Plateau-Reduced Differentiable Path Tracing

Michael Fischer [1] **Tobias Ritschel** [1]

Abstract

Differentiable renderers provide gradients w.r.t. arbitrary scene parameters, but the mere existence of these gradients does not guarantee useful update steps in an optimization. Instead, inverse rendering might not converge due to plateaus, *i.e.*, regions of zero gradient, in the objective function. We propose to alleviate this by convolving the rendering equation with an additional kernel that blurs the parameter space. We describe two Monte Carlo estimators to compute plateau-reduced gradients efficiently, *i.e.*, with low variance, and show that these translate into net-gains in optimization error and runtime. Our approach is a straightforward extension to both black-box and differentiable renderers and enables optimization of problems with intricate light transport, such as caustics or global illumination, that existing differentiable renderers do not converge on. Our code is at `https://github.com/mfischer-ucl/prdpt`.

1. Introduction

Regressing scene parameters from 2D observations is a task of significant importance in graphics and vision, but also a hard, ill-posed problem. When all rendering steps are differentiable, we can derive gradients of the final image w.r.t. the scene parameters. However, differentiating through the discontinuous rendering operator is not straightforward due to, *e.g.*, occlusion. The two prevalent rendering approaches are path tracing and rasterization.

Path-tracing aims to solve the rendering equation by computing a Monte Carlo (MC) estimate for each pixel. Unfortunately, MC is only compatible with modern Automatic Differentiation (AD) frameworks under continuous integrands, *e.g.*, color, but not for spatial derivatives.

[1]Department of Computer Science, University College London, United Kingdom. Correspondence to: Michael Fischer <m.fischer@cs.ucl.ac.uk>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023.

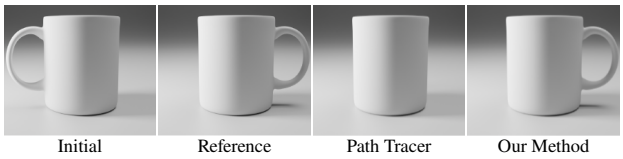

Figure 1. Optimization results with a differentiable path tracer (we use Mitsuba 3 (Jakob et al., 2022)) and our proposed method. The task is to rotate the cup around its z-axis to match the reference. Due to a plateau in the objective function (when the handle is occluded by the cup), regular methods do not converge.

To alleviate this, previous work uses re-sampling of silhouette edges and integrand reparametrizations (Li et al., 2018; Loubet et al., 2019), which enable the optimization of primitive- or light positions. For rasterization, differentiability is achieved by replacing discontinuous edge- and z-tests with hand-crafted derivatives (Loper & Black, 2014; Rhodin et al., 2015; Liu et al., 2019; Le Lidec et al., 2021). Unfortunately, rasterization by design does not capture complex light transport effects, *e.g.*, global illumination or caustics.

However, Metz et al. (2021) show that the mere existence of gradients is no guarantee for an optimization's convergence. In fact, there are surprisingly many cases where they *do not* lead to a successful optimization, due to a plateau in the objective function. An example is the optimization of the mug's rotation in Fig. 1: As soon as the handle disappears behind the cup, no infinitesimally small rotation change will result in a reduced loss. We have hence reached a plateau in the objective function, *i.e.*, a region of zero gradients.

To alleviate this, we take inspiration from differentiable rasterization, where discontinuities are replaced by smooth approximations (Liu et al., 2019). This makes the edge- and z-tests continuous and hence differentiable, and in passing (and much less studied) removes plateaus. In this work, we aim to find a way to apply the same concept to complex light transport. We thus path-trace an alternative, smooth version of the Rendering Equation (RE), which we achieve by convolving the original RE with a smoothing kernel. Our proposed method is a lightweight extension to (differentiable) path tracers that extends the infinitely-dimensional path integral to the product space of paths and scene parameters. We further show that the resulting double integral can be MC-solved efficiently through our derived variance reduction techniques, importance- and antithetic sampling.

2. Background

2.1. Rendering equation

The RE (Kajiya, 1986) defines a pixel P as

$$P(\theta) = \int_\Omega f(\mathbf{x}, \theta) \mathrm{d}\mathbf{x}\,, \tag{1}$$

an integral of the scene function $f(\mathbf{x}, \theta)$, that depends on scene parameters $\theta \in \Theta$, over all light paths $\mathbf{x} \in \Omega$. In *inverse rendering*, we aim to find the parameters θ^* that best explain the pixels P_i in the reference image with

$$\theta^* = \arg\min_\theta \sum_i \mathcal{L}\left(P_i(\theta) - P_i(\theta_{\text{ref}})\right)\,, \tag{2}$$

where $\mathcal{L}$ is the objective function and $P_i(\theta_{\text{ref}})$ are the target pixels created by the (unknown) parameters θ_{ref}.

We will now discuss the two predominant ways to differentiably solve the RE: path tracing, which can simulate all forms of complex light transport but suffers from plateaus (Sec. 2.2), and rasterization, which removes plateaus but is limited to simple once-bounce light transport (Sec. 2.3). Other renderers, *e.g.*, volumetric rendering (Henzler et al., 2019; Mildenhall et al., 2021) or neural rendering (Nalbach et al., 2017; Sitzmann et al., 2021) are mostly limited to simple, approximate light transport or come with their own drawbacks (*e.g.*, static scene constraints) and thus are not within the scope of this work.

2.2. Path tracing

As there is no closed-form solution to Eq. 1, path tracing uses MC to estimate the integral by sampling the integrand at random paths $\mathbf{x}_i$:

$$\widehat{P} \approx \frac{1}{N} \sum_i f(\mathbf{x}_i, \theta) \tag{3}$$

We are interested in the partial derivatives of P with respect to the scene parameters θ, *i.e.*,

$$\frac{\partial P}{\partial \theta} = \frac{\partial}{\partial \theta} \int_\Omega f(\mathbf{x}, \theta) \mathrm{d}\mathbf{x} = \int_\Omega \frac{\partial}{\partial \theta} f(\mathbf{x}, \theta) \mathrm{d}\mathbf{x}\,. \tag{4}$$

In order to make Eq. 4 compatible with automatic differentiation, Li et al. (2018) propose a re-sampling of silhouette edges and Loubet et al. (2019) suggest a re-parametrization of the integrand. Both approaches allow to MC-estimate the gradient as

$$\widehat{\frac{\partial P}{\partial \theta}} \approx \frac{1}{N} \sum_i^N \frac{\partial}{\partial \theta} f(\mathbf{x}_i, \theta)\,. \tag{5}$$

This is now standard practice in modern differentiable rendering packages (Nimier-David et al., 2019; Li et al., 2018; Zeltner et al., 2021; Zhang et al., 2021; 2020), none of which attempt to actively resolve plateaus.

2.3. Rasterization

Rasterization is often used in practical applications due to its simplicity and efficiency, but lacks the ability to readily compute complex light transport phenomena, as it solves a simplified version of the RE, where for every pixel, the light path length is limited to one.

A rasterizer projects the a scene's primitives to screen space and then resolves occlusion. Both these operations are inherently non-differentiable due to jump discontinuities, which therefore – in order to backpropagate gradients through them – need to be replaced with smooth approximations, *e.g.*, a Sigmoid. For a survey on differentiable rasterization and the used employed smoothing approximations, we refer to (Kato et al., 2020) and (Petersen et al., 2022).

Choosing smoothing functions with infinite support (for instance, the Sigmoid), implicitly resolves the plateau problem as well. Our method (Sec. 3) draws inspiration from this concept of "differentiating through blurring". However, most differentiable rasterizers make simplifying assumptions, *e.g.*, constant colors, the absence of shadows or reflections, and no illumination interaction between objects. We will see in later examples that this leads to non-convergence in multi-bounce light transport scenarios, *e.g.*, the optimization from a scene's global illumination. Our formulation, in contrast, does not make such assumptions.

3. Plateau-reduced Gradients

As differentiable rasterization (cf. Sec. 2.3) has established, the blurring of primitive edges is a viable means for differentiation. But what if there is no "primitive edge" in the first place, as we deal with general light paths instead of simple rasterization? The edge of a shadow, for instance, is not optimizable itself, but the result of a combination of light position, reflection, occlusion, etc. Therefore, to achieve an effect similar to that of differentiable rasterizers, we would need a method that blurs the entire light path (instead of just primitive edges) over the parameter space θ. If this method used a blur kernel with infinite support (*e.g.*, a Gaussian distribution), the plateau in the objective would vanish, as a small parameter change in any direction would induce a change in the objective function.

In Fig. 2, we again aim to optimize the cup's rotation around its z-axis to have the handle point to the right, a 1D problem. As we have seen previously, using an image-based objective function leads to a plateau in the "hard" optimization setting, *i.e.*, without blur (the blue line in the plot). Blurring the cup's rotation parameter, on the other hand, leads to θ continuously influencing the value of the objective and therefore resolves the plateau (orange line in the plot). Naturally, it is easy to descend along the gradient of the orange curve, while the gradient is zero on the plateau of the blue curve.

Figure 2. Optimizing the cup's rotation in the hard (left, blue) and smooth (right, orange) setting (note the blurred handle). The image-space loss is displayed on the right: blurring removes the plateau.

3.1. The Plateau-reduced Rendering Equation

We realize our blurring operation as a convolution of the RE (Eq. 1) with a blur kernel κ over the parameter space Θ:

$$Q(\theta) = \kappa \star P(\theta) = \int_\Theta \kappa(\tau) \int_\Omega f(\mathbf{x}, \theta - \tau)\, \mathrm{d}\mathbf{x}\mathrm{d}\tau = \int_{\Theta\times\Omega} \kappa(\tau) f(\mathbf{x}, \theta - \tau)\, \mathrm{d}\mathbf{x}\mathrm{d}\tau\,. \quad (6)$$

While $\kappa(\tau)$ could be any symmetric monotonous decreasing function, we here choose a Gaussian kernel. The kernel acts as a weighting function that weights the contribution of parameters θ that were offset by $\tau \in \Theta$. This means that, in addition to integrating all light paths $\mathbf{x}$ over Ω, we now also integrate over all parameter offsets τ in Θ. Note that we do not convolve across the path space Ω but across the parameter space θ, *e.g.*, the cup's rotation in Fig. 2. Analogous to Eq. 3 and Eq. 4, we can estimate the integral in Eq. 6 through an MC estimator and take its derivative as

$$\widehat{\frac{\partial Q}{\partial \theta}} = \frac{\partial}{\partial \theta} \frac{1}{N} \sum_{i=1}^{N} \kappa(\tau_i) f(\mathbf{x}_i, \theta - \tau_i)\,. \quad (7)$$

Due to the linearity of differentiation and convolution, there are two ways of computing Eq. 7: one for having a differentiable renderer, and one for a renderer that is not differentiable. We discuss both options next.

Plateau-reduced gradients if P is differentiable With access to a differentiable renderer (*i.e.*, access to $\partial P/\partial\theta$), we can rewrite Eq. 7 as

$$\widehat{\frac{\partial Q}{\partial \theta}} = \frac{1}{N} \sum_{i=1}^{N} \kappa(\tau_i) \underbrace{\frac{\partial P}{\partial \theta}(\theta - \tau_i)}_{\text{Diff. Renderer}}\,. \quad (8)$$

Therefore, all that that needs to be done is to classically compute the gradients of a randomly perturbed rendering and weight them by the blur kernel.

Plateau-reduced gradients if P is not differentiable In several situations, we might not have access to a differentiable renderer, or a non-differentiable renderer might have advantages, such as computational efficiency or advanced rendering features. Our derivation also supports this case, as we can rewrite Eq. 7 as

$$\widehat{\frac{\partial Q}{\partial \theta}} \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{\partial \kappa}{\partial \theta}(\tau_i)}_{\text{Diff. Kernel}} \underbrace{P(\theta - \tau_i)}_{\text{Renderer}}\,, \quad (9)$$

which equals sampling a non-differentiable renderer and weighting the result by the gradient of the blur kernel. This is possible due to the additional convolution we introduce: prior work (Li et al., 2018; Loubet et al., 2019) must take special care to compute derivatives (Eq. 5), as in their case, optimizing θ might discontinuously change the pixel integral. We circumvent this problem through the convolution with κ, which ensures that, in expectation, θ *continuously* influences the pixel integral.

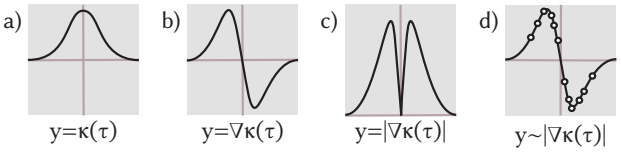

Figure 3. Our kernel κ (a), its gradient $\nabla\kappa$ (b), the positivized gradient (c) and samples drawn proportional thereto (d).

3.2. Variance Reduction

Drawing uniform samples from $\Theta \times \Omega$ will result in a sample distribution that is not proportional to the integrand and hence lead to high-variance gradient estimates and ultimately slow convergence for inverse rendering. In our case, the integrand is the product the kernel κ and the scene function f, which Veach (Veach, 1998) showed how to optimally sample for. As we generally consider the rendering operator a black box, we can only reduce variance by sampling according to the remaining function, the (differentiated) kernel κ (Fig. 3b).

While importance-sampling for a Gaussian ($\tau_i \sim \kappa$, required to reduce variance of Eq. 8) is easily done, importance-sampling for the gradient of a Gaussian ($\tau_i \sim \partial\kappa/\partial\theta$, to be applied to Eq. 9) is more involved, as the gradient of our kernel

$$\frac{\partial \kappa}{\partial \theta}(\tau) = \frac{-\tau}{\sigma^3\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma^2}\right) \quad (10)$$

is negative for $\tau > 0$. We enable sampling proportional to its Probability Density Function (PDF) by "positivization" (Owen & Zhou, 2000), and hence sample for $|\frac{\partial \kappa}{\partial \theta}(\tau)|$ instead (Fig. 3c). The kernel's closed-form PDF allows us to use the inverse Cumulative Distribution Function (ICDF) method for importance sampling. The ICDF of Eq. 10 is

$$F^{-1}(\xi) = \sqrt{-2\sigma^2 \log(\xi)}\,,$$

into which we feed uniform random numbers $\xi \in (0, 1)$ that generate samples proportional to $|\frac{\partial \kappa}{\partial \theta}(\tau)|$ (Fig. 3d).

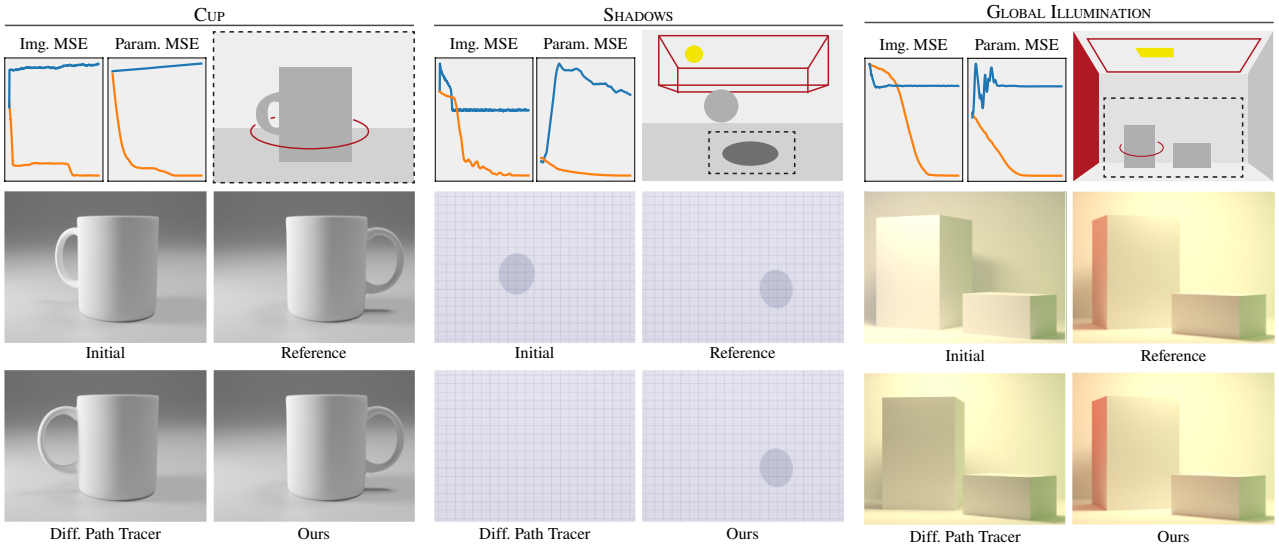

Figure 4. We show the optimization tasks and results for $\texttt{Our}_{\partial\kappa P}$ ("Ours", Eq. 9, orange) and our baseline Mitsuba 3 ("Diff. Path Tracer", blue). The right upper subfigure in each column depicts the scene setup, with the optimizable parameters colored in red. All methods operate in image space only. We additionally plot parameter-space error as a quality control metric.

In order to obtain a zero-variance estimator, we allocate an equal number of samples to the positivized and regular parts (Owen & Zhou, 2000). As the function is point symmetric around the origin, we use antithetic sampling to do so (Hammersley & Mauldon, 1956), *i.e.*, for each sample τ, we additionally generate its negated counterpart $-\tau$.

Moreover, we decrease the amount of smoothing, the kernel's bandwidth σ, over time. For more information on implementation as well as an outline of our plateau-reduced gradient computation, please cf. Appendix A.

4. Results

We analyze our method and its variants on a variety of tasks that feature advanced light transport, plateaus and ambiguities (for detailed task descriptions, cf. Appendix B, for task visualizations, cf. Fig. 4). We compare our method (Eq. 9) against Mitsuba3 (Jakob et al., 2022) and show numerical comparisons against other methods in Appendix B. We run all methods for the same number of iterations and with the same settings.

Performance As is evident from Fig. 4, our method converges reliably on tasks where regular differentiation does not converge due to plateaus in the objective. This is due to our stochastic smoothing of the parameter space, which effectively allows the optimizer to descend a smoother version of the loss landscape. Importantly, we effectively handle complex light transport, as is evident from the rightmost column in Fig. 4, where out-of-view objects in the scene scene are optimized solely from the global illumination scattered from the walls. For a more detailed discussion of our results on each task, cf. Appendix B.

Timing Additionally, we have found that our approach's runtime on average is $8\times$ faster than differentiable rendering with Mitsuba, as our stochastic gradient estimation through the derivative-kernel allows us to skip the gradient computation step of the renderer, which leads to significant savings in computation time. Moreover, as we do not need special treatment of discontinuities (e.g., re-parametrization), we can use Mitsuba's regular forward path tracer at no additional runtime cost.

5. Discussion and Conclusion

Relation to Variational Optimization Indeed, the formalism developed in Sec. 3.1 can be interpreted as a form of variational optimization (Staines & Barber, 2012; 2013), where one would descend along the (smooth) variational objective instead of the true underlying function. As such, Eq. 9 can be seen as an instance of a score-based gradient estimator (Sutton et al., 1999), while Eq. 8 can be interpreted as reparametrization gradient (Kingma et al., 2015; Schulman et al., 2015). Suh et al. (2022) provide intuition on each estimator's performance and align with our findings of the score-based estimator's superiority under a discontinuous objective. It is one of the contributions of this work to connect these variational approaches with inverse rendering.

Conclusion In summary, we have proposed a method for inverse rendering that convolves the rendering equation with a smoothing kernel, which allows straight-forward differentiation and removes plateaus. The idea combines strengths of differentiable rasterization and differentiable path tracing. Our approach is simple to implement, efficient, has theoretical justification and optimizes tasks that existing differentiable renderers so far have diverged upon.

References

Hammersley, J. M. and Mauldon, J. General principles of antithetic variates. In *Mathematical proceedings of the Cambridge philosophical society*, volume 52, 1956.

Henzler, P., Mitra, N., and Ritschel, T. Escaping plato's cave using adversarial training: 3d shape from unstructured 2d image collections. In *Proc. ICCV*, 2019.

Jakob, W., Speierer, S., Roussel, N., Nimier-David, M., Vicini, D., Zeltner, T., Nicolet, B., Crespo, M., Leroy, V., and Zhang, Z. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

Kajiya, J. T. The rendering equation. In *Proc. SIGGRAPH*, 1986.

Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., and Gaidon, A. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Le Lidec, Q., Laptev, I., Schmid, C., and Carpentier, J. Differentiable rendering with perturbed optimizers. *NeurIPS*, 34, 2021.

Li, T.-M., Aittala, M., Durand, F., and Lehtinen, J. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans Graph.*, 37(6), 2018.

Liu, S., Li, T., Chen, W., and Li, H. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proc. ICCV*, 2019.

Loper, M. M. and Black, M. J. Opendr: An approximate differentiable renderer. In *Proc. ECCV*, 2014.

Loubet, G., Holzschuch, N., and Jakob, W. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Tran Graph.*, 38(6), 2019.

Metz, L., Freeman, C. D., Schoenholz, S. S., and Kachman, T. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1), 2021.

Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H.-P., and Ritschel, T. Deep shading: convolutional neural networks for screen space shading. *Comp. Graph. Forum*, 36(4), 2017.

Nimier-David, M., Vicini, D., Zeltner, T., and Jakob, W. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 38(6), 2019.

Owen, A. and Zhou, Y. Safe and effective importance sampling. *J of the American Statistical Association*, 95 (449), 2000.

Petersen, F., Goldluecke, B., Borgelt, C., and Deussen, O. Gendr: A generalized differentiable renderer. In *Proc. CVPR*, 2022.

Rhodin, H., Robertini, N., Richardt, C., Seidel, H.-P., and Theobalt, C. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proc. ICCV*, 2015.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

Sitzmann, V., Rezchikov, S., Freeman, B., Tenenbaum, J., and Durand, F. Light field networks: Neural scene representations with single-evaluation rendering. *NeurIPS*, 34, 2021.

Staines, J. and Barber, D. Variational optimization. *arXiv preprint arXiv:1212.4507*, 2012.

Staines, J. and Barber, D. Optimization by variational bounding. In *ESANN*, 2013.

Suh, H. J., Simchowitz, M., Zhang, K., and Tedrake, R. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pp. 20668–20696. PMLR, 2022.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Veach, E. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

Zeltner, T., Speierer, S., Georgiev, I., and Jakob, W. Monte carlo estimators for differential light transport. *ACM Trans Graph*, 40(4), 2021.

Zhang, C., Miller, B., Yan, K., Gkioulekas, I., and Zhao, S. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4), 2020.

Zhang, C., Dong, Z., Doggett, M., and Zhao, S. Antithetic sampling for monte carlo differentiable rendering. *ACM Trans. Graph.*, 40(4), 2021.

A. Implementation Details

A.1. Adaptive Bandwidth

Adjusting σ, the kernel's spread, controls over how far from the current parameter θ our samples will be spaced out. A high σ may be useful in the early stages of the optimization, when there still is a considerable difference between θ and θ_{ref}, whereas we want a low σ towards the end of the optimization to zero-in on θ_{ref}. Throughout the optimization, we hence decay the initial σ_0 according to a linear schedule, *i.e.*, $\sigma_{t+1} = \sigma_0 - t(\sigma_0 - \sigma_m)$, where σ_m is a fixed minimum value we choose to avoid numerical instabilities that would otherwise arise from $\sigma \to 0$ in, *e.g.*, Eq. 10.

A.2. Implementation

We outline our gradient estimator in pseudo-code in Alg. 1. We importance-sample for our kernel with zero variance, use antithetic sampling and adapt the smoothing strength via σ. As Alg. 1 shows, our method is simple to implement and can be incorporated into existing frameworks with only a few lines of code. We implement our method in PyTorch, with Mitsuba as rendering backbone, and use Adam as our optimizer.

Algorithm 1 Gradient estimation of the scene function f at parameters θ with perturbations $\tau \sim \mathcal{N}(0, \sigma)$ at N samples.

```
# Equation 9
procedure EstimateGradient(P, θ, σ, N)
    G := 0
    for i ∈ (1, N/2) do
        ξ ← Uniform(0, 1)
        τ ← sqrt(−2σ² log(ξ))
        G ← G + P(θ + τ) − P(θ − τ)
    end for
    return G / N
end procedure
```

B. Tasks and Additional Results

The following will detail the task setup and our method's optimization performance on our evaluation tasks.

B.1. Tasks

CUP A mug is rotated around its vertical axis and as its handle gets occluded, the optimization has reached a plateau. Our method differentiates through the plateau. The differentiable path tracer gets stuck in the local minimum after slightly reducing the loss by turning the handle towards the left, due to the direction of the incoming light.

SHADOWS An object outside of the view frustum is casting a shadow onto a plane. Our goal is to optimize the hidden object's position. Differentiable rasterizers can not solve this task, as they a) do not implement shadows, and b) cannot differentiate what they do not rasterize. Again, our method matches the reference position very well. Mitsuba first moves the sphere away from the plane (in negative z-direction), as this reduces the footprint of the sphere's shadow on the plane and thus leads to a lower error, and then finally moves the sphere out of the image, where a plateau is hit and the optimization can not recover. The blue line in the image-space plot in Fig. 4 illustrates this problem, as the parameter-error keeps changing very slightly, but the image-space error stays constant.

GLOBAL ILLUMINATION We here show that our method is compatible with the ambiguities encountered in advanced light transport scenarios. The goal of this optimization task is to simultaneously move the top-light to match the scene's illumination, change the left wall's color to create the color bleeding onto the box, and also to rotate the large box to an orientation where the wall's reflected light is actually visible. The optimization only sees an inset of the scene (as shown in Fig. 4) and hence only ever sees the scattered light, but never the wall's color or light itself. This task therefore is not solvable for differentiable rasterizers, as they do not model such advanced light transport. The differentiable path tracer, on the other hand, cannot resolve the ambiguity between the box's rotation, the light position and the wall's color, as it is operating in a non-smoothed space. Our method finds the correct combination of rotation, light position and wall color.

We will now detail three additional tasks that we did not include in the main text for brevity. Their qualitative results are displayed in Fig. 5.

OCCLUSION Here, a sphere translates along the viewing axis to match the reference. The challenge is that the sphere initially is occluded by another sphere, *i.e.*, we are on a plateau as long as the occluder is closer to the camera than the sphere we are optimizing. The baseline path tracer initially pushes the red sphere towards the back of the box, as this a) reduces the error in the reflection on the bottom glass plane, and b) lets the shadow of the red sphere (visible underneath the blue sphere in the initial configuration) shrink, which again leads to a lower image-space error. Our method, in contrast, successfully differentiates through both the plateau (the red sphere has a negligible effect on the objective) and the discontinuity that arises when the red sphere first moves closer to the camera than the blue occluder.

SORT: In this task, we need to sort a randomly perturbed assortment of 75 colored primitives into disjoint sets. We optimize the x- and y-coordinates of each cube, which leads to a 150-dimensional setting, with a plateau in each dimension, as most of the cubes are initially not overlapping their reference. Mitsuba cannot find the correct position of non-overlapping primitives and moves them around to minimize

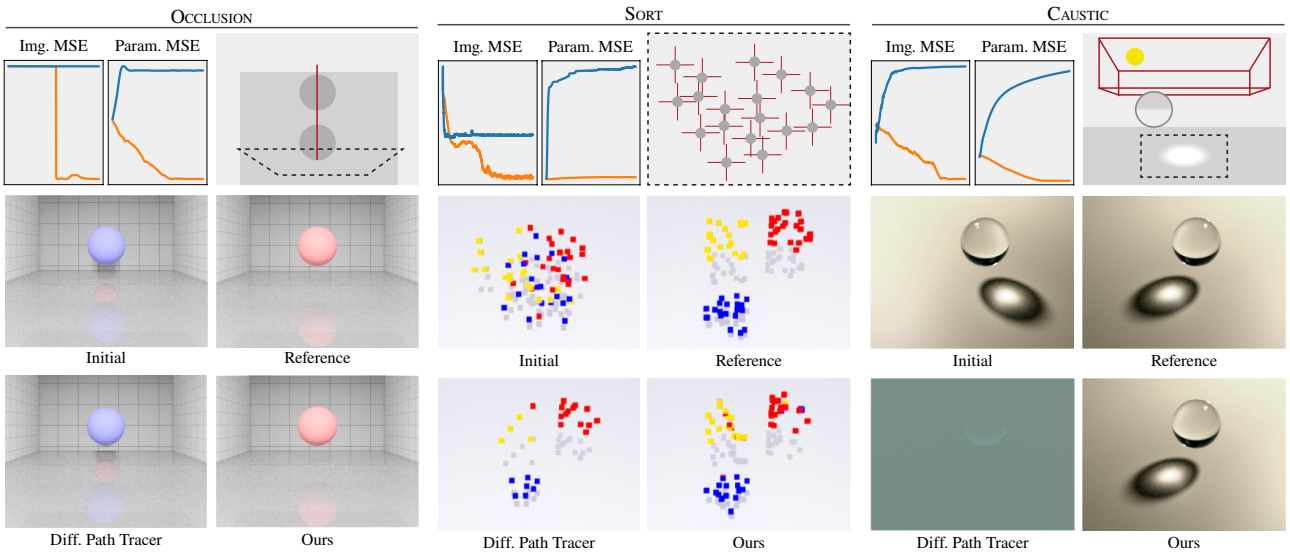

Figure 5. We show the optimization tasks and results for $\texttt{Our}_{\partial\kappa P}$ (Eq. 9, orange) and our baseline Mitsuba 3 ("Diff. Path Tracer", blue).

Table 1. Image- and parameter-space MSE of different methods (columns) on different tasks (rows). Lower is better for both metrics.

	Rasterizer		Path Tracer					
	SoftRas		Mitsuba		$\texttt{Our}_{\partial\kappa P}$		$\texttt{Our}_{\kappa\partial P}$	
	Img.	Para.	Img.	Para.	Img.	Para.	Img.	Para.
Cup	3.66×10^{-1}	2.72×10^{-2}	5.49×10^{-3}	0.75×10^{-1}	$\mathbf{4.92\times10^{-6}}$	$\mathbf{4.18\times10^{-7}}$	4.75×10^{-4}	2.77×10^{-1}
Shadows	1.64×10^{-3}	1.42×10^{-1}	1.64×10^{-3}	5.06×10^{-0}	$\mathbf{1.74\times10^{-5}}$	$\mathbf{1.81\times10^{-3}}$	5.12×10^{-4}	1.28×10^{-0}
Occl.	5.33×10^{-2}	7.18×10^{-3}	5.85×10^{-2}	$5.23\times10^{+1}$	$\mathbf{2.34\times10^{-4}}$	$\mathbf{3.29\times10^{-3}}$	5.37×10^{-2}	$1.87\times10^{+1}$
Global Ill.	–	–	3.78×10^{-2}	3.87×10^{-1}	$\mathbf{5.07\times10^{-5}}$	$\mathbf{8.71\times10^{-4}}$	5.88×10^{-2}	2.55×10^{-1}
Sort	1.85×10^{-2}	1.57×10^{-0}	1.18×10^{-2}	6.64×10^{-0}	$\mathbf{3.81\times10^{-3}}$	$\mathbf{4.19\times10^{-1}}$	1.02×10^{-2}	2.24×10^{-0}
Caustic	–	–	3.12×10^{-1}	8.50×10^{-0}	$\mathbf{1.89\times10^{-5}}$	$\mathbf{9.76\times10^{-5}}$	2.42×10^{-1}	4.03×10^{-0}

the image error, which is ultimately achieved by moving them outside of the view frustum. Our method, admittedly not perfect on this task, finds more correct positions, a result more similar to the reference.

Caustic Lastly, the **Caustic** task features a light source outside the view frustum illuminating a glass sphere, which casts a caustic onto the ground. The goal is to optimize the light's position in order to match a reference caustic. As the sphere does not change its appearance with the light's movement, the optimization has to solely rely on the caustic's position to find the correct parameters. Similar to the **GI** task, this is not solvable for rasterizers. Our method reaches the optimum position with high accuracy. For the baseline path tracer, we see a failure mode that is similar to the **Shadow** task. In this case, the image space error can be reduced by moving the light source far away, as most of the error comes from the caustic not being at the correct position. Moving the light source far away reduces this error, but also leads into a local minimum where there is no illumination at all, resulting in the gray image in Fig. 5.

B.2. Quantitative Results

Tab. 1 reports image- and parameter-space MSE and confirms what Fig. 4 conveyed visually: regular gradient-based path tracers that operate on non-smooth loss landscapes fail catastrophically on our tasks. SoftRas manages to overcome some plateaus, but struggles with achieving accurate results, as it blurs in image space but must compare to the non-blurred reference, which leads to a notable difference between the final state and the reference parameters. Our method $\texttt{Our}_{\partial\kappa P}$, in contrast, achieves errors of as low as 10^{-7}, and consistently outperforms its competitors on all tasks by several orders of magnitude. Interestingly, $\texttt{Our}_{\kappa\partial P}$ (*i.e.*, using the gradients from the differentiable renderer) works notably worse than $\texttt{Our}_{\partial\kappa P}$, which we attribute to the fact that we cannot importance-sample for the gradient here, as we do not know its PDF. Instead, we can only draw samples proportional to $\kappa(\tau)$, which places many samples where the kernel is high, *i.e.*, at the current parameter. As we can see from the rigid optimization by Mitsuba, the gradient at the current position is not very informative, so placing samples there is not very helpful.