Differentiable Heightfield Path Tracing with Accelerated Discontinuities

Xiaochun Tong
University of Waterloo
Canada
xtong@uwaterloo.ca

Hsueh-Ti Derek Liu
Roblox
Canada
hsuehtil@gmail.com

Yotam Gingold
George Mason University
USA
ygingold@gmu.edu

Alec Jacobson
University of Toronto & Adobe Research
Canada
jacobson@cs.toronto.edu

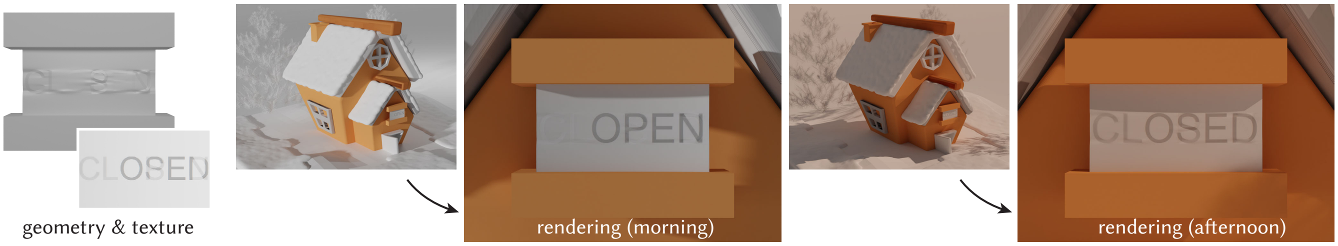


Figure 1: Our method enables real-time performance on inverse rendering tasks on heightfields. For instance, we use our renderer to optimize a heightfield geometry with texture (left) such that the "open" sign appears during the day and the "closed" sign appears in the afternoon.

ABSTRACT

We investigate the problem of accelerating a physically-based differentiable renderer for heightfields based on path tracing with global illumination. On a heightfield with 1 million vertices (1024×1024 resolution), our differentiable renderer requires only 4 ms per sample per pixel when differentiating direct illumination, orders of magnitude faster than most existing general 3D mesh differentiable renderers. It is well-known that one can leverage spatial hierarchical data structures (e.g., the *maximum mipmaps*) to accelerate the forward pass of heightfield rendering. The key idea of our approach is to further utilize the hierarchy to speed up the backward pass—differentiable heightfield rendering. Specifically, we use the maximum mipmaps to accelerate the process of identifying scene discontinuities, which is crucial for obtaining accurate derivatives. Our renderer supports global illumination. we are able to optimize global effects, such as shadows, with respect to the geometry and the material parameters. Our differentiable renderer achieves real-time frame rates and unlocks interactive inverse rendering applications. We demonstrate the flexibility of our method with terrain optimization, geometric illusions, shadow optimization, and text-based shape generation.

CCS CONCEPTS

• **Computing methodologies → Rendering**.

KEYWORDS

differentiable rendering, inverse rendering, heightfield rendering, shading-based editing

ACM Reference Format:
Xiaochun Tong, Hsueh-Ti Derek Liu, Yotam Gingold, and Alec Jacobson. 2023. Differentiable Heightfield Path Tracing with Accelerated Discontinuities. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 6–10, 2023, Los Angeles, CA, USA*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3588432.3591530

1 INTRODUCTION

A heightfield is a 2D grid of scalar values indicating a height over a base elevation. Heightfields are a common geometric representation for terrains, depth, and geometric details in the form of displacement maps. Efficient rendering of heightfield data plays a critical role in visual effects and interactive applications. Such importance has motivated the development of many specialized data structures to construct levels of detail and to accelerate the (forward) rendering of heightfields.

The heightfield is also a popular geometric representation for *inverse* problems in computer vision, such as *shape from shading*. A key challenge is to simulate light transport in a 3D scene and

SIGGRAPH '23 Conference Proceedings, August 6–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08…$15.00
https://doi.org/10.1145/3588432.3591530

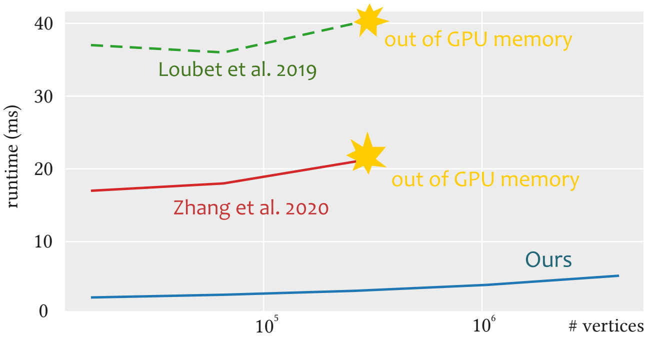


Figure 2: We compare the runtime of our method against generic 3D mesh differentiable renderers, including [Loubet et al. 2019; Zhang et al. 2021]. We report the runtime on the backward pass with varying spatial resolutions. While these renderers are implemented under different frameworks, our method (blue) achieves orders of magnitude speedups and scales to higher resolution geometries. Note that the method by Loubet et al. [2019] (green) was evaluated on a lower resolution image due to memory constraints.

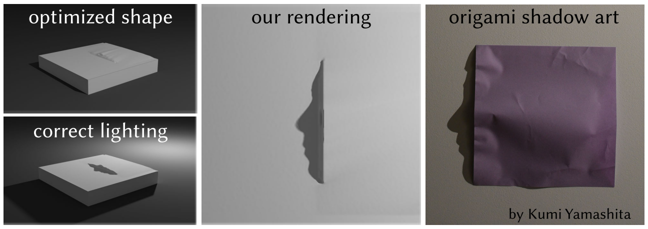


Figure 3: Our differentiable path tracer enables inverse rendering on global illumination effects, such as shadows. We optimize the heightfield geometry (left) so that the shadow it casts looks like a human face, inspired by Kumi Yamashita's shadow art (right). Image obtained from flickr.com photographed by Amaury Laporte under CC BY 2.0

evaluate the gradient with respect to scene parameters. These gradients will then inform the optimization on how to manipulate those scene parameters. Recent progress on *differentiable rendering* has led to breakthroughs estimating such derivatives for generic 3D scenes. Despite being applicable to heightfields, they are slower than necessary. A generic scene often involves arbitrary topologies and irregular discretization, which prevents using acceleration data structures specific to heightfields.

In this work, we propose a differentiable renderer specialized for heightfield geometry. Our renderer is a physically-based path tracer with global illumination, thus it supports taking derivatives with respect to arbitrary scene parameters. Albeit limited to heightfields, our renderer is orders of magnitude faster than generic differentiable renderers (see Fig. 2). The key ingredient to our efficiency is to leverage the well-known *maximum mipmaps* structure, used for forward rendering, to accelerate the backward pass. We demonstrate the efficiency and flexibility of our method in many inverse heightfield problems, including multi-view optimization, geometric illusions (see Fig. 1), shadow optimization (see Fig. 3), and text-based shape generation.

1.1 General Differentiable Rendering

Rendering is a process to turn a 3D scene into a 2D image by modeling the flow of light in the scene [Pharr et al. 2016]. *Differentiable rendering* models the inverse of the rendering process, inferring how a pixel intensity changes with respect to changes in scene parameters. Differentiable rendering has shown to be a powerful tool for many graphics, vision, and machine learning applications [Kato et al. 2020].

Differentiable rendering algorithms are closely related to the rendering approach. Starting with the work by Loper and Black [2014], the earliest works differentiated through *rasterization-based renderers*.

A major focus of this line of work is on approximating derivatives along the silhouette of an object, because it is not differentiable in nature;

approaches have used Gaussian filters [Rhodin et al. 2015], linear approximations [Kato et al. 2018], and sigmoid functions [Liu et al. 2019a; Petersen et al. 2019]. As anti-aliasing techniques are often used to improve rendering quality, Laine et al. [2020] propose an analytic anti-aliasing method to enable visibility derivatives. These rasterization-based differentiable renderers have demonstrated their applicability in 3D reconstruction [Chen et al. 2019; Gao et al. 2020; Munkberg et al. 2021; Wang et al. 2019], geometry editing [Liu et al. 2018], computing adversarial examples [Liu et al. 2019b], and editing vector graphics [Li et al. 2020].

Another line of work focuses on differentiating through *physically based path tracers*. Li et al. [2018] proposed to sample discontinuities in a scene to compute visibility gradients, and the efficiency of boundary sampling is further improved by Yan et al. [2022]. Loubet et al. [2019] propose to use a change of variables to remove discontinuities in the scene. Bangaru et al. [2020] applied the divergence theorem to turn the boundary integral into an area integral and use a boundary-aware convolution to ensure unbiased gradient estimate and differentiability. Zhou et al. [2021] proposed to use beam tracing, instead of conventional ray tracing, to obtain and differentiate anti-aliased visibility. Bangaru et al. [2021] proposed a programmatic solution to automatically differentiating integrals that involve discontinuities. In addition to handling scene discontinuities, people have addressed other aspects of accelerating physically based differentiable rendering. Zhang et al. [2021] presented a *differential path integral* equation, the differential counterpart of the path integral formula. Instead of using automatic differentiation, Nimier-David et al. [2020] proposed to use ray tracing to compute derivatives which leads to a more memory-efficient solution. Vicini et al. [2021a] proposed to re-use the results from forward path tracing, which leads to a fast differentiable renderer with linear complexity. Jakob et al. [2022a] further tackled efficiency by introducing a compiler tailor-made for accelerating differentiable renderers. For a more detailed discussion of physically based differentiable rendering techniques, please refer to [Zeltner et al. 2021].

Our contribution on accelerated discontinuities complements the development of physically based differentiable path tracer by accelerating the computation when the geometry is a heightfield. Albeit limited, this choice leads to efficient and accurate computation of scene discontinuities. Thus, our resulting renderer runs in real-time and unlock interactive applications (see Sec. 3).

We omit a discussion of *neural rendering* techniques which often approximate a physically based renderer with a (differentiable) neural network, e.g., [Che et al. 2018]. For interested readers, please refer to [Tewari et al. 2020] for a recent survey.

1.2 Heightfield (Forward) Rendering

A heightfield is a special type of geometry that represents a shape via a 2D array of height values. Heightfields are commonly used to represent geometric data such as terrains, depth, and surface details (a.k.a. displacement maps). The regular structure of heightfields, in contrast to irregular 3D meshes, has the potential to represent high resolution geometries with faster render times. For instance, one can construct tree structures to adaptively polygonalize a heightfield at different resolutions [Duchaineau et al. 1997; Taylor 1994]. During render time, these spatial hierarchies can be applied to generate continuous levels of detail on-the-fly based on an allowable rasterization error [Lindstrom et al. 1996] or distance to the viewpoint [Röttger et al. 1998]. Hoppe [1998] further extended these techniques to generate temporally-coherent levels of detail. Several approaches to ray tracing heightfields leverage the regular structure of the heightfield to deploy accelerated voxel traversal algorithms [Amanatides and Woo 1987; Henning and Stephenson 2004; Musgrave 1988; Qu et al. 2003]. One can also construct a spatial hierarchy to accelerate the computation of ray-height intersections [Cohen-Or et al. 1996; Cohen-Or and Shaked 1993]. Tevs et al. [2008] introduced a hierarchical data structure called *maximum mipmaps* for efficient rendering with low pre-computation cost. This approach unlocks real-time performance in rendering heightfields with dynamic changes in height values. Snyder and Nowrouzezahrai [2008] showed that one can approximate the visibility function with spherical harmonic bases and obtain real-time performance while rendering soft shadows. Nowrouzezahrai and Snyder [2009] extended this approach to global illumination effects, such as inter-reflections and non-diffuse surfaces, on dynamic heightfields. Despite being efficient, using (low-order) spherical harmonics bases fails in capturing sharp shadows. Thus, Timonen and Westerholm [2010] proposeed an efficient method to determine blocking angles exactly, enabling rendering sharp shadows near real-time. Recently, Jung et al. [2020] combined the idea of maximum mipmaps and dynamic programming to further accelerate ray tracing on heightfields. Other specialized heightfield rendering techniques have also been proposed for computing caustics [Yuksel and Keyser 2009], ambient occlusion [Oat and Sander 2007], and rendering geometric details [Thonat et al. 2021].

Despite a significant amount of work on the forward process of heightfield rendering, the backward pass—differentiable heightfield rendering—has received far less attention. The closest approach to our work is by Zienkiewicz et al. [2016]. They proposed a simplified differentiable heightfield renderer that only considers primary rays and ignores discontinuities in the scene. In contrast, our differentiable heightfield renderer is a general-purpose differentiable path tracer. Our approach handles discontinuities and supports global illumination. We can take derivatives with respect to many scene parameters, such as height values, lighting, and materials, while enjoying the efficiency of heightfield rendering. We believe our work can serve as an important ingredient in many other heightfield optimization tasks, e.g., [Sellán et al. 2020].

2 METHOD

Spatial data structures, such as the maximum-mipmap [Tevs et al. 2008], have traditionally been used to accelerate heightfield rendering. The key idea of our approach is to leverage such a powerful tool used in the *forward* rendering to accelerate the *backward* pass — differentiable heightfield rendering. We enable the backward pass by reparameterizing scene discontinuities (Sec. 2.1). We illustrate how to apply the reparameterization in the context of differentiable heightfield rendering in Sec. 2.2 and apply a mipmap to accelerate the backward passes in Sec. 2.3. We describe how to integrate with a constant memory backpropagation algorithm [Nimier-David et al. 2020] in App. B.

2.1 Reparameterization

Differentiable rendering with respect to the heightfield geometry requires differentiating through an integral (the light transport equation [Pharr et al. 2016]) with integral parameters depending on the scene geometry. We illustrate the main idea of our method by first focusing on a simplified 1D integral and defer the discussion on the full light transport equation in Sec. 2.2.

Let us consider a 1D integral $I(h)$ where the integral bounds $a(h), b(h)$ and the integrand $f(x,h)$ contain discontinuities that depend on the heightfield parameter h.

$$I(h) = \int_{a(h)}^{b(h)} f(x,h)dx \tag{1}$$

We can split the integral domain into M intervals $\{[r_j(h), r_{j+1}(h))\}$, with $r_1(h) = a(h), r_{M+1}(h) = b(h)$. These intervals are computed based on the scene geometry so that each integrand $f_j(x,h)$ within the interval j is continuous.

$$I(h) = \sum_{j=1}^{M} I_j(h) \quad \& \quad I_j(h) = \int_{r_j(h)}^{r_{j+1}(h)} f_j(x,h)dx \tag{2}$$

where $\{f_j\}$ are f restricted to the j-th interval. Computing $\partial I / \partial h$ by using Monte Carlo estimation requires one to estimate derivatives of both the integrand f_j and the integral bound r_j with respect to the scene geometry h. Naively applying automatic differentiation to compute derivatives will miss contributions of the integral bounds r_j as they have zero measure. One could rely on dedicated sampling techniques, such as edge sampling [Li et al. 2018], to estimate the derivative of integral bounds. However, edge sampling scales poorly when the scene involves complex geometric discontinuities. We overcome this limitation by reparameterizing the integral to convert it into an equivalent integral with constant bounds and move the contribution of $r_j(h)$ into the integrand. For each j, we find a differentiable sampling technique $x_j(u,h) = T_j(u,h)$ to map u to $[r_j(h), r_{j+1}(h))$, where u is uniformly distributed on $[0,1)$. In our case, we choose the reparameterization

$$T_j(u,h) = (1-u)r_j(h) + ur_{j+1}(h) \tag{3}$$

The change of variable to T will introduce an additional term, the determinant of the Jacobian $|\det \mathcal{J}_{T_j}(u,h)|$, to accommodate the change in "volume" due to the reparameterization. One can also view it as the reciprocal probability density $1/p_j(x_j,h)$ of sample x_j according to the technique T_j. Thus, incorporating the reparameterization into Eq. (2) results in

$$\begin{aligned} I_j(h) &= \int_0^1 f_j(x_j(u,h),h)\left|\det \mathcal{J}_{T_j}(u,h)\right| du \\ &= \int_0^1 \frac{f_j(x_j(u,h),h)}{p_j(x_j(u,h),h)} du \qquad (4) \end{aligned}$$

$$\frac{\partial I}{\partial h} \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\partial}{\partial h}\frac{f_j(x_j(u,h),h)}{p_j(x_j(u,h),h)} \qquad (5)$$

However, evaluating the derivative accurately for all intervals jointly is impractically expensive because it requires densely sampling each interval. Oftentimes, we are limited to just sampling one interval at a time. In such cases, we randomly select an interval j according to a discrete distribution $p(j,h)$ to obtain the Monte Carlo estimator.

$$\frac{\partial I}{\partial h} \approx \frac{1}{N}\sum_{i=1}^{N} \frac{1}{p(j,h)}\frac{\partial}{\partial h}\frac{f_j(x_j(u,h),h)}{p_j(x_j(u,h),h)} \qquad (6)$$

2.2 Differentiable Rendering on Heightfields

We can apply this gradient estimator in the context of differentiable heightfield rendering. We start with the light transport equation:

$$L_o(\mathbf{x},\omega_o,h) = L_e(\mathbf{x},h) + \int_\Omega L_i(\mathbf{x},\omega_i,h) f(\mathbf{x},\omega_o,\omega_i,h)\, d\omega_i^\perp \qquad (7)$$

where L_i, L_o are the input/output radiance, ω_i, ω_o are the incoming/outgoing lighting directions, $\mathbf{x}$ is a point on the surface, f is the *bidirectional reflectance distribution function* (BRDF) function, $\omega_i^\perp$ is the projected solid angle, and h is the scene parameter, such as geometry, lighting, and material. Usually, we only integrate over the hemisphere that aligns with the surface normal. However, in our method we need to integrate over the entire unit sphere. We will explain why this is required shortly.

Differentiable rendering aims to compute the derivative of outgoing radiance L_o with respect to rendering parameters h. When the integrand is continuous, it is relatively easy to compute the derivatives. We can apply automatic differentiation and use the *radiative backpropagation* method [Nimier-David et al. 2020; Zeltner et al. 2021] or equivalently the *path replay backpropagation* [Vicini et al. 2021b].

$$\begin{aligned} \frac{\partial L_o(\mathbf{x},\omega_o,h)}{\partial h} &= \frac{\partial}{\partial h}\int_\Omega L_i(\mathbf{x},\omega_i,h) f(\mathbf{x},\omega_o,\omega_i,h) d\omega_i^\perp \\ &= \int_\Omega \frac{\partial}{\partial h} L_i(\mathbf{x},\omega_i,h) f(\mathbf{x},\omega_o,\omega_i,h) d\omega_i^\perp \\ &= \int_\Omega L_i(\mathbf{x},\omega_i,h)\frac{\partial f(\mathbf{x},\omega_o,\omega_i,h)}{\partial h} d\omega_i^\perp \\ &\quad + \int_\Omega \frac{\partial L_i(\mathbf{x},\omega_i,h)}{\partial h} f(\mathbf{x},\omega_o,\omega_i,h) d\omega_i^\perp \qquad (8) \end{aligned}$$

We omit $L_e(\mathbf{x},h)$ term here for simplicity. This derivation is valid if both f and L_i are continuous according to the Leibniz rule. With this relationship, we can easily use either method to estimate $\partial L_o/\partial h$ via Monte Carlo integration.

However, L_i often contains discontinuities related to visibility changes such as object silhouettes and shadows. Accurately identifying these discontinuities can be difficult in general 3D mesh rendering. Our observation is that if we parameterize the integral using a spherical coordinate such that the up-direction is aligned with the heightfield, we can reduce the 2D integral over the unit sphere to a nested 1D integral where discontinuities only occur in 1D.

WLOG, assume the scene consists of a single heightfield geometry in which the up direction is aligned with the world space z-axis. We rewrite the integral using world space spherical coordinates, where θ and ϕ are the zenith and azimuth angles respectively,

$$\begin{aligned} L_o(\mathbf{x},\omega_o,h) &= L_e(\mathbf{x},h) \\ &+ \int_0^{2\pi}\int_0^{\pi} \underbrace{L_i(\mathbf{x},\omega_i,h) f(\mathbf{x},\omega_o,\omega_i,h)|\omega_i\cdot\hat{\mathbf{n}}|\ \sin\theta}_{\tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta,h)} d\theta d\phi. \qquad (9) \end{aligned}$$

where $\hat{\mathbf{n}}$ is the surface normal at a surface point $\mathbf{x}$. For any surface point, discontinuities in $L_i(\mathbf{x},\omega_i,h)$ manifest as a series of curves $r_j(\phi,h)$. In other words, the discontinuity curve does not contain a vertical tangent line. If we only consider discontinuities induced by geometric occlusion, this is indeed true. However, there are also other types of discontinuities in rendering. For example, the surface normal jumps discontinuously between texels when we triangulate the heightfield. If nearest neighbor interpolation were used, the BRDF is also discrete along the surface. While our method can be extended to detect these kind of discontinuities, a better approach is to interpolate these features smoothly so that they remain continuous over the entire heightfield. As a result, both normals and BRDF are smooth along the heightfield and no discontinuity curve has a vertical tangent line. Under such conditions, even if points on the discontinuity curve can move in the ϕ dimension, its motion can still be captured by the 1D motion at heightfield vertices. The movement at the end points of each discontinuity curve cannot be captured in 1D as they result in discontinuous change in M w.r.t. to ϕ, where M is the number of continuous intervals (Eq. (2)). However, on a smoothly interpolated heightfield, the end points of one curve always lie on another discontinuity curve. Thus if M changes at θ', $\int \tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta,h)\, d\theta$ evaluates to the same value as $\theta \to \theta'$. In such cases, the integral is continuous and the change in M does not induce a discontinuity in $\tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta,h)$. Thus we can ignore the change in M.

For each ϕ, discontinuity curves separate $[0,\pi)$ into a series of intervals where $L(\mathbf{x},\omega_i,\phi,\theta)$ is continuous w.r.t θ.

$$\begin{aligned} L_o(\mathbf{x},\omega_o,h) &= L_e(\mathbf{x},h) + \int_0^{2\pi}\sum_{j=1}^{M}\int_{r_j(\phi,h)}^{r_{j+1}(\phi,h)} \tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta,h) d\theta d\phi \\ &= L_e(\mathbf{x},h) + \int_0^{2\pi}\sum_{j=1}^{M}\int_0^1 \frac{\tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta(u,h),h)}{p_j(\theta(u,h),\phi,h)} du d\phi \end{aligned} \qquad (10)$$

where $p_j(\theta,\phi,h)$ is the probability density of sample θ according to h and ϕ in the j-th interval. While discontinuity r_j is absent from the bottom equation, determining the reparameterized sampling

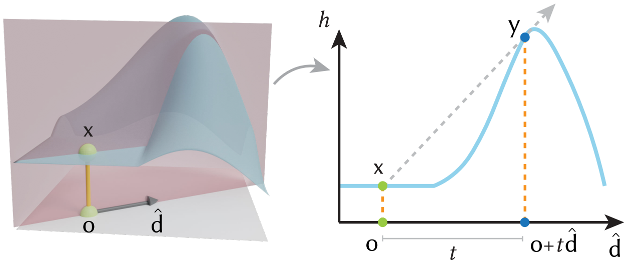


Figure 4: Discontinuities in a heightfield are a subset of the *peaks* in a heightfield. We characterize a peak with respect to a point x with two conditions: (1) line x, y must be a tangent line at y and (2) the curvature of y along direction d̂ must be negative.

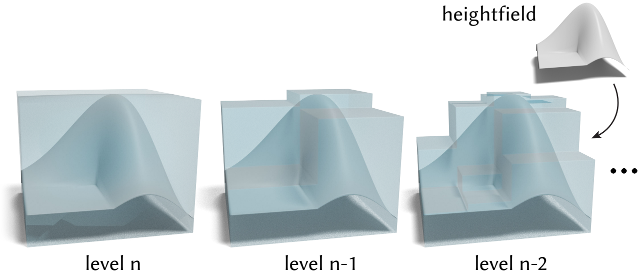


Figure 5: We use the *maximum mipmaps* to accelerate finding discontinuities in the backward pass.

p_j requires knowledge of r_j. In rendering, $\tilde{L_i}$ itself is a recursive integral. Thus evaluating Eq. (10) as is requires exponential complexity w.r.t. path length which is prohibitively expensive. We instead stochastically sample one interval at a time. By using Eq. (6), where we sample the interval according to a probability distribution p_j, we can derive the following estimator.

$$\frac{\partial L_o(\mathbf{x},\omega_o,h)}{\partial h} \approx \frac{\partial L_e(\mathbf{x},h)}{\partial h} + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{p(j,h)}\frac{\partial}{\partial h}\frac{\tilde{L_i}(\mathbf{x},\omega_i,\phi,\theta(u,h),h)}{p_j(\theta(u,h),h)p(\phi)} \tag{11}$$

However, to evaluate this expression, we need to construct the reparameterized sampling strategy $p_j(\theta,\phi,h)$ that depends on the location of discontinuity $r_j(\phi,h)$. Identifying discontinuities $r_j(\phi,h)$ is often the bottleneck when computing $\partial L_o/\partial h$.

2.3 Accelerate Discontinuities Searching

Identifying discontinuities boils down to identifying peaks on the slice of the heightfield along $\hat{\mathbf{d}}$ (see Fig. 4). These peaks can be identified by tracking derivatives. According to the notation in Fig. 4, a peak at distance t along $\hat{\mathbf{d}}$ must satisfy

$$\frac{h(\mathbf{o}+t\hat{\mathbf{d}})-h(\mathbf{o})}{t} = \frac{\partial h(\mathbf{o}+t\hat{\mathbf{d}})}{\partial\hat{\mathbf{d}}} \quad \& \quad \frac{\partial^2 h(\mathbf{o}+t\hat{\mathbf{d}})}{\partial\hat{\mathbf{d}}^2} < 0, \tag{12}$$

where the first condition says that the line from the location $\mathbf{x}$ to the discontinuity $\mathbf{y}$ must be a tangent line at the location of discontinuity $\mathbf{y}$. The second condition indicates that $\mathbf{y}$ should be concave (like a mountain).

However, naively tracking the derivative conditions for all locations is expensive. We can leverage the fact that a lower peak will be blocked by a higher peak and have no effect in the radiance splitting Eq. (10) to accelerate the computation. In Fig. 6, we can see that the peak located at t_2 has no influence on the outgoing radiance at $\mathbf{x}$ because it is blocked by the peak at t_1.

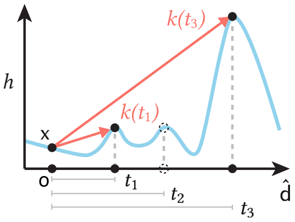


Figure 6: We accelerate the computation by only searching for the discontinuities that have (primary) influence on the incoming radiance. For example, we ignore the peak at t_2 because it is blocked by the peak at t_1 and has no contribution to the incoming radiance.

To identify the peaks that have influence to a location $\mathbf{x}$, we compare the slope from the origin $\mathbf{x}$ to the peak locations. Let t_i, t_j be the horizontal distance to the two peaks that cause discontinuities at $\mathbf{x}$ and $t_i < t_j$. We can compute the slope $k(t)$ from $\mathbf{x} = [\mathbf{o}, h(\mathbf{o})]$ to the location $[\mathbf{o}+t\hat{\mathbf{d}}, h(\mathbf{o}+t\hat{\mathbf{d}})]$

$$k(t) = \frac{h(\mathbf{o}+t\hat{\mathbf{d}})-h(\mathbf{o})}{t} \tag{13}$$

If the peak at t_i does not block the peak at t_j, then they must satisfy $k(t_i) < k(t_j)$. Therefore, once a discontinuity is found at t_i, we can skip the locations where their slopes are smaller than $k(t_i)$. This can be efficiently achieved via maximum mipmaps (Fig. 5). For completeness, we provide pseudocode in Alg. 1. Once we are able to identify the location of discontinuities, we then estimate the derivatives with the method by Nimier-David et al. [2020]. We describe the integration with Nimier-David et al. [2020] in detail in App. A.2 and App. B. We extend our method to handle primary visibility and direct lighting in App. A.1.

3 EXPERIMENTS

We implemented our renderer in Rust using the JIT framework in LuisaRender [Zheng et al. 2022]. To verify the correctness of our method, we compare the derivatives computed with our method against the finite difference approximation. In Fig. 7, we can observe that our efficient lighting and geometry derivatives are correctly computed and converge to finite differences.

We compare the numerical quality of gradient in a similar cone scene. In Fig. A1, we include the gradient images computed by finite differences, Mitsuba 3 [Jakob et al. 2022b], and our method. The reference image was computed using 8192 SPP and the gradient image was computed using 16 SPP. We computed the mean squared error of the gradient image computed by each renderer with its corresponding finite difference image. Under equal sample, our method has 7.7x lower error compared to Mitsuba 3.

We pick a set of inverse rendering tasks to demonstrate the generality of our differentiable heightfield renderer. We show that our method is capable of optimizing the geometry, shadow, and

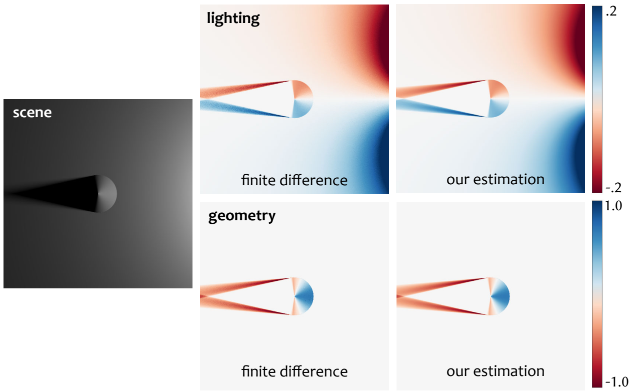


Figure 7: We evaluate the accuracy of our method by comparing against finite differences. On a cone geometry (left), our lighting and geometry gradients (right) converge to the finite difference results (middle). Note that the light source is moving up-and-down w.r.t the image and the geometry is perturbed along the z-axis. The magnitude is the derivative of the pixel intensity w.r.t the lighting/geometry movements.

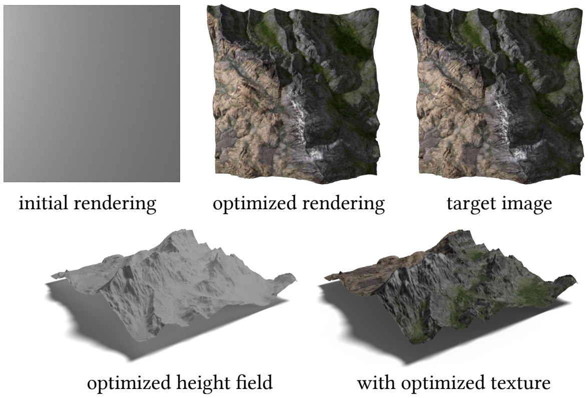


Figure 8: Given a set of target images of a terrain (top right), we optimize a heightfield (top middle) from a plane (top left) to match the target images by changing the geometry and texture. We visualize the optimized terrain model on the bottom row.

texture of the heightfield. We further demonstrate that our method is able to be used jointly with machine learning tasks.

Geometry & Texture Optimization. The simplest task to evaluate our method is to do inverse geometry and texture optimization. Specifically, given a set of target images, we want to optimize a single heightfield from multiple views [Seitz and Kutulakos 1998]. To quantitatively evaluate our method, we design a toy task where ground truth is available. Given textured terrain data T, we render T into a set of target images using a set of lighting and camera parameters $\{\theta_i\}$. Then we use the same rendering parameter θ_i to optimize a heightfield H to match the target images by minimizing the squared L2 pixel difference across all views. Specifically, our optimization can be written as

$$\underset{\mathrm{H}}{\text{minimize}} \sum_{i \in \text{views}} \|\mathrm{R}(\theta_i, \mathrm{H}) - \mathrm{R}(\theta_i, \mathrm{T})\|_2^2 \tag{14}$$

where we use $\mathrm{R}(\theta_i, \mathrm{H})$ to denote the rendered image of the heightfield H under the i-th camera parameter θ_i. We optimize this energy with the ADAM optimizer [Kingma and Ba 2015] and the Laplacian preconditioner [Nicolet et al. 2021]. In Fig. 8, we can observe the effectiveness of our method in turning an initial heightfield (a plane) into a textured terrain. We further visualize the difference between the ground truth terrain and our optimized terrain in Fig. 9.

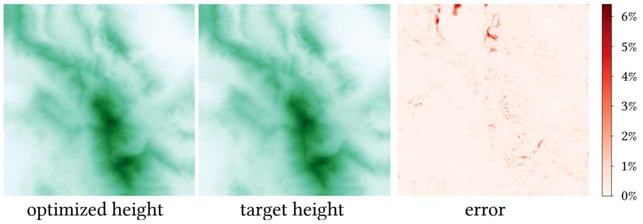


Figure 9: We visualize the error (right) between our optimized heightfield (left) and the ground truth heightfield (middle) in Fig. 8. We can observe that our method faithfully reproduces the target heightfield.

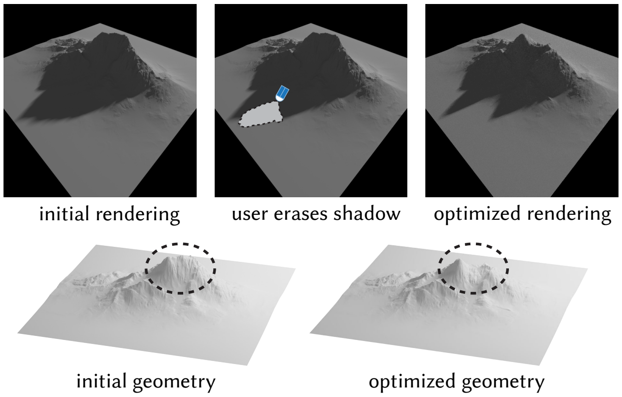
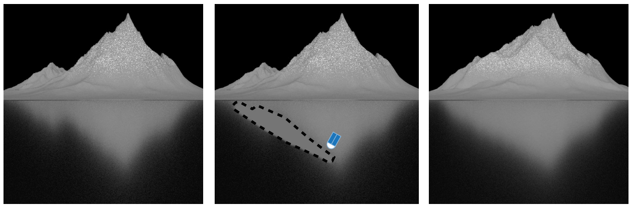


Figure 10: Our approach enables interactive inverse rendering editing. Given a heightfield (top left), the geometry changes interactively (top right) when one erases part of the shadow (top middle). We visualize the results under a different lighting at the bottom.

Geometric Illusions. We use our method to create a geometric illusion. The goal is to generate a single shape whose renderings look like different objects from different views. People have worked on this problem under different settings, such as shadow projection [Mitra and Pauly 2009; Sadekar et al. 2022] or wire geometry [Hsiao et al. 2018]. In our case, we focus on optimizing the heightfield geometry with textures to achieve the illusion. Our setup is similar to the problem considered in [Alexa and Matusik 2010] where they consider illusions that can be achieve with shading changes [Gingold and Zorin 2008; Panozzo and Sorkine-Hornung 2014]. However, the key difference is that our method is optimized under



initial rendering user edits reflection optimized rendering

Figure 11: Our method is able to handle glossy reflection. We can observe the optimized geometry (right) produces a glossy reflection that accurately matches the edited image (middle).

global illumination with complex light transport behavior. In Fig. 1, we provide a target image of "OPEN" I_1 with a manually defined lighting condition θ_1 in the morning and a target image of "CLOSED" I_2 with another lighting condition θ_2 in the evening. Then given an initial heightfield of a plane H, we minimize the squared L2 pixel difference between the rendered heightfield and the target images by changing the geometry of heightfields as

$$\underset{\mathrm{H}}{\text{minimize}} \ \|\mathrm{R}(\theta_1, \mathrm{H}) - \mathrm{I}_1\|_2^2 + \|\mathrm{R}(\theta_2, \mathrm{H}) - \mathrm{I}_2\|_2^2. \tag{15}$$

Similar to Eq. (14), we use ADAM with the Laplacian preconditioner in the optimization.

Shadow Optimization. We use a similar setup to optimize a geometry to cast a shadow of a predefined shape. In Fig. 3, we specify the target image to be a human face and minimize the L2 loss. However, simply optimizing the geometry will result in a non-desirable local minimum where the dark face mainly comes from shading changes, rather than a shadow. Thus, we constrain the left half of the heightfield to be zero so that the dark face can only appear from the shadow cast by the right part of the heightfield.

Shading-Based Editing. The efficiency of our method further enables us to do interactive shadow editing. In Fig. 10, we erase a part of the shadow and we can interactively observe geometry changes. Our method can also optimize glossy reflection. In Fig. 11, we optimize geometry to match user edited reflection. We minimize the L2 loss on the lower half of rendered and edited image by optimizing for the geometry of the mountain. We use 32 samples per pixel in each iteration and the Laplacian preconditioner [Nicolet et al. 2021]. Albeit using uniform BRDF sampling on a highly glossy surface, we are still able to optimize the geometry to closely match the edited reflection.

CLIP Shape Generation. We further show the applicability of our method in learning-based generative tasks. CLIP [Radford et al. 2021] is a pre-trained model that can either encode text or an image into a feature vector. The model is trained in the way that if the text prompt and the image content are similar, then their feature embeddings will be similar. We leverage the power of CLIP to generate a heightfield based on a given text prompt. We set up the optimization to minimize the negative cosine similarity as

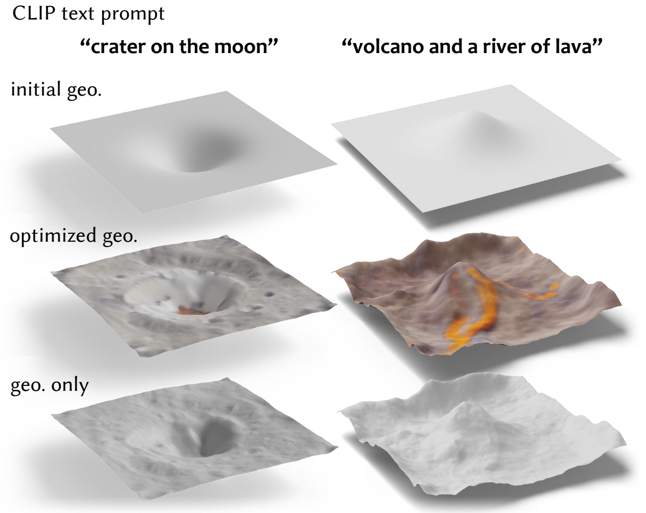


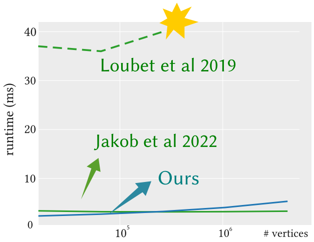
Figure 12: We use our differentiable renderer jointly with CLIP [Radford et al. 2021] to generate a heightfield (second row) given a text prompt (bottom). We also provide the visualization of the initial heightfield (top) and the optimized heightfield without textures (third row).

$$\underset{\mathrm{H}}{\text{minimize}} \sum_{i \in \text{views}} 1 - \mathit{CosSimilarity}\big(f_{\text{text}}, f_{\text{image}}(\mathrm{R}(\theta_i, \mathrm{H}))\big), \tag{16}$$

$$\mathit{CosSimilarity}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \tag{17}$$

where *CosSimilarity* is the cosine similarity function which measures the similarity between the feature of the text prompt f_{text} and the encoded feature of the rendering f_{image}. Similar to the previous experiments, we optimize it with respect to multiple camera views i. We optimize this energy with the network regularization inspired by [Michel et al. 2021]. Specifically, the displacement and the texture of the heightfield is the output of a multilayer perceptron.

4 CONCLUSION & FUTURE WORK



Our method leverages the maximum mipmap technique [Tevs et al. 2008] to accelerate differentiable renderers on heightfield geometry.

As a significant speedup can be obtained with the dedicated compiler proposed in Jakob et al. [2022a] (see inset), we would like to explore using [Jakob et al. 2022a] jointly with our mipmap acceleration to further improve efficiency.

Generalizing to multilevel heightfields [Weiss et al. 2020] and displacement maps on curved geometry [Thonat et al. 2021] will extend real-time differentiable renderers to support a wider variety of geometries. It is also possible to use the spherical coordinate reparameterization to compute derivatives of more general geometry such as triangle meshes. It would require an efficient data structure for plane-triangle or triangle-triangle intersection that is occlusion-aware to detect discontinuities for a given ϕ in spherical coordinates. Incorporating other differentiable representations for the scene, such as spherical harmonics environment maps [Ramamoorthi and Hanrahan 2001], would extend the realm of our method to more scene parameters.

While we discussed in the method section for any point $\mathbf{x}$ on the heightfield surface, the heightfield geometry cannot produce a vertical edge. However, when differentiating primary visibility, the camera is usually not on the surface, vertical edges can still be appear. For example, when the border of the heightfield is perfectly aligned with the camera direction. Extending our method to handle such cases is left as future work. Our prototype implementation produces biased gradients for indirect lighting due to our use of approximate radiance as in Nimier-David et al. [2020]. Further improvement of our method can be obtained by using analytic importance sampling strategies for non-diffuse BRDFs under global spherical coordinates. This would improve the numerical efficiency for sampling highly glossy materials and unify the sampling routine for the forward and backward pass, enabling integration with [Vicini et al. 2021b] to further improve the performance.

If few discontinuities are visible from a viewpoint, the maximum mipmap does not provide a large acceleration because the algorithm cannot skip through the heightfield if no peaks are found. In the worst case where there is no discontinuity, the algorithm has to march along every texel across the heightfield. We analyze this problem and proposed a method to alleviate this issue in App. C. Efficiently detecting and skipping regions that have no discontinuities would further improve scalability.

We notice that the text-based shape generation with CLIP often requires a meaningful initialization to prevent bad local minima. For instance, our initialization in Fig. 12 is a bumpy heightfield, rather than a flat plane. Future work on better regularization or multilevel optimization could further improve the generation.

ACKNOWLEDGMENTS

We would like to thank Otman Benchekroun for sharing his house model in Fig. 1, Peiyu Yu for offering us inspirations for the geometric illusion, and Shaokun Zheng for his help with our automatic differentiation implementation in LuisaRender. Our research is funded in part by NSERC Discovery (RGPIN–2022–04680), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, the DSI Catalyst Grant program and gifts by Adobe Systems.

REFERENCES

Marc Alexa and Wojciech Matusik. 2010. Reliefs as images. *ACM Trans. Graph.* 29, 4 (2010), 60:1–60:7.

John Amanatides and Andrew Woo. 1987. A Fast Voxel Traversal Algorithm for Ray Tracing. In *8th European Computer Graphics Conference and Exhibition, Eurographics 1987, Amsterdam, The Netherlands, August 24-28, 1987, Proceedings*, Guy Maréchal (Ed.). North-Holland / Eurographics Association.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* 39, 6 (2020), 245:1–245:18.

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically differentiating parametric discontinuities. *ACM Trans. Graph.* 40, 4 (2021), 107:1–107:18.

Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2018. Inverse Transport Networks. *CoRR* abs/1809.10820 (2018).

Wenzheng Chen, Huan Ling, Jun Gao, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 9605–9616.

Daniel Cohen-Or, Eran Rich, Uri Lerner, and Victor Shenkar. 1996. A Real-Time Photo-Realistic Visual Flythrough. *IEEE Trans. Vis. Comput. Graph.* 2, 3 (1996), 255–265.

Daniel Cohen-Or and Amit Shaked. 1993. Photo-Realistic Imaging of Digital Terrains. *Comput. Graph. Forum* 12, 3 (1993), 363–373.

Mark Duchaineau, Murray Wolinsky, David E Sigeti, Mark C Miller, Charles Aldrich, and Mark B Mineev-Weinstein. 1997. ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings. Visualization'97 (Cat. No. 97CB36155)*. IEEE, 81–88.

Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2020. Learning Deformable Tetrahedral Meshesfor 3D Reconstruction. In *NeurIPS*.

Yotam I. Gingold and Denis Zorin. 2008. Shading-based surface editing. *ACM Trans. Graph.* 27, 3 (2008), 95.

Christian Henning and Peter Stephenson. 2004. Accelerating the ray tracing of height fields. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. 254–258.

Hugues Hoppe. 1998. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *9th IEEE Visualization Conference, IEEE Vis 1998, Research Triangle Park, North Carolina, USA, October 18-23, 1998, Proceedings*, David S. Ebert, Holly E. Rushmeier, and Hans Hagen (Eds.). IEEE Computer Society and ACM, 35–42.

Kai-Wen Hsiao, Jia-Bin Huang, and Hung-Kuo Chu. 2018. Multi-view wire art. *ACM Trans. Graph.* 37, 6 (2018), 242:1–242:11.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. *Mitsuba 3 renderer.* https://mitsuba-renderer.org.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530099

Dawoon Jung, Fridger Schrempp, and Seunghee Son. 2020. Optimally Fast Soft Shadows on Curved Terrain with Dynamic Programming and Maximum Mipmaps. *CoRR* abs/2005.06671 (2020).

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable Rendering: A Survey. *CoRR* abs/2006.12057 (2020). arXiv:2006.12057 https://arxiv.org/abs/2006.12057

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 3907–3916.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph.* 39, 6 (2020), 194:1–194:14.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6 (2018), 222:1–222:11.

Tzu-Mao Li, Michal Lukác, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.* 39, 6 (2020), 193:1–193:15.

Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nickolas Faust, and Gregory A. Turner. 1996. Real-Time, Continuous Level of Detail Rendering of Height Fields. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, John Fujii (Ed.). ACM, 109–118.

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.* 37, 6 (2018), 221:1–221:11.

Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. 2019b. Beyond Pixel Norm-Balls: Parametric Adversaries using an Analytically Differentiable Renderer. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. 2019a. Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 7707–7716.

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich,* *Switzerland, September 6-12, 2014, Proceedings, Part VII (Lecture Notes in Computer Science, Vol. 8695)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer, 154–169.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* 38, 6 (2019), 228:1–228:14.

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2021. Text2Mesh: Text-Driven Neural Stylization for Meshes. *CoRR* abs/2112.03221 (2021).

Niloy J. Mitra and Mark Pauly. 2009. Shadow art. *ACM Trans. Graph.* 28, 5 (2009), 156.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2021. Extracting Triangular 3D Models, Materials, and Lighting From Images. *CoRR* abs/2111.12503 (2021).

F Kenton Musgrave. 1988. Grid tracing: Fast ray tracing for height fields. *Yale University Dept. of Computer Science Research* (1988).

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Trans. Graph.* 40, 6 (2021), 248:1–248:13.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 146.

Derek Nowrouzezahrai and John M. Snyder. 2009. Fast Global Illumination on Dynamic Height Fields. *Comput. Graph. Forum* 28, 4 (2009), 1131–1139.

Christopher Oat and Pedro V. Sander. 2007. Ambient aperture lighting. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics, SI3D 2007, April 30 - May 2, 2007, Seattle, Washington, USA*, Bruce Gooch and Peter-Pike J. Sloan (Eds.). ACM, 61–64.

Daniele Panozzo and Olga Sorkine-Hornung. 2014. Appearance-mimicking surfaces. *ACM Trans. Graph.* 33, 6 (2014), 216:1–216:10.

Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. 2019. Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer. *CoRR* abs/1903.11149 (2019).

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation.* Morgan Kaufmann.

Huamin Qu, Feng Qiu, Nan Zhang, Arie E. Kaufman, and Ming Wan. 2003. Ray Tracing Height Fields. In *2003 Computer Graphics International (CGI 2003), 9-11 July 2003, Tokyo, Japan.* IEEE Computer Society, 202–209.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763.

Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*, Lynn Pocock (Ed.). ACM, 497–500.

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A Versatile Scene Model with Differentiable Visibility Applied to Generative Pose Estimation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015.* IEEE Computer Society, 765–773.

Stefan Röttger, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. 1998. Real-time generation of continuous levels of detail for height fields. (1998).

Kaustubh Sadekar, Ashish Tiwari, and Shanmuganathan Raman. 2022. Shadow Art Revisited: A Differentiable Rendering Based Approach. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022.* IEEE, 628–636.

Steven M. Seitz and Kiriakos N. Kutulakos. 1998. Plenoptic Image Editing. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV-98), Bombay, India, January 4-7, 1998.* IEEE Computer Society, 17–24.

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of heightfields via rank minimization. *ACM Trans. Graph.* 39, 4 (2020), 109.

John M. Snyder and Derek Nowrouzezahrai. 2008. Fast Soft Self-Shadowing on Dynamic Height Fields. *Comput. Graph. Forum* 27, 4 (2008), 1275–1283.

David C Taylor. 1994. An algorithm for continuous resolution polygonalizations of a discrete surface. In *Proc. Graphics Interface'94.*

Art Tevs, Ivo Ihrke, and Hans-Peter Seidel. 2008. Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics, SI3D 2008, February 15-17, 2008, Redwood City, CA, USA*, Eric Haines and Morgan McGuire (Eds.). ACM, 183–190.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason M. Saragih, Matthias Nießner, Rohit Pandey, Sean Ryan Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. 2020. State of the Art on Neural Rendering. *Comput. Graph. Forum* 39, 2 (2020), 701–727.

Theo Thonat, Francois Beaune, Xin Sun, Nathan Carr, and Tamy Boubekeur. 2021. Tessellation-Free Displacement Mapping for Ray Tracing. 40, 6 (2021).

Ville Timonen and Jan Westerholm. 2010. Scalable Height Field Self-Shadowing. *Comput. Graph. Forum* 29, 2 (2010), 723–731.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021a. Path replay backpropagation: differentiating light paths using constant memory and linear time. *ACM Trans. Graph.* 40, 4 (2021), 108:1–108:14.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021b. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4 (Aug. 2021), 108:1–108:14. https://doi.org/10.1145/3450626.3459804

Yifan Wang, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Trans. Graph.* 38, 6 (2019), 230:1–230:14.

Sebastian Weiss, Florian Bayer, and Rüdiger Westermann. 2020. Triplanar Displacement Mapping for Terrain Rendering. In *41st Annual Conference of the European Association for Computer Graphics, Eurographics 2020 - Short Papers, Norrköping, Sweden, May 25-29, 2020 [online only]*, Alexander Wilkie and Francesco Banterle (Eds.). Eurographics Association, 53–56.

Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph.* 41, 4 (2022), 123:1–123:13.

Cem Yuksel and John Keyser. 2009. Fast real-time caustics from height fields. *Vis. Comput.* 25, 5-7 (2009), 559–564.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo estimators for differential light transport. *ACM Trans. Graph.* 40, 4 (2021), 78:1–78:16.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021. Path-space differentiable rendering of participating media. *ACM Trans. Graph.* 40, 4 (2021), 76:1–76:15.

Shaokun Zheng, Zhiqian Zhou, Xin Chen, Difei Yan, Chuyan Zhang, Yuefeng Geng, Yan Gu, and Kun Xu. 2022. LuisaRender: A High-Performance Rendering Framework with Layered and Unified Interfaces on Stream Architectures. *ACM Trans. Graph.* 41, 6, Article 232 (nov 2022), 19 pages. https://doi.org/10.1145/3550454.3555463

Yang Zhou, Lifan Wu, Ravi Ramamoorthi, and Ling-Qi Yan. 2021. Vectorization for Fast, Analytic, and Differentiable Visibility. *ACM Trans. Graph.* 40, 3 (2021), 27:1–27:21.

Jacek Zienkiewicz, Andrew J. Davison, and Stefan Leutenegger. 2016. Real-time height map fusion using differentiable rendering. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016.* IEEE, 4280–4287.

Algorithm 1: Trace Discontinuities

```
Param. : MipHeight                 // max-mipmap
Input  : $\mathbf{p}$                 // point location
         $\hat{\mathbf{d}}$                 // tracing direction
         $h$                 // heightfield function $h : \mathbb{R}^2 \rightarrow \mathbb{R}$
         $k_{\min}$                 // slope of the previous discontinuity
Output : $ts$                 // list of distances to discontinuities
$t = 0$
$ts = []$
while $(\mathbf{p} + t\hat{\mathbf{d}})$ is in the heightfield do
    if $MipHeight(\mathbf{p} + t\hat{\mathbf{d}}) > h(\mathbf{p}) + k_{min}t$ then
        if $level == 0$ then
            $k(t) = (h(\mathbf{p} + t\hat{\mathbf{d}}) - h(p))/t$
            $t_{prev}$ = distance to previous edge
            $t_{next}$ = distance to next edge
            $k_{prev} = (h(\mathbf{p} + t\hat{\mathbf{d}}) - h(\mathbf{p} + t_{prev}\hat{\mathbf{d}}))/(t - t_{prev})$
            $k_{next} = (h(\mathbf{p} + t_{next}\hat{\mathbf{d}}) - h(\mathbf{p} + t\hat{\mathbf{d}}))/(t_{next} - t)$
            if $k(t) < k_{prev}$ and $k(t) > k_{next}$ then
                Add $t$ to $ts$
                $k_{min} = k(t)$
        else
            Decrease level
            continue
    $t = t_{next}$    // March a step on mipmap [Tevs et al. 2008]
    if $\mathbf{p} + t\hat{\mathbf{d}}$ is at Mip Cell Boundary then
        Increase level
return $ts$
```

A IMPLEMENTATION DETAILS

Our implementation separates the computation of the forward and backward pass. We compute forward rendering using a traditional path tracer. The backward pass is handled via our method. This not only ensures that all memory usage is local to each thread and ephemeral, but also the forward render does not suffer from increased noise due to suboptimal sampling in the backward pass.

We performed our experiments on an RTX 3070 Ti Laptop GPU with 8GB of VRAM. For the performance comparison, we use the terrain heightfield in Fig. 8and measure the average time per sample for each renderer to compute the gradients of height values under direct illumination. We use the ADAM optimizer [Kingma and Ba 2015] with learning rate of $5 \cdot 10^{-3}$ with 1200 iterations for the inverse rendering task.

A.1 Primary Visibility & Direct Lighting

The computation of primary visibility includes performing area sampling on each pixel to generate a primary ray. The color for pixel p can be written as

$$I_p = \int_{h_p(q)\neq 0} h_p(q)L(q)dq$$

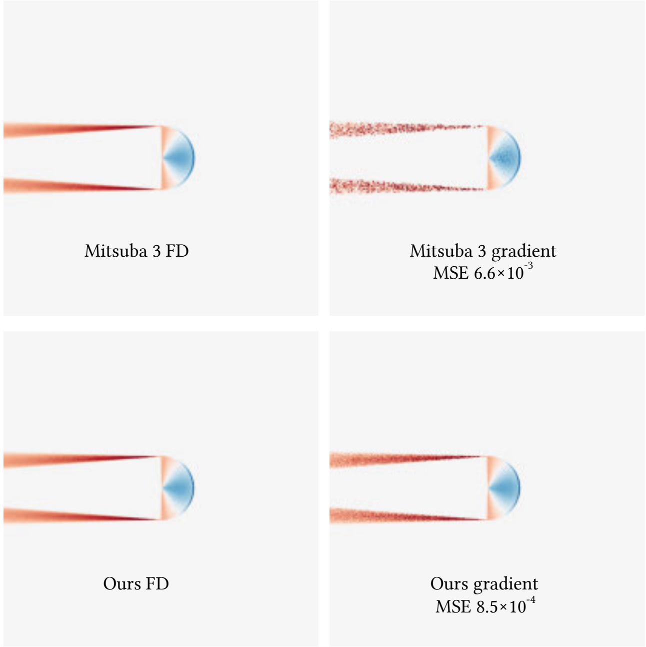


Figure A1: We compute the variance of gradient estimation with Mitsuba 3 under same sample count. Our method is able to capture discontinuity induced gradients very efficiently.

where p and q are pixel coordinates, $h_p(q)$ is the value of the pixel filter centered at p and $L(q)$ is the radiance returned from Eq. (7)by tracing a primary ray starting at q. To use our method, we need to convert the area sampling of each pixel into a sampling of spherical coordinates. For each pixel, we first determine the integral domain along ϕ axis. After sampling a ϕ, we construct a vertical plane in the ϕ direction and then intersect the plane with the image plane to determine the bounds in θ. Once the integral bound in θ is identified, we can find discontinuities within the θ interval and perform reparameterization. This sampling method is not limited to sample pixels but can be applied to sample an arbitrary shape, as long as we can determine the bounds in θ after a ϕ is sampled. Direct lighting involving area lights can be handled in a similar way.

A.2 BRDF Sampling

While it is tempting to reuse the importance sampling strategy for forward rendering in the backward pass, this approach is incompatible with our method. Forward BRDF importance sampling strategy is performed under solid angle measure in local frames while our method requires separate sampling of ϕ and θ under global spherical coordinates. Reusing existing BRDF sampling strategies requires analytic evaluation of the marginal density of ϕ and subsequently importance sampling the conditional density $p(\theta|\phi)$. It is unknown how to perform this sampling analytically for non-diffuse BRDFs. We instead uniformly sample a ϕ and attempt to find discontinuities on θ on the entire $[0, \pi)$ and sample a reparameterized θ afterward. Due to geometry occlusion, this effectively reduces to uniform sampling of the hemisphere in the local frame. However, it does not introduce BRDF parameter-dependent sampling. Thus we can integrate our method under the framework proposed by Nimier-David et al. [2020].

B BACKPROPAGATING GRADIENTS

The most straightforward way to integrate our method to compute gradients is to write a forward rendering pass using our reparameterization and use automatic differentiation on the entire forward pass. However, this comes at a great memory cost as the entire computation graph has to be stored. Instead, we integrate our method with reparameterized radiative backpropagation [Nimier-David et al. 2020], [Zeltner et al. 2021], which only requires constant memory.

$$\begin{aligned}
&\frac{\partial L_o(\mathbf{x},\omega_o)}{\partial h}\\
&= \int_\Omega \frac{\partial}{\partial h} L_i(\mathbf{x}, R(\omega_i,h)) f(\mathbf{x},\omega_o,R(\omega_i,h)) |J_R(\omega_i,h)| d\omega_i^\perp\\
&= \int_\Omega L_i(\mathbf{x}, R(\omega_i,h)) \frac{\partial f(\mathbf{x},\omega_o,R(\omega_i,h))}{\partial h} d\omega_i^\perp\\
&+ \int_\Omega \frac{\partial L_i(\mathbf{x}, R(\omega_i,h))}{\partial h} f(\mathbf{x},\omega_o,R(\omega_i,h)) d\omega_i^\perp\\
&+ \int_\Omega L_i(\mathbf{x}, R(\omega_i,h)) f(\mathbf{x},\omega_o,R(\omega_i,h)) \frac{\partial |J_R(\omega_i,h)|}{\partial h} d\omega_i^\perp
\end{aligned}$$

where $R(\omega_i, h)$ is the reparameterization that removes discontinuities in the integrand. As previously discussed, since our method performs non-parameter-dependent uniform BRDF sampling, no additional secondary reparameterization is required [Zeltner et al. 2021]. The reparameterization $R(\omega_i, h)$ in our case is the same as the sampling strategy. Thus the gradient of Jacobian $\partial_h |J_R(\omega_i, h)|$ is simply the gradient of reciprocal pdf $1/p_j(\theta(u,h),\phi,h)$ in Eq. (11)

C IMPROVING WORST CASE PERFORMANCE

As mentioned in Sec. 4, for heightfields with few discontinuities our method can reach its worst case where every texel on the heightfield has to be checked along a direction. We provide a brief analysis of such cases and propose a preliminary method for alleviating the performance issue.

The heart of this problem boils down to the fact that our discontinuity searching is not a point query in contrast with ray tracing. Instead of querying the intersection point in a specific direction ω_i, our discontinuity searching queries the scene on a range of values: for primary visibility and direct lighting, once we project the integral domain onto spherical coordinates and sampled ϕ, we search for discontinuities in a range of θ, which in the worst case is $[0, \pi)$, similarly for global illumination. When the range is large and the scene doesn't provide enough discontinuities for acceleration, the cost of discontinuity searching can be much higher than ray tracing as the number of texels visited is *linear* w.r.t height field resolution compared to the logarithmic cost of ray tracing. We include an example of such scenes to illustrate the effect in Fig. A2.

Fortunately, if the noise in the gradient is not the biggest concern for the application, then there is a method to accelerate the discontinuity searching so that it has similar computation cost as ray tracing, at a cost of increased variance. In Eq. (11), our method builds on the fact that we can reparameterize the integral by splitting the

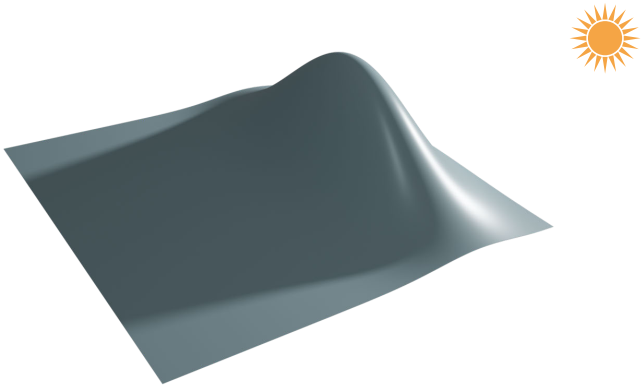


Figure A2: We construct a scene where our method achieves its worst-case complexity. The scene consists of a smooth bump where the only discontinuities are around the peak. We create a *very large* area light source such that discontinuity searching has to march through the entire heightfield.

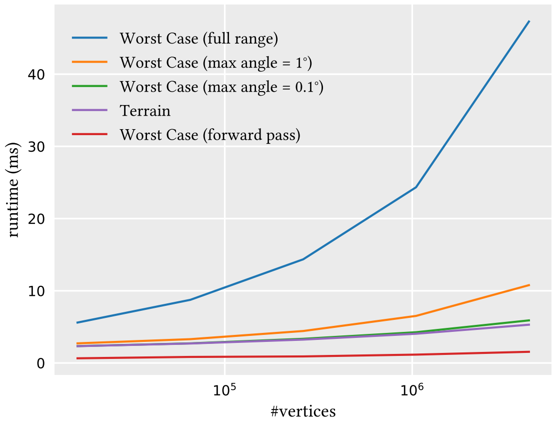


Figure A3: We compare the performance of the backward pass on the worst case scene in Fig. A2 using different θ search ranges. Here, "max angle=1°" means that each subrange of θ is no more than 1°. The asymptotic complexity of discontinuity searching is linear w.r.t #vertices compared to $\log(\#vertices)$ of ray tracing in forward pass. However, by decreasing the maximum search range for θ, we can dramatically improve running time (orange) and even obtain a similar running time (green) as in average case scenes such as the terrain in Fig. 8(purple).

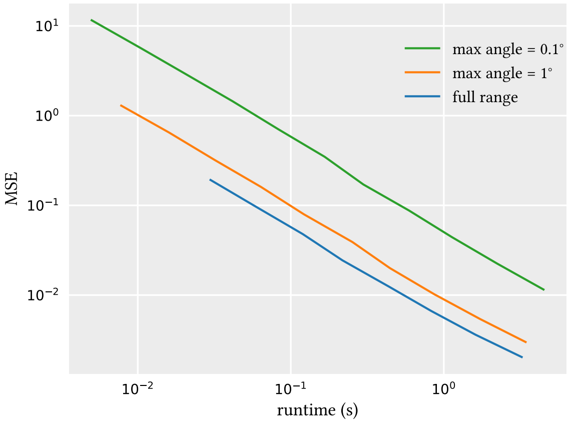


Figure A4: We compare the variance vs time plot for different θ search ranges. Full-range discontinuity searching (blue) provides the best numerical quality in gradient estimation at a cost of slowest per-sample performance. Reducing the search range (orange) can dramatically improve performance without introducing too much noise. Further reducing the search range can improve the performance even more at a cost of a much higher variance (green).

integrand into continuous intervals. However, the splitting point does not have to be a discontinuity. We can even split the integrand at points where it is continuous. Therefore, instead of searching for discontinuities on the full range of θ, we first divide it into multiple subranges and only search for discontinuities on one stochastically chosen subrange. This reduces the computation cost for producing one sample but increases variance as a discontinuity is more likely to be missed. We include some preliminary results demonstrating this performance-variance trade-off in Fig. A3 and A4. It will be interesting to investigate whether we can importance sample such intervals before discontinuity searching to reduce variance while improving performance at the same time.