Then, maintaining all other model parameters at the same settings as this baseline,

Error Estimate	Sponza		Sponza glossy		Classroom		Living room		San Miguel	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
None	0.0413	0.929	0.062	0.853	0.034	0.921	0.034	0.919	0.085	0.700
MAD	0.0343	0.938	0.065	0.852	0.035	0.919	0.033	0.921	0.083	0.70
VAR	0.037	0.933	0.068	0.847	0.035	0.921	0.034	0.920	0.084	0.692

Table 4.2: This table presents a statistical comparison between the base model and the model enhanced with noise estimation techniques, using either MAD or VAR. The comparison is conducted using SSIM and RMSE as performance metrics, where lower RMSE values and higher SSIM scores indicate better performance.

we incorporate a noise estimation step into the calculation of the coherence map (as detailed in Section 3.2.5). We explore two noise estimation methods: MAD and spatiotemporal variance (VAR). We then compare the results obtained by these techniques.

Statistical results presented in Table 4.2 show that noise estimation steps significantly improve the denoising results of the Sponza scene. For other scenes, the impact is comparatively smaller. From a purely statistical standpoint, the difference between the results produced by models utilizing MAD and VAR noise estimation methods is minimal.

Nevertheless, upon considering subjective image quality, the results of models that incorporate noise estimation methods exhibit notable differences compared to those that do not. This can be observed in Figures 4.8 and 4.9. By weighting the TV term with the noise estimation matrix, we are able to effectively remove noise that the baseline model did not eliminate, particularly noise located at sharp geometric areas. Although this process may result in the loss of some very detailed features, which explains the minor improvements shown in the statistical data, we believe that this is a worthwhile trade-off for significant noise reduction. In terms of visual differences, images denoised by models using MAD or VAR for error estimation show only marginal differences, consistent with the minor differences in SSIM and RMSE values. In certain areas, the model using MAD better preserves details, while in others, the model using VAR performs better. However, for complex scenes, noise estimation with VAR leads to superior noise reduction compared to MAD, as evidenced by the insets in the denoised San Miguel scene and the blue insets in the Sponza-glossy scene. This observation is consistent with the declaration in [KS13], where it is suggested that the MAD technique might encounter difficulties in complex scenarios.

For a real-time implementation of our proposed method, both MAD and VAR could be taken into consideration due to their similar denoising performance. The choice between them would be influenced by their respective implementation speeds.

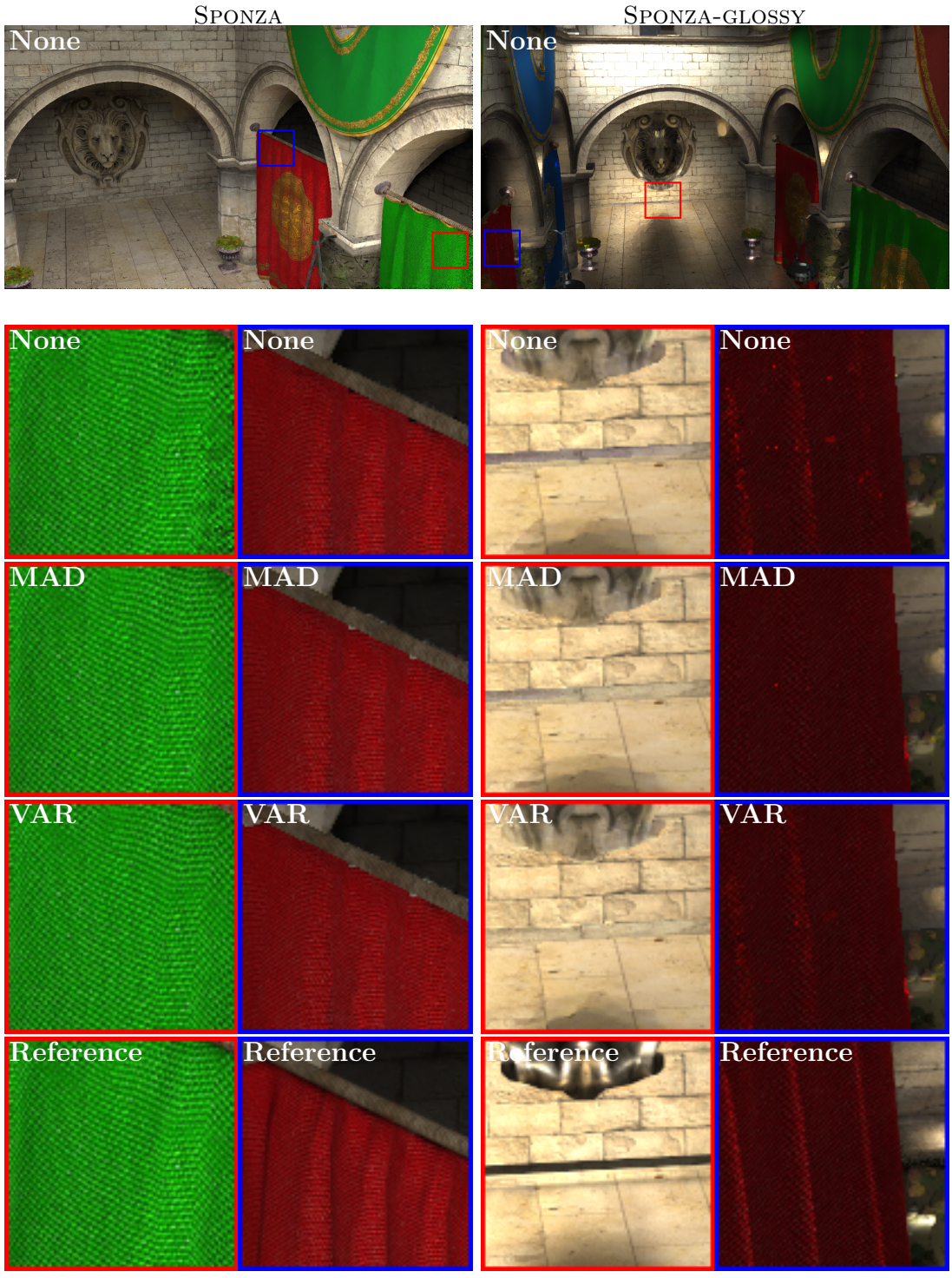


Figure 4.8: This figure displays the visual differences between the results obtained from models that incorporated error estimation versus those that did not.The models incorporate either MAD or VAR for error estimation. The insets marked with "None" shows the results of the model without noise estimation.

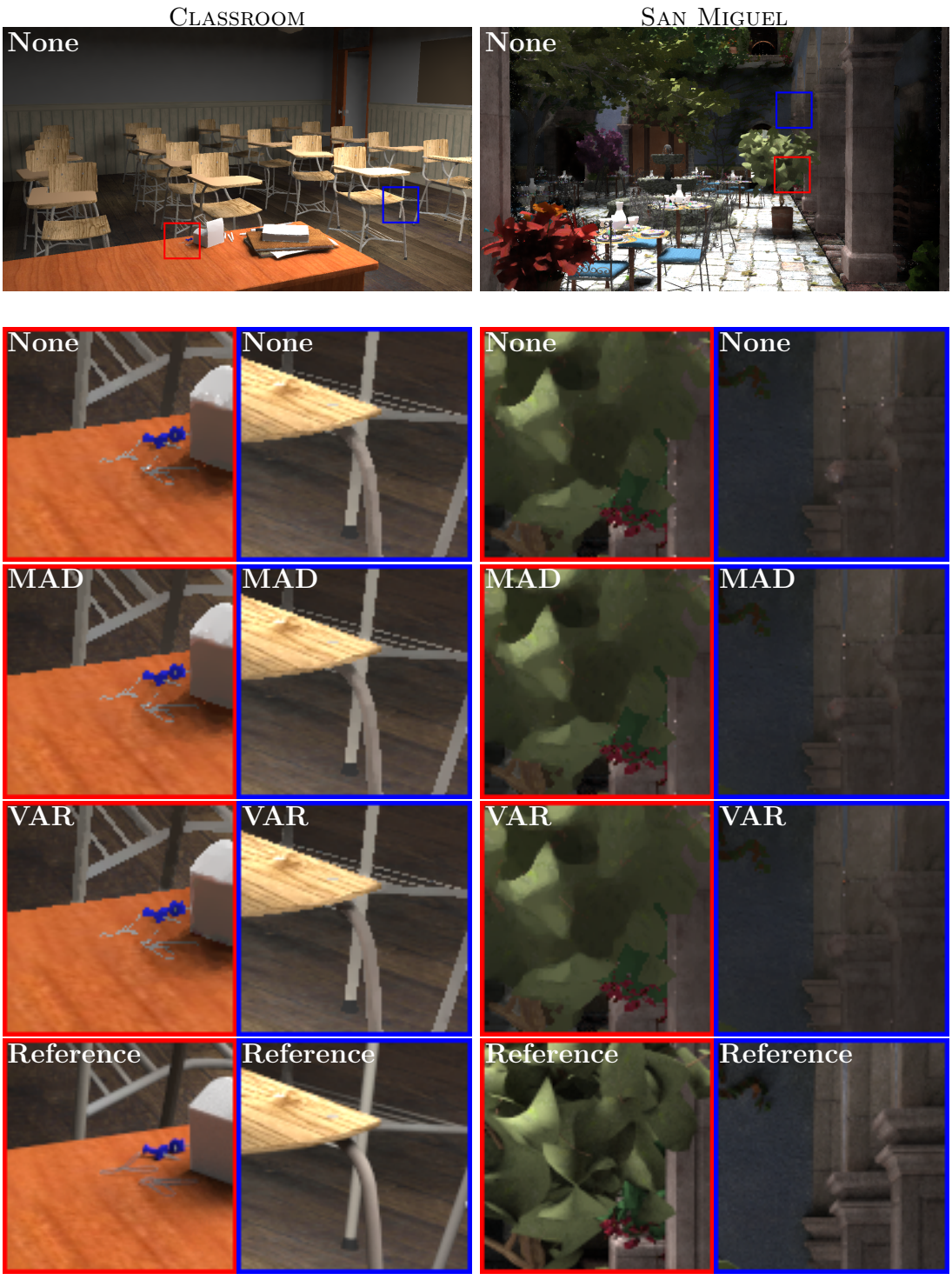


Figure 4.9: This figure displays the visual differences between the results obtained from models that incorporated error estimation versus those that did not.The models incorporate either MAD or VAR for error estimation. The insets marked with "None" shows the results of the model without noise estimation.

4.2.4 Results

In this section, we compare our proposed methods with five state-of-the-art methods: Blockwise Multi-Order Feature Regression (BMFR) [KIM+19], Nonlinearly Weighted First-order Regression (NFOR) [**?**], Guided Filtering (GF) [**?**], OptiX Neural Network Denoise (ONND) [CKS+17], and Spatiotemporal Variance-Guided Filtering (SVGF) [SKW+17]. We present both objective and subjective comparisons of these methods.

Objective quality comparisons are presented in Tables 4.3 and 4.4. We employ the mean RMSE and mean SSIM to evaluate image quality. Video Multi-method Assessment Fusion (VMAF)[1] is used to assess video quality. VMAF is a metric designed to evaluate the perceived visual quality of videos and has shown a good correlation with subjective quality.

Error Estimate	Sponza		Sponza glossy		Classroom		Living room		San Miguel	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
BMFR	**0.028**	0.906	0.056	0.849	0.036	**0.926**	0.032	0.938	0.090	0.698
GF	0.051	0.937	0.074	0.844	0.059	0.911	0.061	0.903	0.101	0.692
NFOR	0.031	0.891	**0.050**	0.835	**0.032**	**0.926**	**0.027**	**0.940**	**0.081**	**0.704**
ONND	0.059	0.744	0.067	0.774	0.043	0.866	0.053	0.912	0.098	0.625
SVGF	0.066	0.880	0.090	0.854	0.056	0.924	0.043	0.920	0.116	0.702
Ours(M)	0.034	**0.938**	0.051	**0.872**	0.035	0.919	0.033	0.921	0.083	0.700
Ours(V)	0.037	0.933	0.055	0.869	0.035	0.921	0.034	0.920	0.084	0.692

Table 4.3: This table presents a statistical comparison between our proposed methods and the state-of-the-art methods. The comparison is conducted using mean SSIM and mean RMSE as performance metrics, where lower RMSE values and higher SSIM scores indicate better performance.

Filter	Sponza	Sponza glossy	Classroom	Living room	San Miguel
	VMAF	VMAF	VMAF	VMAF	VMAF
BMFR	**65.63**	32.33	44.59	50.83	20.86
GF	60.48	23.62	26.43	21.76	27.65
NFOR	59.84	34.48	**50.88**	55.06	23.89
ONND	39.76	24.92	43.58	**55.34**	19.65
SVGF	46.41	18.94	37.59	39.94	10.59
Ours(M)	63.70	**36.64**	50.14	50.30	**34.10**
Ours(V)	59.39	27.06	50.02	50.26	31.21

Table 4.4: This table presents a statistical comparison between our proposed methods and the state-of-the-art methods using VAMF, where higher VAMF scores indicate better perceived visual quality of videos.

[1] model version v0.6.1 from VMAF-on-browser: https://vmaf.dev/

Parameters	Sponza	Sponza glossy	Classroom	Living room	San Miguel
α_0	1.0	1.0	0.5	0.5	0.5
s_A	10.0	10.0	10.0	5.0	5.0
s_{E_M}	3.0	3.0	5.0	3	10.0
s_{E_V}	1.0	1.0	1.0	1.0	5.0

Table 4.5: $\lambda_1 = 0.1$, $\theta_1 = 0.5$, $\lambda_2 = 0.5$, $\theta_2 = 5$, $k_{max} = 20$, $tol = 4e-3$ fixed for all scenes.

Overall, based on statistical comparisons, our reconstructed results closely align with the reference images. In the majority of our test scenes, our results align comparably with other real-time methods in terms of RMSE, SSIM, and VMAF, with only minor differences at most. Specifically, our method demonstrated superior performance in certain aspects: it achieved the highest mean SSIM for the Sponza scene, the highest mean SSIM and VMAF for the Sponza-glossy scene, and the highest VMAF for the San Miguel scene.

The quality of results by TV-based methods are dependent on the balance parameters. To achieve the best results, the parameters of the models were individually selected for each scene. Individually selected the parameters for each specific scene allows us to fully exploit the potential of our method and achieve the best possible results. The selected parameters corresponding to each scene are enlisted in Table 4.5. The figures used for subjective quality evaluation also employ models with scene-dependent parameters. On the contrary, we also conducted tests where the same parameters were used for all scenes. These experiments demonstrated that our methods still deliver reliable results under consistent conditions. The statistical data for this can be found in Tables 4.6 and 4.7.

Error Estimate	Sponza		Sponza glossy		Classroom		Living room		San Miguel	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
BMFR	**0.028**	0.906	0.056	0.849	0.036	**0.926**	0.032	0.938	0.090	0.698
GF	0.051	0.937	0.074	0.844	0.059	0.911	0.061	0.903	0.101	0.692
NFOR	0.031	0.891	**0.050**	0.835	**0.032**	**0.926**	**0.027**	**0.940**	**0.081**	**0.704**
ONND	0.059	0.744	0.067	0.774	0.043	0.866	0.053	0.912	0.098	0.625
SVGF	0.066	0.880	0.090	0.854	0.056	0.924	0.043	0.920	0.116	0.702
Ours(M)	0.034	**0.938**	0.051	**0.872**	0.038	0.912	0.045	0.908	0.084	0.684
Ours(V)	0.037	0.933	0.055	0.869	0.044	0.904	0.045	0.908	0.091	0.665

Table 4.6: This table presents a statistical comparison between our proposed methods and the state-of-the-art methods. The model employs the same parameters set for all scenes.

Filter	Sponza	Sponza glossy	Classroom	Living room	San Miguel
	VMAF	VMAF	VMAF	VMAF	VMAF
BMFR	**65.63**	32.33	44.59	50.83	20.86
GF	60.48	23.62	26.43	21.76	27.65
NFOR	59.84	34.48	**50.88**	55.06	23.89
ONND	39.76	24.92	43.58	**55.34**	19.65
SVGF	46.41	18.94	37.59	39.94	10.59
Ours(M)	63.70	**36.64**	41.04	36.62	**28.61**
Ours(V)	59.39	27.06	30.56	36.39	20.28

Table 4.7: This table presents a statistical comparison between our proposed methods and the state-of-the-art methods using VAMF. The model employs the same parameters set for all scenes.

Subjective quality of the proposed methods are evaluated, with examples displayed in Figures 4.10, 4.11, 4.12, 4.13, and 4.14. Broadly speaking, our method demonstrates a strong ability to preserve detail features and illumination across most scenes. For instance, the Classroom scene insets and the red insets from the San Miguel scene represent cases where our algorithm successfully maintains the original illumination and shadows, while methods like GF and SVGF often underperform, tending to overblur these elements. The Sponza scene insets show that our methods outperform in preserving sharp geometric features and detail textures. Importantly, our method stands out in handling regions with reflection, as shown in denosied results of Sponza-glossy scene. Where most methods struggle with overblurring in reflective regions, our method with noise estimation by spatiotemporal variance effectively eliminates high-frequency noise while maintaining good reflection quality. This performance is consistent with the impressive statistical results shown previously.

Our methods have been implemented in MATLAB and remains an offline approach. However, given the straightforward implementation and that satisfactory results are achieved within 20 iterations, we believe that an implementation on a GPU can achieve processing times of around 10 ms per frame. This performance would meet the requirements for real-time denoising applications.

Our methods do present certain limitations. In some cases, such as noticeable in the blue insets of the Living Room scene, we also encounter oversmoothing. This problem typically occurs due to an aggressive selection of balance parameters. To optimize performance for a specific scene, these parameters should ideally be chosen in a manner that is dependent on the scene itself. Furthermore, the red insets in Figure 4.10 emphasize areas where our methods lightly darken less illuminated regions. It is also noticeable in the ONND method's results. This effect is especially apparent in the first 2-3 frames, owing to the limited valid samples. This is a consequence of the l1 norm data fidelity term in our denoising model. Although the 11 norm data fidelity term preserves shadow features better, it also introduces this darkening effect as a trade-off. Moreover, our methods can also introduce artifacts in more challenging areas of a scene, particularly where sharp geometric transitions coincide with intense color changes, our method sometimes introduces artifacts. These artifacts

often appear as overly bright pixels within the background. Specifically, dark pixels, which originally belong to the background but are adjacent to bright, high-intensity edges, can be smoothed out by our model. This results in these dark pixels being mistakenly smoothed into bright pixels. This phenomenon is particularly evident in the blue insets of Figure 4.10. It's worth noting that this type of artifact is not unique to our method, but can also be found in results produced by the GF method, although GF typically generates more severe artifacts. Another characteristic issue is staircasing artifacts, a consequence of the TV term's nature. These can be observed in the red insets of the Classroom and Living room scenes. Additionally, as the camera moves in and out, pixels on the sides of the frames often have fewer valid samples even after reprojection, and our method struggles to remove strong noise in these areas without oversmoothing neighboring regions. Finally, the insets of the San Miguel scene illustrate that our methods struggle to effectively eliminate high-frequency noise, even at the cost of overblurring many features. This reveals the need for additional refinement in handling complex noise distributions.

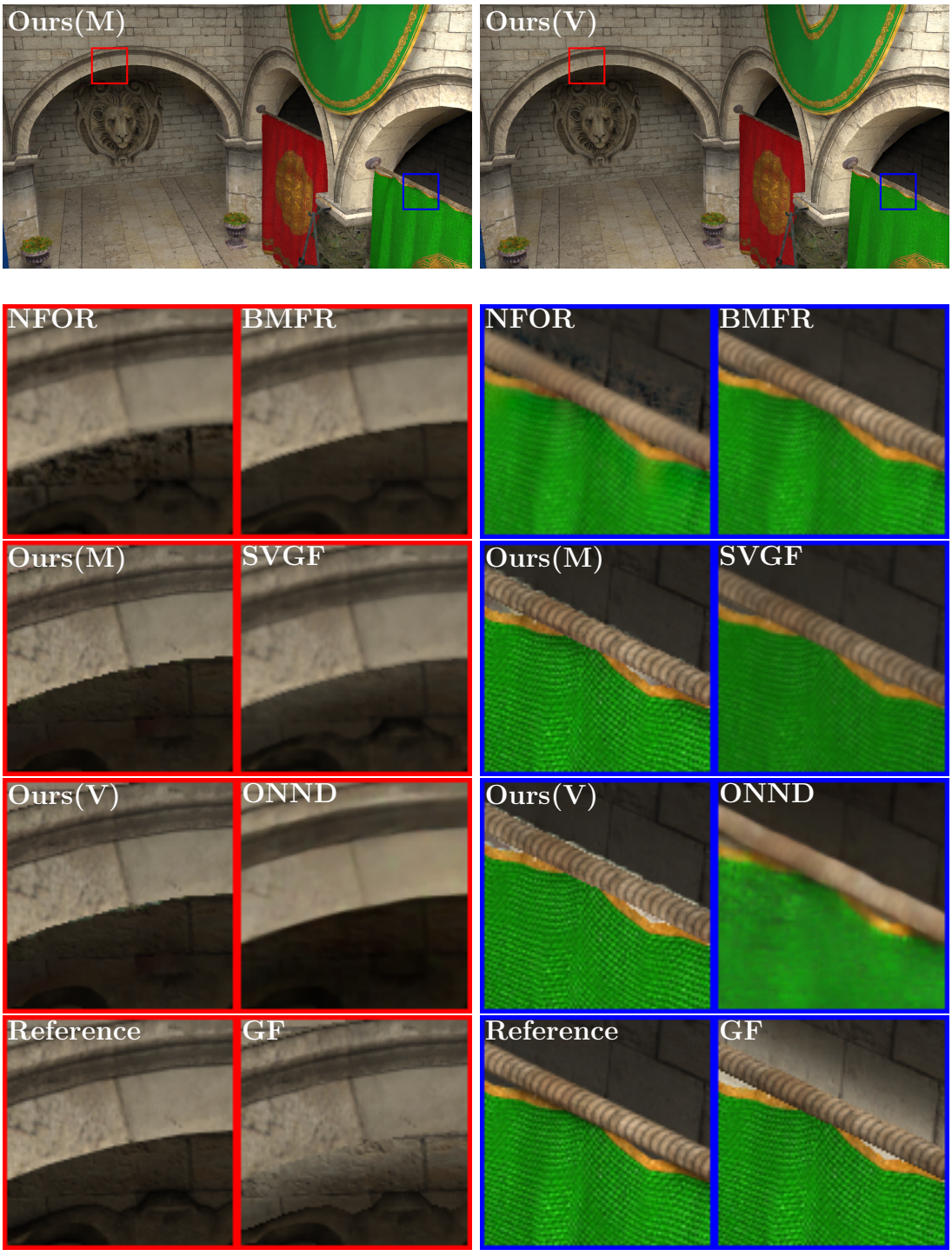


Figure 4.10: This figure illustrates the Sponza scene denoised using our method and compares it with a reference image rendered at 4096 spp. Our method successfully preserves sharp features and shadows, demonstrating performance comparable to the BMFR method and superior to the ONND and GF methods.

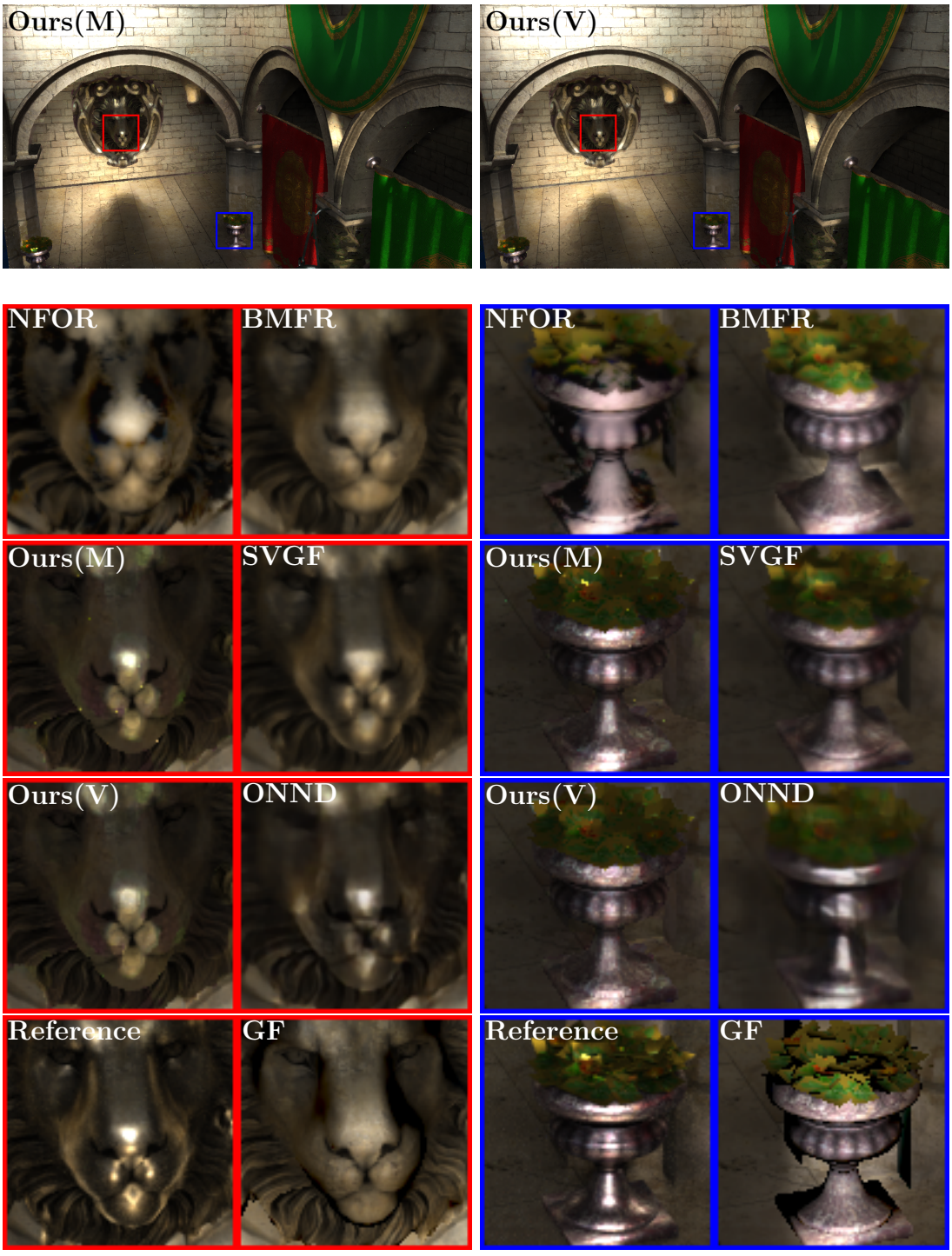


Figure 4.11: This figure illustrates the Sponza-glossy scene denoised using our method and compares it with a reference image rendered at 4096 spp. Our method with noise estimation by spatiotemporal variance effectively eliminates high-frequency noise while maintaining good reflection quality.

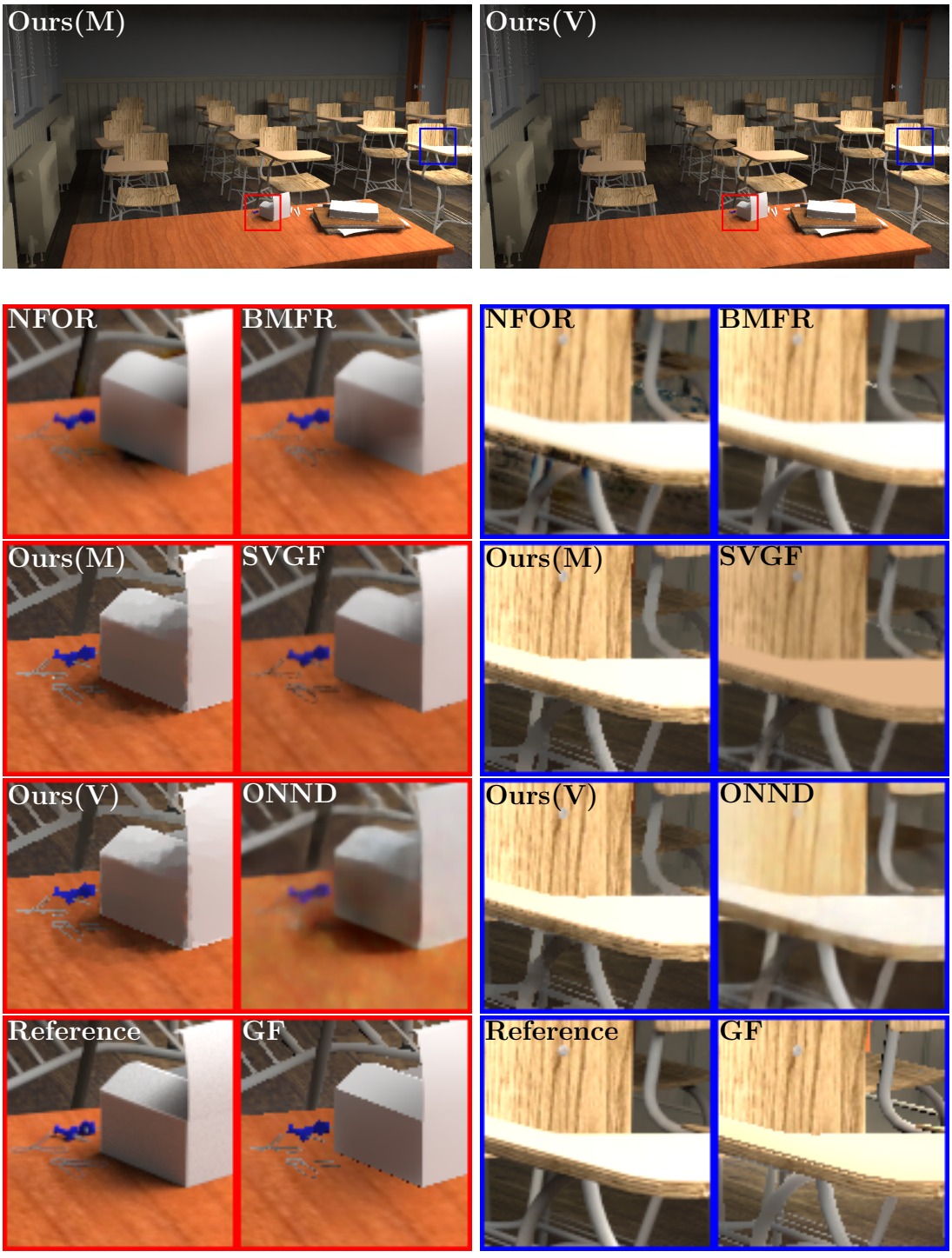


Figure 4.12: This figure illustrates the Classroom scene denoised using our method and compares it with a reference image rendered at 4096 spp. Our method demonstrates effective preservation of sharp features and illumination, outperforming the ONND and SVGF methods.

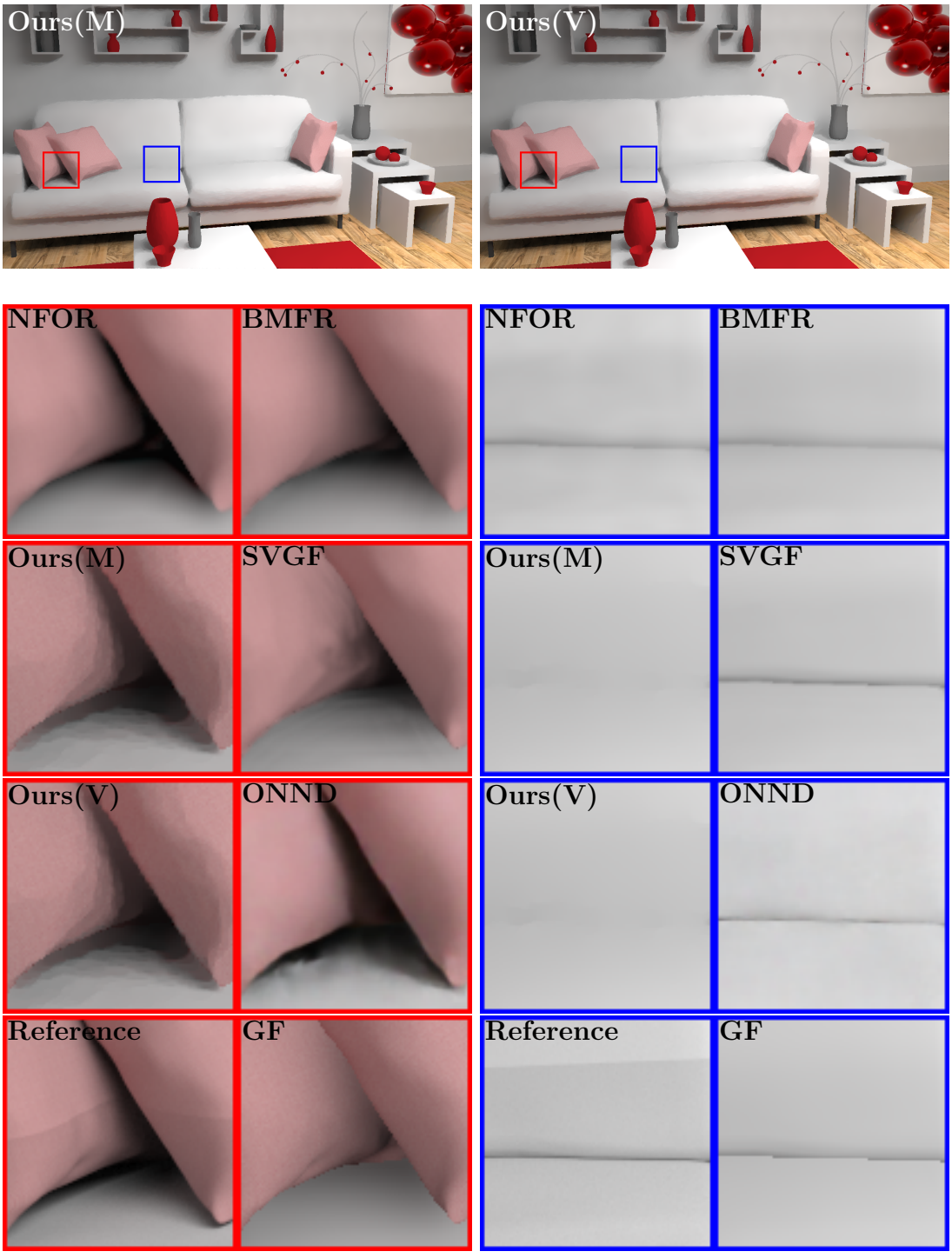


Figure 4.13: This figure illustrates the Living room scene denoised using our method and compares it with a reference image rendered at 4096 spp.These figures exhibit staircasing artifacts due to the nature of the Total Variation term and oversmoothing issue in certain regions.

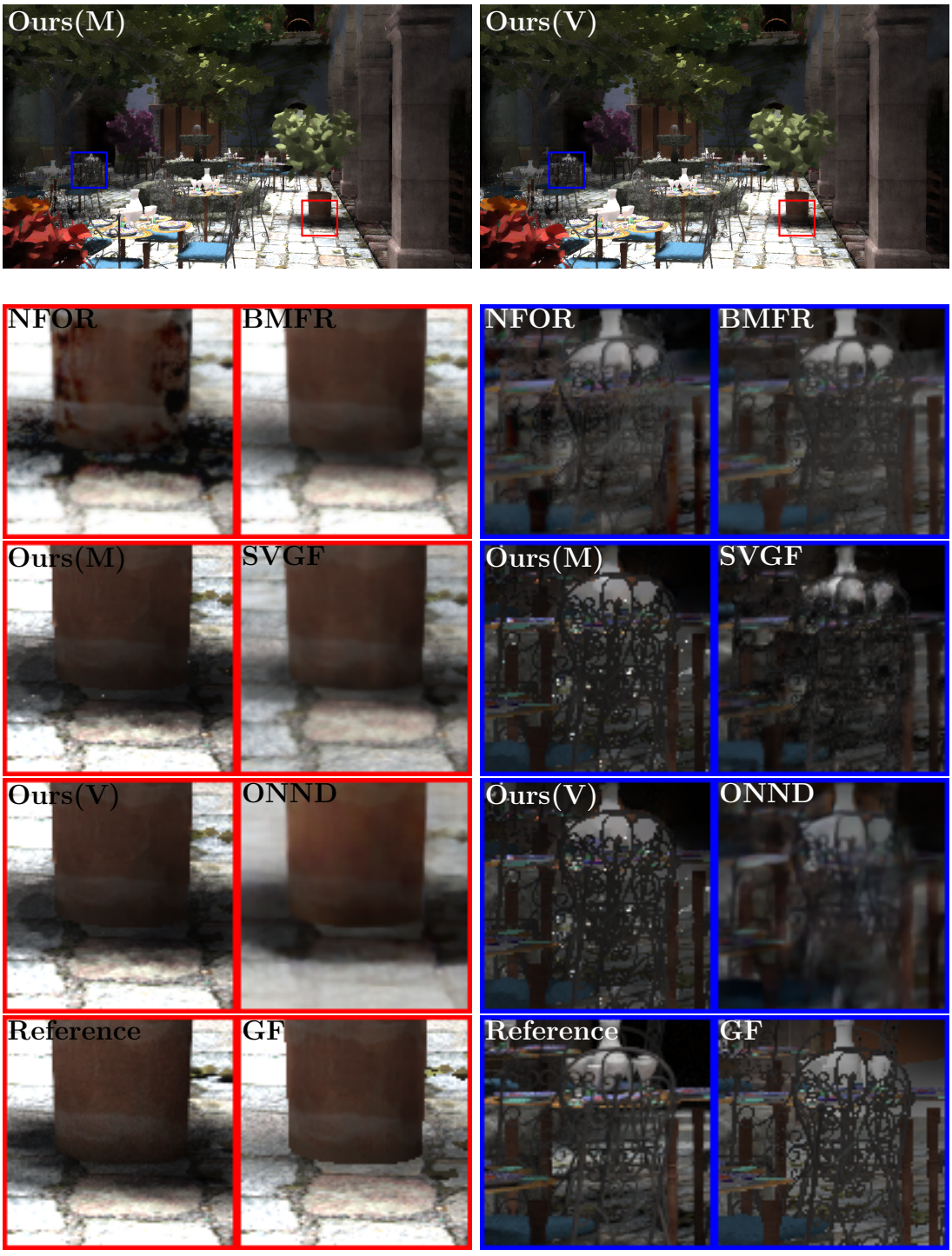


Figure 4.14: This figure illustrates the San Miguel scene denoised using our method and compares it with a reference image rendered at 4096 spp. Our method effectively preserves shadow features. In comparison, the GF method falls short in preserving shadow features. However, our method encounters challenges in effectively removing high-frequency noise.

Chapter 5

Conclusion and Future Work

In this master thesis, we have advanced the application of TV-based methods to two tasks within computer graphics: smoothing discontinuous textured surfaces from point-rendered images and denoising images from Monte Carlo rendering. Concerning the smoothing of discontinuous surfaces, several alternative smoothing pipelines were developed. These pipelines, built upon specific TV models and integrating geometric information, successfully addressed discontinuities in the depth map, and subsequently reconstructed textured surfaces without introducing significant boundary artifacts. Our methods have demonstrated their efficacy in producing appealing, uniformly smoothed reconstructions after several optimization iterations. Furthermore, we proposed an efficient pipeline that incorporates the advanced Pull-Push algorithm to produce smooth, infilled surfaces even with highly discontinuous inputs, within a few numbers of iterations. These methods have proven effective in smoothing discontinuous surfaces while preserving essential geometric features from the original rendering. In the field of denoising Monte Carlo rendering, our TV model was extended and enhanced by incorporating auxiliary features and noise estimation techniques. By leveraging characteristics of the TV-based methods, we designed a hybrid, adaptive model that reduces noise without significantly compromising feature preservation. Coupled with temporal accumulation, our approach can efficiently mitigate noise in 1 spp path traced frames, maintaining key features and offering competitive results.

Looking forward, we foresee our approach to be used as a post-processing step within real-time point-based rendering pipelines. However, the temporal stability of video frames featuring the inpainted textured surfaces is yet to be assessed. Future research should address this to ensure the consistent performance of our method over time.

In terms of denoising Monte Carlo rendering, the denoising of temporally accumulated 1 spp frames can be achieved in a few numbers of iterations, making the transition to a GPU implementation a natural progression for real-time applications. Notably, our results still display aliasing issues, which could be mitigated by incorporating a Temporal Anti-Aliasing (TAA) step into our denoising pipeline. It could also be worthwhile to test our method

on high spp frames, without relying on temporal accumulation. Moreover, considering the different noise characteristics of direct and indirect illumination, our TV model could be employed to denoise these two aspects separately, using balance parameters with different levels of aggressiveness. The separately denoised images for direct and indirect illumination can be recombined to generate the final denoised image. This approach could potentially yield more effective noise reduction and overall improved image quality.

List of Figures

List of Algorithms

List of Tables

Acronyms

BMFR Blockwise Multi-Order Feature Regression. 55

EWA high-quality Elliptical Weighted Average. 6

FFT Fast Fourier Transform. 16

GF Guided Filtering. 55

HDR High Dynamic Range. 38

HQS High-Quality-Splatting. 39

HTVL1-N model HTVL1 model with the normal-based regularizer. 24

HTVL2 model High-order TV model with a squared l2 data fidelity term. 15

HTVL2-N model HTVL2 model with the normal-based regularizer. 23

MAD Median Absolute Deviation. 36

NFOR Nonlinearly Weighted First-order Regression. 55

ONND OptiX Neural Network Denoise. 55

PDEs partial differential equations. 16

PP Point-to-Pixel. 39

RMSE Root Mean Square Error. 45

ROF Rudin-Osher-Fatemi. 4

SSIM Structural Similarity Index Measure. 45

SVGF Spatiotemporal Variance-Guided Filtering. 55

TAA Temporal Anti-Aliasing. 64

TV Total Variation. 4

TV-L1 model TV model with a l1 data fidelity term. 25

TV-L2 model TV model with a squared l2 data fidelity term. 25

VAR spatiotemporal variance. 52

VMAF Video Multi-method Assessment Fusion. 55

Bibliography

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003.

[BDG18] Hassan Bouchiba, Jean-Emmanuel Deschaud, and Francois Goulette. Raw point cloud deferred shading through screen space pyramidal operators. In *39th Eurographics*, 2018.

[BLMD18] Giang Bui, Truc Le, Brittany Morago, and Ye Duan. Point-based rendering enhancement via deep learning. *The Visual Computer*, 34:829–841, 2018.

[BWP+20] 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)*, 39:148–1, 2020.

[CKS+17] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36:1–12, 2017.

[CP11] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.

[CZL+21] Yuantao Chen, Haopeng Zhang, Linwu Liu, Jiajun Tao, Qian Zhang, Kai Yang, Runlong Xia, and Jingbo Xie. Research on image inpainting algorithm of improved total variation minimization method. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2021.

[DJ94] David L Donoho and Iain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[DSHL10] Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik PA Lensch. Edge-avoiding a-trous wavelet transform for fast global illumination filtering.

In *Proceedings of the Conference on High Performance Graphics*, pages 67–75, 2010.

[EB92] Jonathan Eckstein and Dimitri P Bertsekas. On the douglasârachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical programming*, 55:293–318, 1992.

[EGO+20] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*, pages 108–124. Springer, 2020.

[FZFZ19] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2:1–12, 2019.

[GO09] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM journal on imaging sciences*, 2(2):323–343, 2009.

[HNAD11] Ankur Handa, Richard A Newcombe, Adrien Angeli, and A Davison. Applications of legendre-fenchel transformation to computer vision problems. 2011.

[KB04] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0, 2006.

[KCRH20] Misha Kazhdan, Ming Chuang, Szymon Rusinkiewicz, and Hugues Hoppe. Poisson surface reconstruction with envelope constraints. In *Computer graphics forum*, volume 39, pages 173–182. Wiley Online Library, 2020.

[KH13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32:1–13, 2013.

[KIM+19] Matias Koskela, Kalle Immonen, Markku Mäkitalo, Alessandro Foi, Timo Viitanen, Pekka Jääskeläinen, Heikki Kultala, and Jarmo Takala. Blockwise multi-order feature regression for real-time path-tracing reconstruction. *ACM Transactions on Graphics (TOG)*, 38:1–14, 2019.

[KMŽ+22] Petrus EJ Kivi, Markku J Mäkitalo, Jakub Žádník, Julius Ikkala, Vinod Kumar Malamal Vadakital, and Pekka O Jääskeläinen. Real-time rendering of point clouds with photorealistic effects: a survey. *IEEE Access*, 10:13151–13173, 2022.

[KS13] Nima Khademi Kalantari and Pradeep Sen. Removing the noise in monte carlo rendering with general image denoising algorithms. In *Computer Graphics Forum*, volume 32, pages 93–102. Wiley Online Library, 2013.

[LDQ+16] Wenqi Lu, Jinming Duan, Zhaowen Qiu, Zhenkuan Pan, Ryan Wen Liu, and Li Bai. Implementation of high-order variational models made easy for image processing. *Mathematical Methods in the Applied Sciences*, 39:4208–4233, 2016.

[LHL14] Gang Liu, Ting-Zhu Huang, and Jun Liu. High-order tvl1-based images restoration and spatially adapted regularization parameter selection. *Computers & Mathematics with Applications*, 67(10):2015–2026, 2014.

[LW85] Marc Levoy and Turner Whitted. The use of points as a display primitive. 1985.

[LZ22] Rong Li and Bing Zheng. A spatially adaptive hybrid total variation model for image restoration under gaussian plus impulse noise. *Applied Mathematics and Computation*, 419:126862, 2022.

[McC99] Michael D McCool. Anisotropic diffusion for monte carlo noise reduction. *ACM Transactions on Graphics (TOG)*, 18(2):171–194, 1999.

[MIGMM17] Bochang Moon, Jose A Iglesias-Guitian, Steven McDonagh, and Kenny Mitchell. Noise reduction on g-buffers for monte carlo filtering. In *Computer Graphics Forum*, volume 36, pages 600–612. Wiley Online Library, 2017.

[MZV+20] Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. Real-time monte carlo denoising with the neural bilateral grid. In *EGSR (DL)*, pages 13–24, 2020.

[PGA11] Ruggero Pintus, Enrico Gobbetti, and Marco Agus. Real-time rendering of massive unstructured raw point clouds using screen-space operators. In *Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*, pages 105–112, 2011.

[PSS13] Konstantinos Papafitsoros, Carola Bibiane Schoenlieb, and Bati Sengul. Combined first and second order total variation inpainting using split bregman. 2013.

[RCSW14] Florian Reichl, Matthäus G Chajdas, Jens Schneider, and Rüdiger Westermann. Interactive rendering of giga-particle fluid simulations. In *High Performance Graphics*, pages 105–116, 2014.

[RFS22] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.

[RL08] Paul Rosenthal and Lars Linsen. Image-space point cloud rendering. In *Proceedings of Computer Graphics International*, pages 136–143, 2008.

[RMZ13] Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. Robust denoising using feature and color information. In *Computer Graphics Forum*, volume 32, pages 121–130. Wiley Online Library, 2013.

[ROF92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[Sch23] Moritz Schöpf. Neuronal rendering of point clouds in realtime. 2023.

[SKW+17] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, pages 1–12. 2017.

[SKW22] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Software rasterization of 2 billion points in real time. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(3):1–17, 2022.

[VRM+18] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37:1–15, 2018.

[WT10] Chunlin Wu and Xue-Cheng Tai. Augmented lagrangian method, dual methods, and split bregman iteration for rof, vectorial tv, and high order models. *SIAM Journal on Imaging Sciences*, 3(3):300–339, 2010.

[ZB14] Dominique Zosso and Aurélien Bustin. A primal-dual projected gradient algorithm for efficient beltrami regularization. *Computer Vision and Image Understanding*, pages 14–52, 2014.

[ZPVBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001.