Plateau-free Differentiable Path Tracing: Supplemental

Michael Fischer
University College London
m.fischer@cs.ucl.ac.uk

Tobias Ritschel
University College London
t.ritschel@ucl.ac.uk

This supplemental contains the hyperparameters we used for our experiments (Sec. 1), including an additional analysis of our two main parameters N and σ (Sec. 2), experiments on compatibility with plateau-free problems and other renderers (Sec. 3) and the derivations of the equations presented in the main text (Sec. 4).

1. Hyperparameters

Hyperparameters: Tab. 1 shows all the hyperparameters we use for our main experiments for all tasks. The first two columns are hyperparameters of our approach: N is the number of samples we use during an optimization iteration (for an analysis, cf. Fig. 2), and σ_0 is the kernel spread with which we start the optimization (for an analysis, cf. Fig. 3). Samples per pixel (spp) is the rendering setting we use for rendering with Mitsuba, which we generally didn't tune and hence don't regard as a hyperparameter of our method, but set such that the noise is less than the signal we want to optimize. We use the same spp across all path-tracing methods. LR is the optimizer's learning rate (we use Adam with default parameters) and the last column shows the number of optimization iterations we run. We warm-start our σ annealing schedule after approx. 50% of the optimization and use $\sigma_m = 0.01$ for all experiments as the lowest value we decrease σ to during the annealing schedule, in order to avoid numerical instabilities.

Table 1. Experiment parameters (columns) for all tasks (rows).

	N	σ_0	spp	LR	Iter.
CUP	2	0.250	16	0.01	400
SHAD.	2	0.500	32	0.02	400
OCCL.	2	0.800	32	0.02	600
GI	4	0.125	16	0.05	500
SORT	16	0.500	32	0.01	4000
CAUST.	4	0.125	32	0.01	500

Average spread: We additionally re-ran all experiments where $\sigma_0 \neq 0.5$ with the average kernel spread of $\sigma_0 = 0.5$ and show the optimization outcome in Tab. 2. Our method still performs very well and achieves results that are comparable with our findings from the main text.

Table 2. Image- and parameter-space Mean Squared Error (MSE) (rows) for $\sigma_0 = 0.5$ on different tasks (columns). Our method still performs well and finds the correct parameters.

	CUP	OCCL.	GI	CAUST.
Img.	1.8×10^{-7}	2.2×10^{-3}	1.0×10^{-4}	2.4×10^{-3}
Param.	3.0×10^{-7}	7.2×10^{-3}	6.6×10^{-2}	2.2×10^{-4}

Additional Information: Fig. 6 shows the full view of the GLOBAL ILLUMINATION task. We include this here as, in the main text, we only show the inset that the optimization sees. Note how the left wall changes color, the light changes position, and the large box changes rotation around its horizontal axis.

2. Parameter Analysis: N, σ

Timing: We further investigate the influence of the number of samples N on the convergence and runtime of our method. Recall that N is the number of *perturbations*, and not the number of samples per pixel (cf. the main text and Alg. 1 for details). Fig. 1 shows that our method's runtime scales linearly with the number of samples we use. This is as the bulk of our method's time is spent in evaluating f, *i.e.*, within the rendering operation. Using more samples means evaluating f more often, which leads to an increased runtime. The overhead of the sampling operation and the gradient computation is small in comparison and, given the linear increase in Fig. 1, can be neglected. We also show Mitsuba's runtime as the blue, dotted line. It is constant, as Mitsuba only renders a single sample, but does so with complicated methods like re-parametrization, gradient tracking or adjoint scattering. We can thus render approx. 16 samples before reaching Mitsuba's runtime (cf. also Tab. 3, main text). Therefore, our runtime does not significantly change with the number of problem dimensions (*e.g.*, 1D vs. nD), but with the time it takes to evaluate f.

Convergence: How does rendering with a higher number of samples N affect the performance of our method? To an-

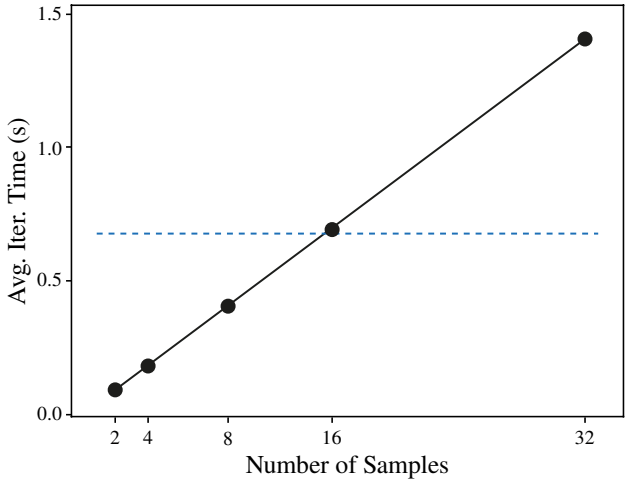

Figure 1. Runtime (vertical) of our method for different N (horizontal) on the SHADOW task. We show Mitsuba's runtime as the blue dashed line.

swer this question, Fig. 2 shows our method's convergence for different values of N. A low number of $N = 2$ (*i.e.*, a single sample and its antithesis) achieves the slowest convergence rate, while converge improves with more samples and stagnates at around $N = 10$. The final error decreases slightly with higher N (param.-MSE $9.5{\times}10^{-5}$ for $N = 2$ vs. $4.7{\times}10^{-6}$ for $N = 16$), but this improvement translates to no visible rendering improvement due to the small scale (10^{-5}) of the values. Using more than $N = 2$ hence yields no improvement here, as the faster convergence is offset by the longer runtime (cf. Fig. 1). This relation might, however, change for different tasks.

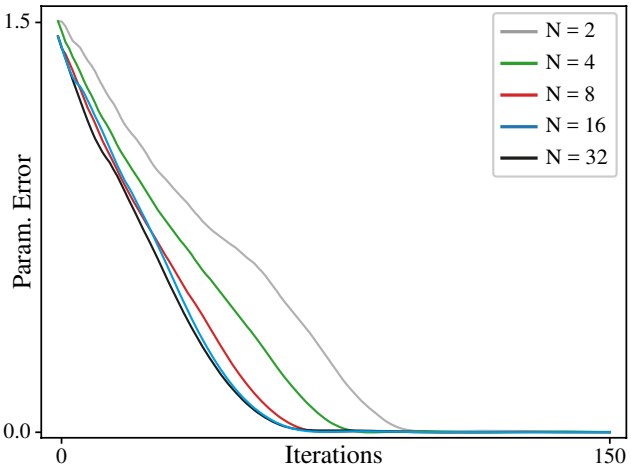

Figure 2. Convergence comparison of our method for different N (colored lines) on the SHADOW task.

Choosing σ: Moreover, we investigate how the choice of the initial kernel spread σ_0 affects the optimization outcome. With otherwise equal hyperparameters, we run the SHADOW task with σ_0 varying in $[0, 1]$ and show the results in Fig. 3.

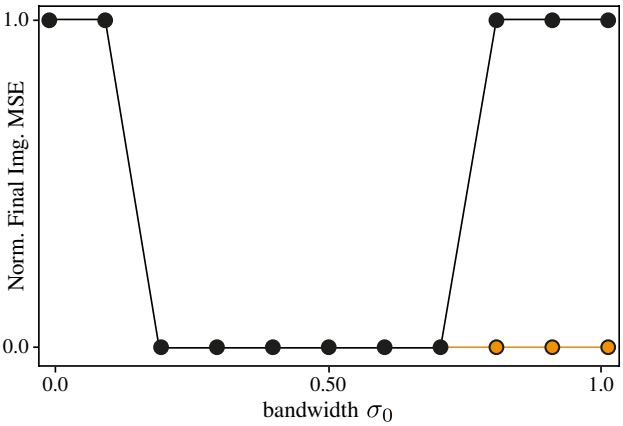

Figure 3. The effect of σ_0 (horizontal) on the optimization outcome (vertical). We show the outcome with an enlarged camera FOV in orange.

For very small σ_0, *i.e.*, $\sigma_0 < 0.2$, the optimization does not converge and produces a similar failure case to the differentiable path tracer by moving the sphere out of the image. This is as for $\sigma_0 \to 0$, our method approaches the rigid optimization by Mitsuba. The loss landscape is not smoothed and the optimization stagnates or fails. For $\sigma_0 \to 1$, we encounter a different failure case: the sampled values are so far spaced out that some of them lie outside the view frustum. As we use only $N = 2$ samples, it is thus very unlikely that we sample the proximity of the true position, leading to very noisy gradients that let our method diverge. This issue can easily be alleviated by enlarging the camera's Field of View (FOV), upon which our method converges again (orange dots in Fig. 3, camera FOV changed from $40°$ to $60°$), as the samples are then back inside the view frustum. In general, we normalize all parameter spaces to $[0, 1]$ where possible, *e.g.*, the rotation in the CUP task.

3. Compatibility

Plateau-free problems: In this section, we show that our method is compatible with optimization problems that are already plateau-free by design. To this end, we optimize an image texture that is rendered onto a plane under environment illumination. The texture has dimension $128{\times}128$ in RGB space, making this a 49,152-dimensional problem.

Fig. 4 shows the reference texture and our method's final results, alongside the convergence curves for the image (orange) and parameter (black) error.

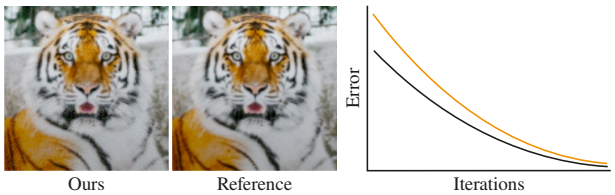

Figure 4. Texture optimization using our approach.

Path Tracer: Subsequently, we will use a different path tracing engine as backbone for our method and show that our methods also works with a different rendering backbone. For this experiment, we use `Redner`, which uses edge-sampling to derive gradient expressions during path tracing. As we can see from Fig. 5, this method fails similarly to `Mitsuba`, whereas our method again successfully delivers a complete optimization and finds the correct parameters.

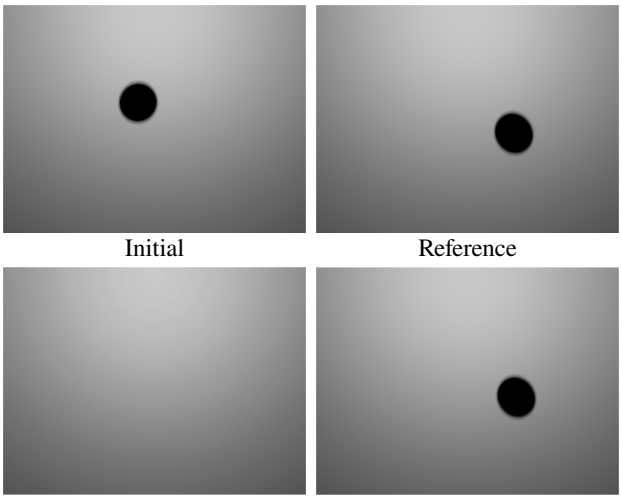

Figure 5. The SHADOW task re-run with `Redner` as renderer.

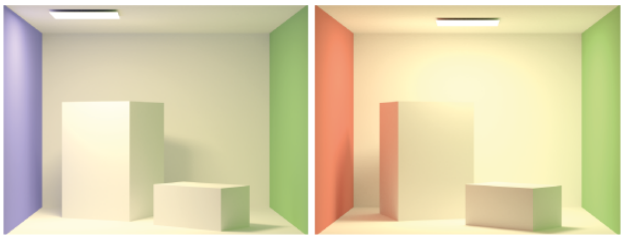

Figure 6. The full view of the GLOBAL ILLUMINATION task.

4. Additional Derivations

We derive here show how we differentiate our kernel and arrive at the equations presented in the main text.

We define our kernel as a Normal distribution in parameter space with mean 0, *i.e.*,

$$\kappa(\tau) = \mathcal{N}(0, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\tau^2}{2\sigma^2}\right)$$

which we will then use to offset our current parameters $\theta' = \theta - \kappa(\tau)$. Performing this translation with the original kernel is equivalent to convolving directly with the translated kernel (cf. Fig. 7), which naturally also holds for the derivative kernel. This allows us to rewrite the translated kernel as

$$\kappa'(\tau) = \mathcal{N}(\theta, \sigma) = -\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\tau-\theta)^2}{2\sigma^2}\right).$$

Differentiating the above equation yields

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

which evidently is a translated version of Eq. 11 in the main text. To avoid clutter in the notation, we hence write

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

In order to find the Cumulative Distribution Function (CDF) of this function, we must integrate its Probability Density Function (PDF). The PDF must integrate to 1 over the entire domain. As explained in the main text, we treat each halfspace separately and hence normalize the PDF on the positive halfspace to integrate to 0.5, yielding

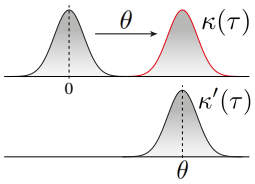

Figure 7

$$\frac{\tau}{2\sigma^2} \exp\left(\frac{-\tau^2}{2\sigma^2}\right),$$

which, upon integration, results in the CDF

$$-0.5 \exp\left(\frac{-\tau^2}{2\sigma^2}\right) + C^+,$$

where C^+ is the integration constant on the positive halfspace. Handling the negative halfspace analogously results in the same equation, but with a flipped sign and C^- as integration constant. The fact that the CDF must be continuous, monotonically increasing and defined in $(0, 1)$ tells us that $C^+ = 1$ and $C^- = 0$. To enable importance sampling with the CDF, we must invert it into the Inverse CDF (ICDF), which yields

$$F^{-1}(\xi) = \begin{cases} \sqrt{2\sigma^2 \log(2(1-\xi))} & \tau > 0 \\ \sqrt{2\sigma^2 \log(2\xi)} & \tau < 0\,. \end{cases} \quad (1)$$

Solving the domain constraints of the square-root and the logarithm, we find that the ICDF for positive halfspace is defined for $\xi \in [0.5, 1)$, whereas its negative counterpart is defined in $\xi \in (0, 0.5]$, which, given $\xi \in (0, 1)$, can be simplified to yield the final equation presented in the main text

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