Practical Gaussian Process Implicit Surfaces with Sparse Convolutions

KEHAN XU, Dartmouth College, USA
BENEDIKT BITTERLI, NVIDIA, USA
EUGENE D'EON, NVIDIA, Switzerland
WOJCIECH JAROSZ, Dartmouth College, USA

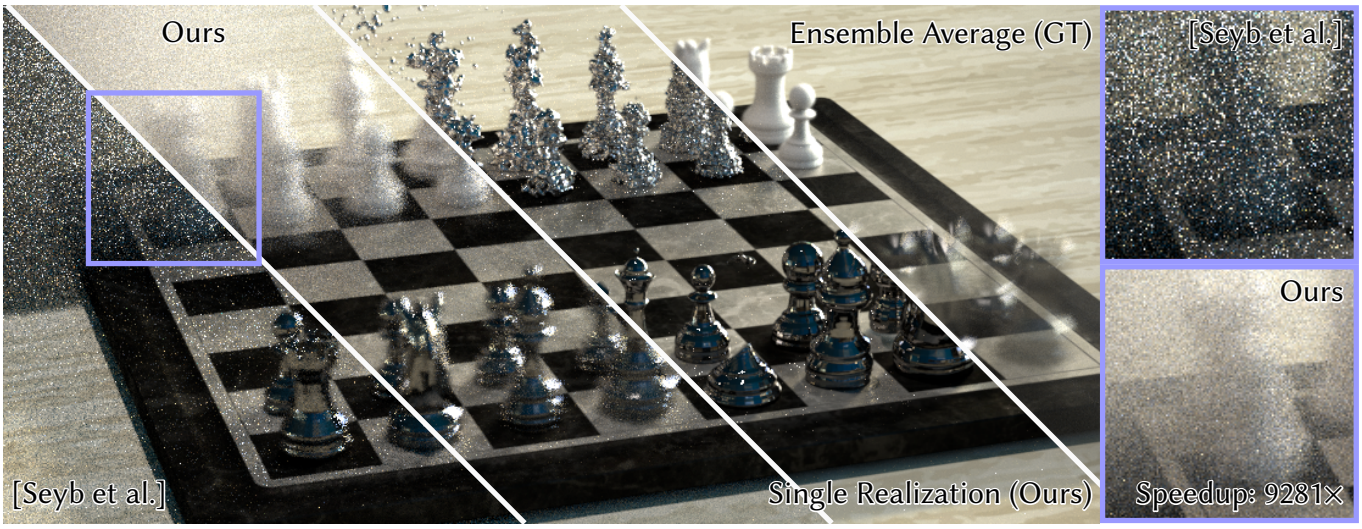


Fig. 1. Our novel sparse convolution representation for Gaussian Process Implicit Surfaces (GPISes) enables, for the first time, the visualization of single non-stationary realizations (split screen, middle right) alongside their ensemble-averaged light transport (right). The single realization allows examining the underlying structures of the stochastic geometry that lead to the aggregate rough surface or volumetric appearance. Furthermore, our algorithm (middle left) significantly boosts rendering efficiency for ensemble light transport compared to previous work by Seyb et al. [2024] (left) at equal time (40 min). The rightmost column presents magnified insets comparing the two methods, with speedup quantified by the ratio of mean squared error (MSE). This improvement stems from a combination of our efficient sparse convolution representation and the ability to perform next-event estimation even on specular micro-surfaces.

A fundamental challenge in rendering has been the dichotomy between surface and volume models. Gaussian Process Implicit Surfaces (GPISes) recently provided a unified approach for surfaces, volumes, and the spectrum in between. However, this representation remains impractical due to its high computational cost and mathematical complexity. We address these limitations by reformulating GPISes as procedural noise, eliminating expensive linear system solves while maintaining control over spatial correlations. Our method enables efficient sampling of stochastic realizations and supports flexible conditioning of values and derivatives through pathwise updates. To further enable practical rendering, we derive analytic distributions for surface normals, allowing for variance-reduced light transport via next-event estimation and multiple importance sampling. Our framework achieves efficient, high-quality rendering of stochastic surfaces and volumes with significantly simplified implementations on both CPU and GPU, while preserving the generality of the original GPIS representation.

CCS Concepts: • **Computing methodologies** → **Rendering; Ray tracing**.

Authors' addresses: Kehan Xu, Dartmouth College, USA, kehan.xu.gr@dartmouth.edu; Benedikt Bitterli, NVIDIA, USA, bbitterli@nvidia.com; Eugene d'Eon, NVIDIA, Switzerland, edeon@nvidia.com; Wojciech Jarosz, Dartmouth College, USA, wojciech.k.jarosz@dartmouth.edu.

Additional Key Words and Phrases: stochastic processes, implicit surfaces

ACM Reference Format:
Kehan Xu, Benedikt Bitterli, Eugene d'Eon, and Wojciech Jarosz. 2025. Practical Gaussian Process Implicit Surfaces with Sparse Convolutions. *ACM Trans. Graph.* 44, 6, Article 188 (December 2025), 18 pages. https://doi.org/10.1145/3763329

1 INTRODUCTION

Traditional light transport simulation categorizes scene components as either solid surfaces or participating media, with each governed by a distinct rendering equation. This binary distinction is inadequate for capturing the diverse appearances found in the real world, particularly when considering level-of-detail techniques that might require turning hard surfaces into volumetric effects with surface-like correlation.

Seyb et al. [2024] demonstrated a unified framework using ensemble-averaged light transport through stochastic geometry, covering surfaces, volumes, and the spectrum in between. A key contribution was establishing a rigorous mathematical formalism for ensemble-averaged transport. Despite its potential, this method remains impractical, with rendering speeds at least two orders of magnitude slower than conventional path tracing. This performance gap arises

because intersecting rays with the Gaussian Process Implicit Surface (GPIS) requires solving large linear systems that exhibit cubic growth in complexity with the ray length. The approach also places unintuitive constraints for authoring spatial correlations, because the linear systems become singular unless special care is taken to ensure covariance remains a positive semi-definite function. Moreover, the mathematical machinery underlying GPISes is currently less familiar to graphics practitioners, creating substantial barriers to usability and adoption.

In this paper we make significant progress on alleviating each of these concerns. We start by showing (Sec. 4) that Gaussian process implicit surfaces can be equivalently represented by certain classes of procedural noise. This allows us to view Gaussian processes from a graphics-centric perspective and leverage decades of work on efficient point evaluation of procedural noise. By linking Gaussian processes explicitly to sparse convolution noise, we eliminate the computational bottleneck of solving large linear systems, ensuring constant-time evaluations and guaranteed covariance validity. This improves performance and makes it more intuitive to author and edit this new representation.

A key advantage of the prior formulation of GPISes, however, is its inherent support for conditioning. This is crucial for efficiently computing ensemble-averaged transport where recursive rays remain consistent with observations along prior path segments. Procedural noise representations, in contrast, typically lack direct support for such conditioning. To overcome this limitation, we introduce (Sec. 5) an efficient solution based on "pathwise conditioning," which decouples the generation of procedural noise realizations from the conditioning step. We extend this approach further by supporting joint conditioning of both values and derivatives along a ray, significantly broadening the practical capabilities and appearance modeling potential of procedural noise-based GPISes.

Another significant limitation of GPISes arises when rendering stochastic geometry with highly specular materials. Without knowledge of the conditional distribution of surface normals encountered by rays intersecting the GPIS, techniques like NEE and multiple importance sampling (MIS) [Veach and Guibas 1995] – which are essential for reducing variance in scenes with high-frequency illumination – cannot be applied. We address this limitation (Sec. 6) by analytically deriving the required normal distributions, which significantly reduces variance when rendering such scenes (Fig. 1). This greatly extends the practical reach of GPISes and opens the door to bidirectional light transport in the future.

The end result is a fast and efficient framework for non-stationary, controllable stochastic surfaces. This has implications that extend beyond rendering, and opens up the potential use of GPISes in authoring pipelines and inverse rendering as a general, controllable representation that spans the gamut from surfaces to volumes.

2 RELATED WORK

2.1 Stochastic Geometry and Media

Statistical models of appearance and transport are foundational in computer graphics and vision. Canonical examples include microfacet models and participating media for efficiently rendering complex surfaces and volumes without explicitly sampling microgeometry, or volumetric/neural [Mildenhall et al. 2020] scene representations for novel view synthesis. Both Seyb et al. [2024] and Miller et al. [2024] recently showed that a stochastic geometry perspective can unify such specialized models. Seyb et al.'s GPIS framework, for instance, can express Beckmann [Beckmann and Spizzichino 1963] microfacet surfaces as well as volumes that exhibit exponential or non-exponential scattering behavior. Crucially, GPISes support longer-range memory and more general spatial correlations, allowing them to capture transport effects beyond the assumptions made by standard models. This allows lifting the independence assumptions of Smith theory for surfaces [Heitz et al. 2016], as well as going beyond renewal [Bitterli et al. 2018; d'Eon 2018; Jarabo et al. 2018] and Markovian [Pomraning 1991] approximations typically used to make transport in random media tractable.

Depending on the scale at which stochastic microgeometry is resolved in the image, it may be important to explicitly render individual realizations rather than average over them. Our approach naturally supports this use case, enabling efficient rendering of detailed non-stationary microstructure with minimal effort (Fig. 1). This capability aligns with prior work on rendering granular media [Meng et al. 2015; Moon et al. 2007; Müller et al. 2016; Zhang and Zhao 2020], and work on glints and glittery surfaces [Jakob et al. 2014; Yan et al. 2014], where high-frequency surface detail produces visually salient, realization-dependent effects. Our ability to render both individual realizations and ensemble averaged transport is an important step towards level of detail between these extremes.

2.2 Gaussian Processes

The ability of GPISes to model long-range correlations in stochastic geometry stems from the foundational properties of Gaussian processes (GPs) – a flexible and widely used framework for modeling distributions over functions. GPs have found broad application across machine learning [Rasmussen and Williams 2006], geostatistics [Murakami et al. 2020], astrophysics [Hoffman and Ribak 1991], control theory [Kocijan 2016], signal processing [Ricciardi and Sato 1986], and geometric modeling in computer graphics and robotics [Martens et al. 2017; Sellán and Jacobson 2022, 2023].

GPs are typically interpreted in two ways [Rasmussen and Williams 2006]. The *function-space* view directly defines a multivariate Gaussian over a set of evaluation points, offering great flexibility in specifying correlations but at the cost of significant complexity. The alternative *weight-space* view assumes each realization is a linear combination of basis functions, providing a more constructive and often more efficient perspective.

Seyb et al. [2024] adopted the function-space view to derive a formalism for light transport through stochastic geometry. While this enabled general covariance structures and path-dependent conditioning, it introduced complexity in both computation and implementation. Global realizations were visualized using weight-space methods with random Fourier features [Rahimi and Recht 2007], but this approach was limited to stationary processes.

In contrast, we show that the weight-space view, when framed as sparse convolution noise, naturally supports both stationary and

non-stationary Gaussian processes. This perspective yields a practical and graphics-friendly path to GPIS realization, while preserving the expressive power of the original model.

2.3 Procedural Noise

Procedural noise has long served as a cornerstone of texture synthesis and natural phenomena modeling in graphics, from fractals to terrain generation [Ebert et al. 2003; Lagae et al. 2010; Perlin 1985]. While rarely described in the same language, many procedural noise functions approximate Gaussian processes. A sum of multiple octaves of Perlin noise, for instance, approximates fractional Brownian motion. Most traditional noise functions, however, offer limited control over spatial correlations and lack a formal link to the covariance structures central to Gaussian processes.

Sparse convolution noise (SC noise) stands out as an exception. Introduced by Lewis [1986, 1989] and later extended by Lagae et al. [2009], SC noise builds a random field by convolving white noise impulses with compact kernels. We show that this naturally yields a weight-space approximation of a Gaussian process, where the covariance function is directly controlled by the choice of convolution kernel. Further extensions, including multi-scale [Lagae et al. 2011] and filtered [Lagae and Drettakis 2011] variants, enable spatially varying (non-stationary) covariance and efficient evaluation along rays – features essential for rendering GPIS realizations efficiently and intuitively.

In contrast to the function-space view used by Seyb et al. [2024], which requires careful design to maintain positive semi-definiteness, SC noise guarantees a valid Gaussian process under mild conditions on the kernel [Higdon et al. 1999; Paciorek and Schervish 2006]. This vastly simplifies the task of authoring valid spatial correlations.

While traditional procedural noise does not support conditioning, casting SC noise as a GP opens the door. Naively conditioning such a representation remains expensive due to kernel overlap, but we instead adopt pathwise conditioning – a technique originally developed in geostatistics [Journel and Huijbregts 1978] and reinvented in astrophysics [Hoffman and Ribak 1991] before being rediscovered for machine learning [Wilson et al. 2020, 2021]. Pathwise conditioning separates the sampling of the prior from the conditioning step. This aligns naturally with SC noise, since both are defined by kernel placement. We extend this approach to allow conditioning on not just the value, but also the derivatives of the GP.

3 BACKGROUND

To ground our sparse-convolution formulation of GPISes, we begin by reviewing key concepts and notation related to implicit surfaces and Gaussian processes.

3.1 Implicit Surfaces

An implicit surface is defined as the level-set of a scalar function f. Without loss of generality, we will consider zero-level sets, i.e. the points $\boldsymbol{p} \in \mathbb{R}^3$ for which $f(\boldsymbol{p}) = 0$. Where f is differentiable, the normal of the implicit surface is given by the normalized gradient $\mathbf{n}(\boldsymbol{p}) \coloneqq \nabla f(\boldsymbol{p}) / \|\nabla f(\boldsymbol{p})\|$.

For general f, computing ray intersections with the implicit surface requires numerical root finding. Although many root finding methods are available, they are challenging to apply to the surfaces we consider, and we will use ray marching for simplicity.

3.2 Gaussian Processes

A Gaussian process $f(\boldsymbol{p}) \sim \mathbf{GP}(\mu(\boldsymbol{p}), \kappa(\boldsymbol{p}, \boldsymbol{p'}))$ is a distribution over functions f characterized by the *mean* $\mu(\boldsymbol{p}) = \mathbb{E}_f[f(\boldsymbol{p})]$ and the *covariance function* $\kappa(\boldsymbol{p}, \boldsymbol{p'})$ which encodes how $f(\boldsymbol{p})$ and $f(\boldsymbol{p'})$ co-vary for any pair of locations $\boldsymbol{p}, \boldsymbol{p'}$. We give a more in-depth overview of Gaussian processes in Appendix A.

To be a valid covariance, the function κ must be positive semi-definite. That is, for any set of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, the *covariance matrix* with entries $\boldsymbol{\kappa}_{ij} = \kappa(\boldsymbol{p}_i, \boldsymbol{p}_j)$ must be positive semi-definite, i.e. $\boldsymbol{q}^\intercal \boldsymbol{\kappa} \boldsymbol{q} \geq 0$ for any $\boldsymbol{q} \in \mathbb{R}^3$.

3.3 Light Transport on Gaussian Process Implicit Surfaces

A Gaussian Process Implicit Surface (GPIS) is simply the zero level-set of a Gaussian process. Because the Gaussian process is a distribution over functions, the implicit surface is *stochastic*: We obtain a different surface for each realization f drawn from the Gaussian process. We can equivalently view a realization f as the sum $f(\boldsymbol{p}) = \mu(\boldsymbol{p}) + \psi(\boldsymbol{p})$ of a deterministic implicit function $\mu(\boldsymbol{p})$ and a stochastic function $\psi(\boldsymbol{p}) \sim \mathbf{GP}(0, \kappa(\boldsymbol{p}, \boldsymbol{p'}))$ drawn from the GP with the mean removed. To compute the surface normal, we require the covariance function κ to be twice-differentiable.

Within each realization, the light transport is given by the classical rendering equation [Immel et al. 1986; Kajiya 1986]

$$L^f(\boldsymbol{p}, \boldsymbol{\omega}) = \int_{\mathcal{S}^2} \rho^f(\boldsymbol{p}_s^f) L^f(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s) \, \mathrm{d}\boldsymbol{\omega}_s, \tag{1}$$

where $L^f(\boldsymbol{p}, \boldsymbol{\omega})$ is the radiance arriving at point $\boldsymbol{p}$ from direction $\boldsymbol{\omega}$, given in terms of the radiance leaving the closest intersection $\boldsymbol{p}_s^f \coloneqq \boldsymbol{p} + s\boldsymbol{\omega}$ of the ray with the surface induced by f. This is computed as the spherical integral of light arriving at $\boldsymbol{p}_s^f$ and being reflected into direction $\boldsymbol{\omega}$, the latter being given by the cosine-weighted BRDF $\rho^f(\boldsymbol{p}_s^f) \coloneqq \rho(\boldsymbol{p}_s^f, \boldsymbol{\omega}, \boldsymbol{\omega}_s, \mathbf{n}_s^f) |\mathbf{n}_s^f \cdot \boldsymbol{\omega}_s|$. We use a superscript of f to denote quantities that depend on the realization. To simplify the exposition, we omit the emitted radiance L_e from all equations in this paper; its inclusion is straightforward.

Ensemble-averaged transport. L^f represents transport within a specific realization f. The total radiance received is then the ensemble average over all possible realizations:

$$\left\langle L^f(\boldsymbol{p}, \boldsymbol{\omega}) \right\rangle_\zeta = \int_{\mathrm{GP}} L^f(\boldsymbol{p}, \boldsymbol{\omega}) \, \mathrm{d}\gamma(f \mid \zeta). \tag{2}$$

Here, $\langle \cdot \rangle_\zeta = \int_{\mathrm{GP}} \cdot \, \mathrm{d}\gamma(f \mid \zeta)$ represents an ensemble average restricted to realizations satisfying the set of constraints ζ (e.g. those imposed by user-controlled editing), and $\gamma(f \mid \zeta)$ denotes the classic Wiener measure – the probability density of sampling $f \sim \mathrm{GP}_{|\zeta}$.

3.4 Memory Models

Although Eq. (2) can be readily approximated with a Monte Carlo estimator that draws a random realization f for each light path and performs standard path sampling, doing so naively – by generating a full realization up-front – is computationally infeasible using the function-space sampling method from Seyb et al. [2024]. To make

this practical, Seyb et al. leverage the fact that path tracing only requires observing values along the ray's traversal. Consequently, Gaussian processes can be sampled *on the fly* by sequentially conditioning values on all prior observations along the path segment, and the result is equivalent to path tracing a full realization.

While this reduces the cost of rendering a GPIS from $O(n^9)$ to $O(m^3)$ (n being the discretization resolution and m the number of steps along the ray), the cubic cost with the number of conditioning points remains potentially unbounded for long paths. To address this, Seyb et al. introduce *memory models* to selectively discard conditioning (i.e. "memory" of past observed f values) between path segments. Mathematically, this is accomplished first by decoupling the ensemble averages of subsequent path segments:

$$\left\langle L^f(\boldsymbol{p},\boldsymbol{\omega})\right\rangle_\zeta = \left\langle \int_{S^2} \rho^f(\boldsymbol{p}_s^f)\left\langle L^{f'}(\boldsymbol{p}_s^f,\boldsymbol{\omega}_s)\right\rangle_{\zeta'} \mathrm{d}\boldsymbol{\omega}_s \right\rangle_\zeta. \tag{3}$$

Starting with only the prior constraints, radiance is obtained by averaging Eq. (1) over realizations f as before. However, the recursive term $L^{f'}$ in the above equation is computed with a *nested average* over realizations f'. This nested average uses an augmented set of conditions ζ', which restricts the GP realizations f' to those that match the observations of f along the ray segment $[\boldsymbol{p},\boldsymbol{p}_s]$.

This procedure is equivalent to the "on the fly" sampling approach described earlier; both allow for sampling values from a consistent realization by conditioning on previously observed path information. Conceptually however, this formulation is quite different: instead of selecting a realization f up front and progressively sampling it, this formulation is comparable to tracing a path through an implicit surface with uncertainty, which is progressively "locked down" as path segments observe more values from it.

Equation (3) corresponds to the *global* memory model of Seyb et al., where all path history is retained, leading to an unbounded growth in conditioning constraints with path length. Specifically, the augmented set of conditions during a single bounce is $\zeta' = \zeta \wedge \zeta_{(\boldsymbol{p},\boldsymbol{p}_s)} \wedge \zeta_\delta$. This includes $\zeta_{(\boldsymbol{p},\boldsymbol{p}_s)}$, values observed along the ray segment $(\boldsymbol{p},\boldsymbol{p}_s)$ in f, and ζ_δ, the value and gradient at $\boldsymbol{p}_s$ (i.e. $\zeta_\delta = \{f'(\boldsymbol{p}_t) = 0 \wedge \nabla f'(\boldsymbol{p}_s) = \nabla f(\boldsymbol{p}_s)\}$).

To manage computational cost, Seyb et al. propose a *Renewal+* memory model, which restricts the recursive average $\langle L^{f'}\rangle_{\zeta_\delta}$ to realizations that have a surface at the intersection point $\boldsymbol{p}_s$ with the same normal (i.e. satisfy ζ_δ), thereby excluding any prior conditions ζ and values observed along the ray segment $\zeta_{(\boldsymbol{p},\boldsymbol{p}_s)}$. Seyb et al. show that in practice this approximation leads to minor visual differences, while keeping the computational cost bounded by only carrying over a constant amount of conditioning between path segments.

4 GAUSSIAN PROCESSES VIA (SPARSE) CONVOLUTION

Despite the advances in prior work, rendering with a GPIS remains expensive: The cost of evaluating a consistent realization f increases cubically in the number of prior evaluations of f. The memory model approach of Seyb et al. "resets" this number for each path segment, but within each segment, the evaluation cost still exhibits cubic growth in the segment length.

In this section, we show how to evaluate a GPIS at *constant cost*, regardless of the number of prior observed points. We begin (Sec. 4.1) by showing that Gaussian processes can be written as white noise convolved with a kernel. We then show (Sec. 4.2) that approximating white noise with a finite number of impulses gives a consistent approximation of the Gaussian process, which ultimately admits a simple and efficient evaluation scheme (Sec. 4.2.2) in constant time by summing a finite number of kernel evaluations.

4.1 Gaussian processes from convolved white noise

Let W be a realization of independent Gaussian white noise with variance σ^2, i.e. $W(\boldsymbol{p}) \propto \mathcal{N}(0,\sigma^2)$ for any $\boldsymbol{p}$. We can write W as belonging to a Gaussian process $W \sim \mathbf{GP}(0, \sigma^2\delta(\boldsymbol{p}-\boldsymbol{p}'))$ with a Dirac delta δ covariance.

Consider now the convolution of W with a kernel h (Fig. 2 top):

$$\psi_{\text{dense}}(\boldsymbol{p}) \coloneqq \int_{\mathbb{R}^d} h(\boldsymbol{s},\boldsymbol{p})\, W(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}. \tag{4}$$

Because linear operations on a Gaussian process result in another Gaussian process, and W belongs to a zero-mean process, we know that the convolved white noise is also a Gaussian process $\psi_{\text{dense}}(\boldsymbol{p}) \sim \mathbf{GP}(0, \kappa_{\text{dense}})$ with covariance

$$\begin{aligned}\kappa_{\text{dense}}(\boldsymbol{p},\boldsymbol{p}') &= \mathbb{E}\left[\psi_{\text{dense}}(\boldsymbol{p})\,\psi_{\text{dense}}(\boldsymbol{p}')\right] \\ &= \iint h(\boldsymbol{s},\boldsymbol{p})\, h(\boldsymbol{u},\boldsymbol{p}')\, \mathbb{E}[W(\boldsymbol{s})W(\boldsymbol{u})]\, \mathrm{d}\boldsymbol{s}\, \mathrm{d}\boldsymbol{u} \\ &= \underbrace{\mathbb{E}[W^2]}_{\sigma^2} \int h(\boldsymbol{s},\boldsymbol{p})\, h(\boldsymbol{s},\boldsymbol{p}')\, \mathrm{d}\boldsymbol{s}. \end{aligned} \tag{5}$$

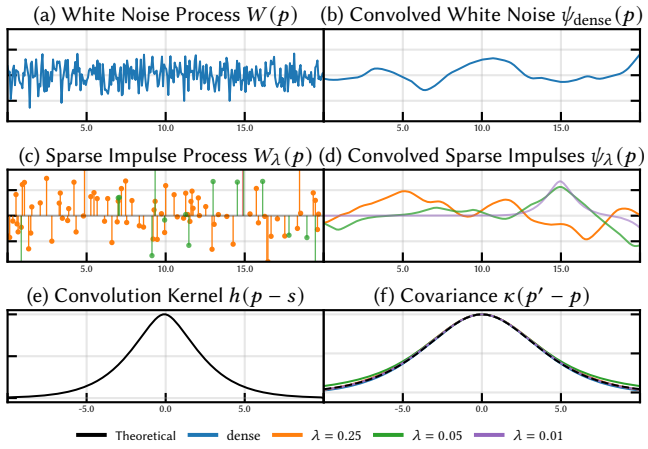


Fig. 2. We show (a) a white noise process $W(\boldsymbol{p})$ and (c) several sparse impulse processes $W_\lambda(\boldsymbol{p})$ of decreasing densities (orange, green, purple). In (b) and (d) we show the outcome of convolving these processes with the stationary kernel $h(\boldsymbol{s},\boldsymbol{p}) = h(\boldsymbol{p}-\boldsymbol{s})$ shown in (e). The convolved white noise $\psi_{\text{dense}}(\boldsymbol{p})$ is a Gaussian process, and the sparse convolution noises $\psi_\lambda(\boldsymbol{p})$ with increasing density visually approach $\psi_{\text{dense}}(\boldsymbol{p})$. (f) shows the average covariance statistics with respect to distance. Though all processes share identical covariance, this information alone does not fully characterize them. See Fig. 3 for further analysis.

This means that we can create Gaussian processes with non-trivial covariance simply by convolving white noise with a kernel. Depending on the convolution kernel h, the resulting covariance function[1] κ may be known analytically (see Table 1 for examples). More importantly however, the covariance is *guaranteed* to be positive semi-definite for nearly any h.

Proof. Inserting Eq. (5) into the definition of positive semi-definiteness (Sec. 3.2), we get [Paciorek and Schervish 2006]:

$$\begin{aligned}\boldsymbol{q}^\intercal \boldsymbol{\kappa} \boldsymbol{q} &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_i\, q_j\, \kappa(\boldsymbol{p}_i, \boldsymbol{p}_j) \qquad (6)\\ &= \sigma^2 \sum_{i=1}^{n}\sum_{j=1}^{n} q_i\, q_j \int h(\boldsymbol{s}, \boldsymbol{p}_i)\, h(\boldsymbol{s}, \boldsymbol{p}_j)\, \mathrm{d}\boldsymbol{s}\\ &= \sigma^2 \int \Big(\sum_{i=1}^{n} q_i h(\boldsymbol{s}, \boldsymbol{p}_i)\Big)\Big(\sum_{j=1}^{n} q_j h(\boldsymbol{s}, \boldsymbol{p}_j)\Big)\, \mathrm{d}\boldsymbol{s}\\ &= \sigma^2 \int \Big(\sum_{i=1}^{n} q_i\, h(\boldsymbol{s}, \boldsymbol{p}_i)\Big)^2 \mathrm{d}\boldsymbol{s} \;\geq 0. \qquad (7)\end{aligned}$$

As long as h is square-integrable, the covariance function of the resulting Gaussian process is positive semi-definite. See Appendix B for a second proof using a frequency-domain argument.

This gives us a lot of freedom in controlling the covariance of Gaussian processes, without requiring careful analysis of whether the process remains valid: We simply convolve white noise with (nearly) any kernel of our choosing.

4.2 Sparse Convolution Noise as a GP Approximation

Although evaluating Eq. (4) directly is not practical, Lewis [1986, 1989] showed how a "dense" white noise process may be approximated by a "sparse" version formed by a random impulse process:

$$W(\boldsymbol{s}) \approx W_\lambda(\boldsymbol{s}) \coloneqq \sum_i w_i\, \delta(\boldsymbol{s} - \boldsymbol{s}_i). \qquad (8)$$

The impulse locations $\{\boldsymbol{s}_i\}$ are uncorrelated and drawn from a Poisson point process with density λ, and independent random weights $\{w_i\}$ with mean $\mathbb{E}[W_\lambda] \coloneqq 0$ and variance $\mathbb{E}[W_\lambda^2] \coloneqq \sigma^2/\lambda$.

Inserting $W_\lambda(\boldsymbol{s})$ into (4) gives the sparse convolution noise $\psi_\lambda(\boldsymbol{p})$ (Fig. 2 (c) and (d)):

$$\psi_{\text{dense}}(\boldsymbol{p}) \approx \psi_\lambda(\boldsymbol{p}) \coloneqq \sum_i w_i\; h(\boldsymbol{s}_i, \boldsymbol{p}). \qquad (9)$$

Lagae et al. [2009] derive the variance of Eq. (9) and showed that the (one-point) distribution of the noise is well approximated by a normal distribution $\mathcal{N}(0, \sigma^2)$. Additionally, we can prove that the covariance of ψ_λ matches that of its dense version ψ_{dense} (see Appendix C):

$$\kappa_\lambda(\boldsymbol{p}, \boldsymbol{p}') = \underbrace{\lambda \mathbb{E}[W_\lambda^2]}_{\sigma^2} \int h(\boldsymbol{s}, \boldsymbol{p})\, h(\boldsymbol{s}, \boldsymbol{p}')\, \mathrm{d}\boldsymbol{s}. \qquad (10)$$

[1]Some GP literature refers to the covariance function κ as the covariance *kernel*. We call κ the covariance *function* to avoid confusion with the convolution *kernel* h.

Table 1. Normalized versions of convolution kernel-covariance function pairs [Matérn 1960]. We use $\mathbf{s} \coloneqq \boldsymbol{p} - \boldsymbol{p}'$ with $s \coloneqq |\mathbf{s}|$ and d the dimension ($d = 3$ for rendering GPISes). The gamma function is $\Gamma(n)$ and $\Lambda_v(s) \coloneqq v!\left(\frac{2}{s}\right)^v J_v(s)$ where $J_v(s)$ is the Bessel function of the first kind and $K_v(s)$ is the modified Bessel function of the second kind. The parameters are as follows: v, l are scalars, and the normalization factors C_h^i, C_κ^i for the convolution kernel and the covariance function respectively are: $C_h^1 = (\sqrt{2\pi}l)^{-d}$, $C_\kappa^1 = (2\sqrt{\pi}l)^{-d}$; $C_h^2 = 2^d\pi^{-\frac{1+d}{2}}\Gamma(\frac{1+d}{2})l^{-d}$, $C_\kappa^2 = \pi^{-\frac{1+d}{2}}\Gamma(\frac{1+d}{2})l^{-d}$; $C_h^3 = (2\sqrt{\pi}l)^{-d}\Gamma(\frac{1+v}{2})\Gamma(\frac{1+d+v}{2})^{-1}$, $C_\kappa^3 = (2\sqrt{\pi}l)^{-d}\Gamma(v)\Gamma(\frac{d+2v}{2})^{-1}$; $C_h^4 = 2\pi^{-\frac{d}{2}}(2l)^{-\frac{3d+2v}{4}}\Gamma(\frac{d+2v}{4})^{-1}$, $C_\kappa^4 = 2\pi^{-\frac{d}{2}}(2l)^{-d-v}\Gamma(\frac{d+2v}{2})^{-1}$.

Name	Convolution kernel h	Covariance function κ
Squared exp.	$C_h^1 \exp(-s^2/2l^2)$	$C_\kappa^1 \exp(-s^2/4l^2)$
Cauchy	$C_h^2 (1 + 4s^2/l^2)^{-\frac{d+1}{2}}$	$C_\kappa^2 (1 + s^2/l^2)^{-\frac{d+1}{2}}$
Bessel	$C_h^3 \Lambda_{\frac{d+v-1}{2}}(s/l)$	$C_\kappa^3 \Lambda_{\frac{d+2v-2}{2}}(s/l)$
Matérn	$C_h^4 s^{\frac{2v-d}{4}} K_{\frac{2v-d}{4}}(s/l)$	$C_\kappa^4 s^v K_v(s/l)$

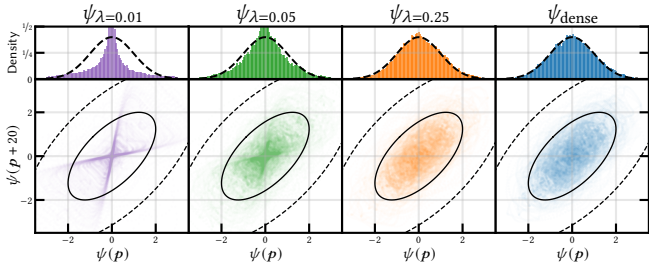


Fig. 3. 1-point amplitude distribution (top row) and 2-point joint distribution for a specific distance (bottom row) for the four processes from Fig. 2 (b) and (d). Fig. 2 (f) only shows the mean of these distributions. With low impulse density λ, the distributions of the sparse convolution noise ψ_λ are clearly different from those of the convolved white noise ψ_{dense}. As the density increases, the statistics of ψ_λ approaches ψ_{dense}, i.e. a Gaussian process.

While sparse convolution noise is *not* a Gaussian process, it does converges *in distribution* to the Gaussian Process:

$$\psi_\lambda(\boldsymbol{p}) \xrightarrow{d} \mathrm{GP}(0, \kappa_{\text{dense}}(\boldsymbol{p}, \boldsymbol{p}')) \quad \text{as} \quad \lambda \to \infty, \qquad (11)$$

as a consequence of the central limit theorem of random measures [Kallenberg and Kallenberg 1997]. This implies that ψ_λ is a consistent approximation of a Gaussian process that becomes increasingly accurate as the impulse density λ increases. Figure 3 visualizes the statistical properties of ψ_λ across increasing densities, confirming its convergence towards a Gaussian process.

4.2.1 Interpretation as Weight-Space Evaluation with Local Bases. Interestingly, sparse convolution noise can be interpreted as a weight-space approximation (30) of a Gaussian process with an infinite number of basis functions $\phi_i(\boldsymbol{p}) = h(\boldsymbol{s}_i, \boldsymbol{p})$, and weights w_i. In contrast to random Fourier features [Rahimi and Recht 2007], if the convolution kernel has compact support, only finitely many points lie within the support of h around any $\boldsymbol{p}$, and the expansion becomes sparse.

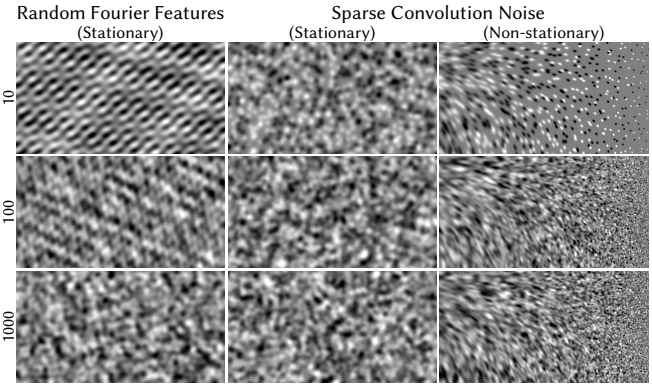


Fig. 4. In the left two columns we approximate the same 2D stationary GP via random Fourier features and sparse convolution noise, with an increasing number of basis functions (sinusoids or overlapping kernel splats) from top to bottom. Sparse convolution noise visually converges more rapidly toward the ground-truth GP and can synthesize non-stationary GPs (right column).

The first two columns of Fig. 4 compare 2D GP realizations using sparse convolution noise (with increasing impulse density) and random Fourier features (with increasing feature count), at equal computational cost. For sparse convolution noise, cost depends on the number of kernels overlapping each query point. With 100 features (middle row), sparse convolution noise closely approximates the true GP, while random Fourier features still exhibits repetitive artifacts and require more features for comparable accuracy. Moreover, sparse convolution noise can synthesize non-stationary GP realizations (right column), while random Fourier features cannot.

4.2.2 Efficient Evaluation. A major benefit of working in the sparse convolution view is that it allows dramatically more efficient evaluations of Gaussian processes. If h has compact support (e.g. a truncated Gaussian), then only $O(1)$ impulses lie near any query location $\boldsymbol{p}$, yielding $O(1)$ cost per evaluation. This can be computed efficiently on-the-fly [Lagae et al. 2009] (see Algorithm 1): we form a grid with cell width equal to the support radius of h and generate a set of random impulse positions $\boldsymbol{s}_i$ within each cell, where the number of impulses is drawn from a Poisson distribution with density λ. Due to the compact support of the kernels, the only impulses that contribute to Eq. (9) for an evaluation point $\boldsymbol{p}$ lie in the cell containing $\boldsymbol{p}$ and its immediate neighboring cells, for a total of 3^d cells (where d is the dimension). The impulses can be generated on-the-fly using a random number generator seeded by the cell index, leading to constant-time evaluation of the Gaussian process anywhere and without precomputation. Sampling distinct realizations of the process is also trivial by globally permuting the per-cell seeds.

Unlike classical weight-space approximations, sparse convolution noise also supports non-stationary processes using spatially varying convolution kernels (Fig. 4 right). Any distribution of kernels may be chosen (such as Gabor convolution noise [Charpenay et al. 2014; Lagae et al. 2009]), although extreme variability in the kernel support makes evaluation less efficient since the cell size is based on the worst-case kernel support, which in turn influences the impulse density needed for a close GP approximation. We use multi-scale sparse convolution noise [Lagae et al. 2011] – which can efficiently blend independent realizations of noise of different length scales. We could similarly benefit from many other improvements on sparse convolution noise [Lagae and Drettakis 2011; Tavernier et al. 2019].

Algorithm 1 Evaluate sparse convolution noise

function SCNoise($\boldsymbol{p}$, seed)
  $f \leftarrow 0$
  $\lambda \leftarrow$ numKernelsPerCell/cellRadius^3
  cellIndex $\leftarrow \lfloor \boldsymbol{p}/\text{cellRadius} \rfloor$
  for xyz $\in \{-1, 0, 1\}^3$ **do**
    rng $\leftarrow$ RNG(hash(seed, cellIndex + xyz))
    cellPosition $\leftarrow$ cellIndex $*$ cellRadius
    for $i = 1$ to numKernelsPerCell **do**
      $\boldsymbol{s}_i \leftarrow$ cellPosition + rng.nextFloat3()
      $l_i, \sigma_i \leftarrow$ queryKernelParameters($\boldsymbol{s}_i$)
      $w_i \leftarrow \frac{\sigma_i}{\sqrt{\lambda}}$ sampleStandardNormal(rng)
      $f \leftarrow f + w_i h(\boldsymbol{p}, \boldsymbol{s}_i, l_i)$
  return f

Algorithm 2 Render global GPIS realizations

function $L(\boldsymbol{p}, \boldsymbol{\omega}$, seed)
  for each ray marching step s **do**
    $\boldsymbol{p}_s^f \leftarrow \boldsymbol{p} + s\boldsymbol{\omega}$
    $f \leftarrow \mu(\boldsymbol{p}_s^f)$ + SCNoise($\boldsymbol{p}_s^f$, seed)
    if $f \le 0$ **then**
      $\mathbf{n}_s^f \leftarrow$ normalize(∇f)
      $\boldsymbol{\omega}_s, p_{\boldsymbol{\omega}_s} \leftarrow$ sampleScatteredDirection($\mathbf{n}_s^f, \boldsymbol{\omega}$)
      $\rho \leftarrow$ evalBSDFCosine($\boldsymbol{p}_s^f, \boldsymbol{\omega}, \boldsymbol{\omega}_s, \mathbf{n}_s^f$)
      return $L_e(\boldsymbol{p}_s^f) + \frac{\rho}{p_{\boldsymbol{\omega}_s}} L(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s$, seed)
  return 0

4.3 Rendering global GPIS realizations

We can efficiently generate and render global GPIS realizations $f(\boldsymbol{p}) = \mu(\boldsymbol{p}) + \psi(\boldsymbol{p})$ by simply modeling the stochastic portion ψ via sparse convolution noise ψ_λ (see Algorithm 2). The left and middle rows of Fig. 7 compare using random Fourier features vs. sparse convolution noise to render stationary GPISes. Similar to the 2D case (Fig. 4), sparse convolution noise is also a more efficient representation in 3D, closely approximating the ground truth GP at a low feature number (Fig. 7 middle top). Utilizing sparse convolution noise, we can render single realizations of non-stationary Gaussian processes for the first time (right bottom), allowing variation in length scale and anisotropy. We can compute the ensemble-averaged light transport (Fig. 7 right top) by simply changing the random seed in Algorithm 2 for each traced path.

5 CONDITIONING WITH PATHWISE UPDATES

Now that we can efficiently sample GP realizations, we turn our attention to conditioning. There are two forms of conditioning that we want to support: 1) *modeling-time* conditioning to shape

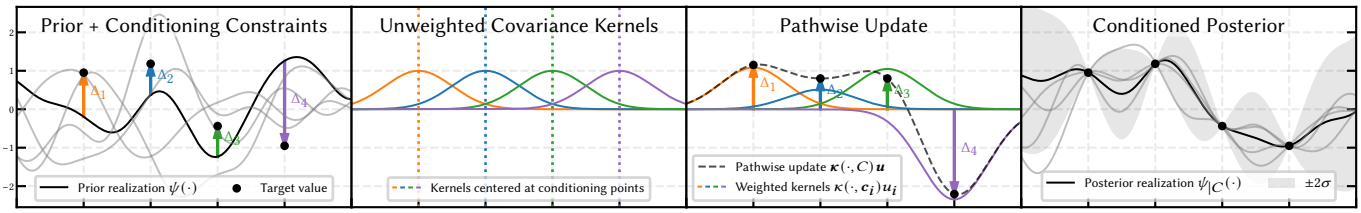


Fig. 5. Visualization of the pathwise value conditioning procedure. We sample a prior realization via sparse convolution noise (left) and combine it with (middle) a "pathwise update", constructed as a weighted sum of covariance functions centered at each conditioning point. The weights are computed by solving a linear system—with size equal to the number of conditioning points—ensuring that the sum of the prior and the update exactly interpolates the constraints (right). Each prior realization (faded curves, left) yields a distinct set of kernel weights, producing posterior realizations that strictly satisfy the value constraints (faded curves, right).

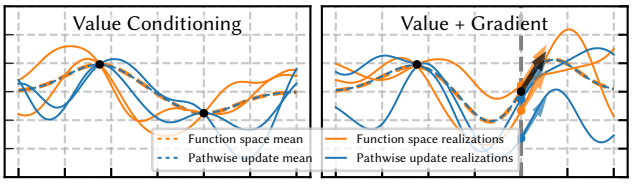


Fig. 6. Given identical prior distributions and constraints, function space conditioning (orange) and pathwise updates (blue) yield equivalent posterior distributions. We evaluate this equivalence in value conditioning (left) and a mixture of both value and gradient conditioning (right). For each case, we plot the prior and posterior means (dotted lines) alongside two sampled realizations (solid lines). Value constraints are indicated by black dots, while gradient constraints are represented by black arrows. Critically, both methods satisfy all specified constraints and produce identical posterior means in every scenario.

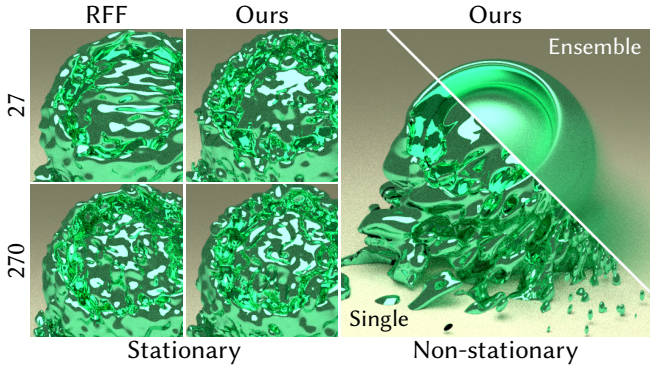


Fig. 7. Both random Fourier features (RFF) and our sparse convolution noise approach can render single realizations of *stationary* GPISes. Compared to RFF, our method more closely resembles the ground truth GP under a low number of features (left top vs. middle top). Our approach also supports rendering GPISes with general *non-stationarity* (right bottom), including spatially varying length scale and anisotropy. The ensemble light transport (right top) is readily achieved by averaging distinct realizations per sample.

the properties of the GPIS before rendering, and 2) *render-time* conditioning as a way to e.g. enable Seyb et al.'s memory models between consecutive path segment. We can afford larger upfront cost for modeling-time conditioning, but want render-time conditioning to be as lightweight as possible.

Unfortunately, traditional conditioning approaches are inefficient in our context and do not easily allow distinguishing between the performance needs of these two stages. Instead, we show how to use "pathwise updates" [Wilson et al. 2020, 2021] to efficiently condition on value, before deriving an extension of this approach for joint conditioning of value and gradient.

5.1 Value Conditioning

Given a collection $(C, \boldsymbol{v})$ of $|C|$ location-value pairs, we want to sample realizations that pass through these observations. Instead of conditioning before sampling a realization, we first sample a prior GP realization $\psi(\cdot) \sim \mathbf{GP}(\mathbf{0}, \kappa)$ (via sparse convolution noise), and then add a "pathwise" update:

$$\psi_{|C}(\cdot) = \underbrace{\psi(\cdot)}_{\text{prior}} + \underbrace{\sum_{i=1}^{|C|} \kappa(\cdot, \boldsymbol{c}_i) u_i}_{\text{update}=\boldsymbol{\kappa}(\cdot,C)\boldsymbol{u}} . \tag{12}$$

This takes a prior realization (Fig. 5, left) and adds $|C|$ weighted covariance functions $\kappa(\cdot, \boldsymbol{c}_i)u_i$ centered at the conditioning points (Fig. 5, middle). To enforce that $\psi_{|C}(\cdot)$ passes through the conditioned values, the weights u_i must satisfy the $|C| \times |C|$ linear system:

$$\boldsymbol{\kappa}(C, C)\boldsymbol{u} = \boldsymbol{v} - \psi(C), \quad \text{i.e.} \quad \boldsymbol{u} = \boldsymbol{\kappa}(C, C)^{-1}(\boldsymbol{v} - \psi(C)), \tag{13}$$

which is sparse (due to the compact support of κ). Equation (13) can be seen as solving an RBF scattered interpolation [Anjyo et al. 2014] problem (with covariance functions acting as RBF kernels) to "correct" the prior realization $\psi(\cdot)$ to match the conditioning constraints (Fig. 5, right).

Prior work [Wilson et al. 2020, 2021] has already proven that GP realizations computed via Eqs. (12) and (13) are drawn from the correct posterior distribution. Figure 6 (left) demonstrates this empirically by comparing realizations drawn from the posterior GP distribution using both function-space and pathwise conditioning.

5.2 Gradient Conditioning

We also want to support conditioning the derivatives of the GPIS (for instance to influence the local surface orientation during modeling, or to leverage the Renewal+ memory model during rendering).

To generalize the pathwise update framework to enforce derivative conditioning, we exploit the fact that the derivative of any Gaussian process $\psi \sim \mathbf{GP}(0, \kappa)$ is another Gaussian process $\nabla\psi \sim \mathbf{GP}(0, \nabla_1\nabla_2^\intercal\,\kappa)$ where ∇_1 and ∇_2 denote the gradient operators with respect to all elements of the first or second parameters of a function, and $\nabla_1\nabla_2^\intercal$ (with entries $\frac{\partial^2}{\partial p_i\,\partial q_j}$) is the mixed derivative operator with respect to all elements of both parameters.

We can perform pathwise gradient updates analogously to Eq. (12), but using a weighted sum of first derivative kernels centered at each of the conditioning points:

$$\psi_{|C'}(\cdot) = \psi(\cdot) + \sum_{i=1}^{|C'|} \nabla_2^\intercal\,\kappa(\cdot, \boldsymbol{c}_i')\,\boldsymbol{u}_i'. \tag{14}$$

To see that this produces the correct gradient-conditioned realization, we take its derivative

$$\nabla\psi_{|C'}(\cdot) = \nabla\psi(\cdot) + \sum_{i=1}^{|C'|} \nabla_1\nabla_2^\intercal\,\kappa(\cdot, \boldsymbol{c}_i')\,\boldsymbol{u}_i', \tag{15}$$

which is precisely the value pathwise update formula (12) but using the prior and covariance of the gradient process. From Eqs. (13) and (15), we see that the weights must satisfy:

$$\nabla_1\nabla_2^\intercal\,\boldsymbol{\kappa}(C', C')\boldsymbol{u}' = \boldsymbol{v}' - \nabla\psi(C'), \tag{16}$$

i.e. $\boldsymbol{u}' = [\nabla_1\nabla_2^\intercal\,\boldsymbol{\kappa}(C', C')]^{-1}(\boldsymbol{v}' - \nabla\psi(C'))$, where $\nabla_1\nabla_2^\intercal\,\boldsymbol{\kappa}(C', C')$ is a $(3|C'|) \times (3|C'|)$ matrix, and $\boldsymbol{u}'$ and $\boldsymbol{v}'$ are $(3|C'|)$-vectors (with 3 elements per conditioning point) of weights and target gradients.

5.3 Value-Gradient Conditioning

Combining Eqs. (12) and (14) allows us to condition both values and gradients

$$\psi_{|C,C'}(\cdot) = \psi(\cdot) + \sum_{i=1}^{|C|} \kappa(\cdot, \boldsymbol{c}_i)u_i + \sum_{i=1}^{|C'|} \nabla_2^\intercal\,\kappa(\cdot, \boldsymbol{c}_j')\,\boldsymbol{u}_i', \tag{17}$$

where the weights $\boldsymbol{u}$ and $\boldsymbol{u}'$ satisfy the coupled $(|C| + 3|C'|) \times (|C| + 3|C'|)$ system:

$$\begin{bmatrix} \boldsymbol{\kappa}(C, C) & \nabla_2^\intercal\,\boldsymbol{\kappa}(C, C') \\ \nabla_1\boldsymbol{\kappa}(C', C) & \nabla_1\nabla_2^\intercal\,\boldsymbol{\kappa}(C', C') \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{u}' \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} - \psi(C) \\ \boldsymbol{v}' - \psi(C') \end{bmatrix}. \tag{18}$$

Figure 6 (right) illustrates this scenario in 1D.

5.4 Unidirectional rendering of conditioned GPISes

Pathwise conditioning allows us to efficiently incorporate model-time conditioning and Seyb et al.'s memory models when rendering ensemble-averaged light transport.

Model-time conditioning. When rendering a single realization, we can efficiently precompute any modeling-time conditioning weights by solving Eq. (18) once, and then render the conditioned realization via Monte Carlo estimation of Eq. (1). Pathwise conditioning (17) essentially splats a spatially varying (realization-dependent) additive term to the GPIS being rendered, and evaluating it is linear in the number of conditioning constraints. Figure 8 illustrates the idea of using pathwise updates to condition a single 3D GPIS realization, providing control over the local height and orientation of the surface.

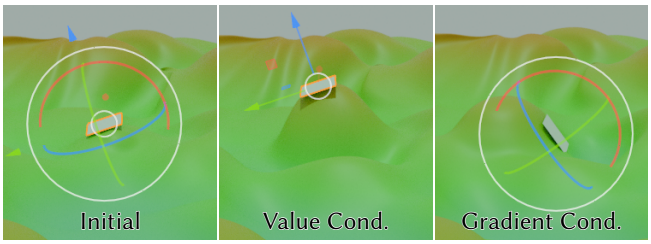


Fig. 8. Value and gradient pathwise updates allow intuitively conditioning a GPIS realization created with sparse convolution noise (left) to pass through some location (middle) or take on a certain orientation (right).

Algorithm 3 Renewal+ ensemble-average, unidirectional

function $L(\boldsymbol{p}, \boldsymbol{\omega}, \boldsymbol{v}', d, \text{seed})$
  $f \leftarrow \mu(\boldsymbol{p}) + \text{SCNoise}(\boldsymbol{p}, \text{seed})$
  $l_0, \_ \leftarrow \text{queryKernelParameters}(\boldsymbol{p})$
  if $d = 0$ **then** ▷ *depth = 0, camera ray*
    $u, \boldsymbol{u}' \leftarrow 0, \boldsymbol{0}$
  else
    $u, \boldsymbol{u}' \leftarrow \text{getWeights}(\boldsymbol{p}, \underbrace{f, 0}_{\text{Values}}, \boldsymbol{\omega}, l_0, \underbrace{\nabla f, \boldsymbol{v}'}_{\text{Gradients}})$ ▷ *Eq. (18)*
  for each ray marching step s **do**
    $\boldsymbol{p}_s^f \leftarrow \boldsymbol{p} + s\boldsymbol{\omega}$
    $f \leftarrow \underbrace{\mu(\boldsymbol{p}_s^f) + \text{SCNoise}(\boldsymbol{p}_s^f, \text{seed})}_{\text{Unconditioned GP}} + \underbrace{\kappa(\boldsymbol{p}_s^f, \boldsymbol{p}, l_0)u + \nabla_2^\intercal\,\boldsymbol{\kappa}(\boldsymbol{p}_s^f, \boldsymbol{p}, l_0)\boldsymbol{u}'}_{\text{Pathwise update}}$
    ▷ *Lines 5–8 of Algorithm 2* ◁
    return $L_e(\boldsymbol{p}_s^f) + \frac{\rho}{p_{\omega_s}} L(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s, \nabla f, d + 1, \text{seed})$
  return 0

Since the pathwise update is realization dependent, we can't precompute the weights before simulating ensemble-averaged transport. We can however precompute the LU decomposition of the system matrix in Eq. (18) before rendering, which would make evaluation quadratic. Additionally, we can solve for the weights once per pixel sample and then reuse the same conditioned realization when estimating Eq. (2) across all pixels.

Rendering with memory models. If we adopt Seyb et al.'s Renewal+ model, we obtain the particularly simple and efficient Algorithm 3. In this case, render-time conditioning requires constraining the value and gradient only at the origin of scattered rays. Hence, $|C| = |C'| = 1$ and Eq. (17) adds one weighted value-conditioning splat, and a weighted derivative-conditioning splat for each of the three dimensions. This makes Eq. (18) a small 4×4 system, which must be solved only once per path vertex during path tracing. Evaluating Eq. (17) is then constant time.

6 NEXT-EVENT ESTIMATION

Section 4 introduced an efficient method to render *global realizations* of Gaussian Process Implicit Surfaces (GPIS). While this enables faithful visualization of single realizations, it fully "locks down" the

geometry along each light path. Consequently, the normals become deterministic once a realization is fixed, eliminating essential degrees of freedom that variance reduction techniques rely on. This is particularly problematic when the GPIS surface BRDF ρ is specular. The scattered direction $\boldsymbol{\omega}_s$ is then uniquely determined by the incoming ray direction $\boldsymbol{\omega}$ and normal $\mathbf{n}_s$, collapsing the integral in Eq. (1), and rendering conventional next-event estimation (NEE) impossible. This is in stark contrast to classical microfacet models, where the micro-surface normal is not sampled until after an intersection is found. There, the rendering algorithm maintains a distribution over microfacet normals at the shading point, allowing importance sampling strategies like NEE to guide rays toward light sources effectively.

Although a specific realization is required to locate an intersection on a GPIS, the data we gather along a path forms only a *one-dimensional slice* of the stochastic geometry. In fact, when integrating Eq. (2) there exist *infinitely many* realizations consistent with these partial observations – all sharing the same intersection location and consistent values along the path segments. We can hence express the outgoing radiance L_o in direction $\boldsymbol{\omega}_o = -\boldsymbol{\omega}$ at a shading location $\boldsymbol{p}_s^f$ as the integral:

$$\left\langle L_o^f(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o)\right\rangle_\zeta = \int_{\mathcal{S}^2} \rho(\boldsymbol{p}_s^f)\left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s)\right\rangle_{\zeta'} p_n(\mathbf{n}_s \mid \zeta)\,\mathrm{d}\mathbf{n}_s, \quad (19)$$

where we use the shorthand $\rho(\boldsymbol{p}_s^f) \coloneqq \rho(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o, \boldsymbol{\omega}_s)$ for the BRDF and $p_n(\mathbf{n} \mid \zeta)$ is the *conditional distribution* of normals across these compatible realizations at $\boldsymbol{p}_s^f$, where ζ encapsulates all observations about the GPIS so far. Conceptually, this distribution of normals means that a GPIS with a fully specular BRDF may still appear non-specular since p_n will generally induce a distribution of outgoing directions.

We have a way to sample from $p_n(\mathbf{n} \mid \zeta)$ using Sec. 5, i.e. by adding conditioning splats at all prior ray marching steps. However, this is a) expensive and b) doesn't give us a way to evaluate the PDF of $\mathbf{n}$. This is the missing piece to combine our rendering method with improved sampling techniques like NEE.

6.1 From Normals to Gradients

Since $\mathbf{n}$ is the normalized gradient ∇f, the two distributions are connected by a change of variables involving the magnitude (length) of ∇f. Specifically, the normal distribution p_n is the *marginal* of the gradient distribution p_∇ over all gradient magnitudes $t = \|\nabla f\|$:

$$p_n(\mathbf{n} \mid \zeta) = \int_0^\infty p_\nabla(t\mathbf{n} \mid \zeta)\, t^2\,\mathrm{d}t, \quad (20)$$

where the Jacobian factor t^2 accounts for volume change in spherical coordinates.

With this connection, Eq. (19) can be reformulated in terms of the distribution of gradients:

$$\begin{aligned}\left\langle L_o^f(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o)\right\rangle_\zeta &= \int_{\mathcal{S}^2}\int_0^\infty \rho(\boldsymbol{p}_s^f)\left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s)\right\rangle_{\zeta'} p_\nabla(t\mathbf{n}_s \mid \zeta)\, t^2\,\mathrm{d}t\mathrm{d}\mathbf{n}_s \\ &= \int_{\mathbb{R}^3} \rho(\boldsymbol{p}_s^f)\left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s)\right\rangle_{\zeta'} p_\nabla(\nabla f \mid \zeta)\,\mathrm{d}(\nabla f) \\ &= \int_{\mathbb{R}^3} \rho(\boldsymbol{p}_s^f)\left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s)\right\rangle_{\zeta'} p(\mathbf{g} \mid \zeta)\,\mathrm{d}\mathbf{g}, \end{aligned} \quad (21)$$

where in the last step we use the shorthand $\mathbf{g} \coloneqq \nabla\psi(\boldsymbol{p}_s^f)$, with $p(\mathbf{g} \mid \zeta) \coloneqq p_\nabla(\nabla\mu + \mathbf{g} \mid \zeta)$ for the gradient of the zero-mean sparse convolution noise.

Since the gradient of the Gaussian process ψ is itself a Gaussian process, we know that the conditional distribution $p(\mathbf{g} \mid \zeta) = \mathcal{N}(\mathbf{m}, \mathbf{K})$ is *exactly* a multivariate Gaussian in $\mathbb{R}^3$ with some mean $\mathbf{m}$ and covariance $\mathbf{K}$ dependent on the conditioning ζ. If we could find this mean and covariance, we could trivially evaluate $p(\mathbf{g} \mid \zeta)$ and estimate Eq. (21) using Monte Carlo integration with an arbitrary sampling PDF. Unfortunately, we have no efficient way of doing this: Conditioning on all observations on the ray, either with function-space conditioning (28) or pathwise updates (Sec. 5), is too computationally expensive. We show how to overcome this problem in the next subsections.

6.2 The Conditional Gradient Distribution

Without loss of generality we work in a coordinate system where the local z axis is aligned with the ray direction and denote the components of the gradient in this space as $\mathbf{g} \coloneqq (g_x, g_y, g_z)$.

We first consider the distribution of $\mathbf{g}(\boldsymbol{p}_s^f)$ for a camera ray hit point $\boldsymbol{p}_s^f$. When marching from the camera, we observe $\psi(\boldsymbol{p})$ at s evenly spaced locations $\boldsymbol{p}_i = (0, 0, z_i)$, with $\boldsymbol{p}_s^f$ being the first zero crossing ($f(\boldsymbol{p}_s^f) = \mu(\boldsymbol{p}_s^f) + \psi(\boldsymbol{p}_s^f) = 0$). This gives us the length-s observation vector $\boldsymbol{\zeta} \coloneqq (\psi(\boldsymbol{p}_1), \ldots, \psi(\boldsymbol{p}_s^f))^\intercal$.

The joint distribution of $\boldsymbol{\zeta}$ and $\mathbf{g}$ is the multivariate Gaussian:

$$\begin{bmatrix}\boldsymbol{\zeta}\\ \mathbf{g}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K}_{\zeta\zeta} & \mathbf{K}_{\zeta\mathrm{g}}\\ \mathbf{K}_{\mathrm{g}\zeta} & \mathbf{K}_{\mathrm{gg}}\end{bmatrix}\right), \quad \text{with blocks} \quad (22)$$

- $[\mathbf{K}_{\zeta\zeta}]_{ij} = \mathrm{Cov}(\psi(\boldsymbol{p}_i), \psi(\boldsymbol{p}_j))$: the $s \times s$ covariance between all entries in $\boldsymbol{\zeta}$,
- $[\mathbf{K}_{\mathrm{gg}}]_{ab} = \mathrm{Cov}(\partial_a\psi(\boldsymbol{p}_s^f), \partial_b\psi(\boldsymbol{p}_s^f))$: the 3×3 covariance between gradient components of $\mathbf{g}(\boldsymbol{p}_s^f)$, and
- $[\mathbf{K}_{\mathrm{g}\zeta} = \mathbf{K}_{\zeta\mathrm{g}}^\intercal]_{ai} = \mathrm{Cov}(\partial_a\psi(\boldsymbol{p}_s^f),\ \psi(\boldsymbol{p}_i))$: the $3\times s$ cross-covariance matrix between the gradient components and observed values.

The conditional distribution of $\mathbf{g}$ given $\boldsymbol{\zeta}$ is therefore:

$$p(\mathbf{g} \mid \boldsymbol{\zeta}) = \mathcal{N}(\mathbf{m}, \mathbf{K}),\ \mathbf{m} = \mathbf{K}_{\mathrm{g}\zeta}\mathbf{K}_{\zeta\zeta}^{-1}\boldsymbol{\zeta},\ \ \mathbf{K} = \mathbf{K}_{\mathrm{gg}} - \mathbf{K}_{\mathrm{g}\zeta}\mathbf{K}_{\zeta\zeta}^{-1}\mathbf{K}_{\zeta\mathrm{g}}. \quad (23)$$

6.2.1 Stationary Isotropic Case. We now assume a stationary isotropic covariance, $\kappa(\boldsymbol{p}_i, \boldsymbol{p}_j) \coloneqq k(r = |\boldsymbol{p}_j - \boldsymbol{p}_i|)$, and discuss how to lift this assumption in Sec. 6.4. In this case, $\mathbf{K}_{\mathrm{gg}} = k''(0)\,\mathbf{I}_3$. Furthermore, with all observations $\boldsymbol{p}_i$ along the z-axis, $\mathbf{K}_{\mathrm{g}\zeta}$ also simplifies considerably: Generally $[\mathbf{K}_{\mathrm{g}\zeta}]_{zi} \neq 0$, but since the differences in the x and y coordinates of $\boldsymbol{p}_s^f$ and $\boldsymbol{p}_i$ are zero, $[\mathbf{K}_{\mathrm{g}\zeta}]_{xi} = [\mathbf{K}_{\mathrm{g}\zeta}]_{yi} = 0$ (the only special consideration is $i = s$ because then $r = 0$, but this simplification will still hold as long as $k'(0) = 0$). This means $\mathbf{K}_{\mathrm{g}\zeta}$ contains zeros in the first two rows (for g_x and g_y) and possibly non-zero elements only in the last row (for g_z). Consequently, the mean and covariance of $\mathbf{g}_{xy} \coloneqq (g_x, g_y)^\intercal$ become *independent* of $\boldsymbol{\zeta}$, allowing us to factorize the gradient distribution as:

$$\boxed{p(\mathbf{g} \mid \zeta) = p(g_z \mid \zeta)p(\mathbf{g}_{xy}) \text{ where } p(\mathbf{g}_{xy}) = \mathcal{N}\big(\mathbf{0},\ k''(0)\,\mathbf{I}_2\big).} \quad (24)$$

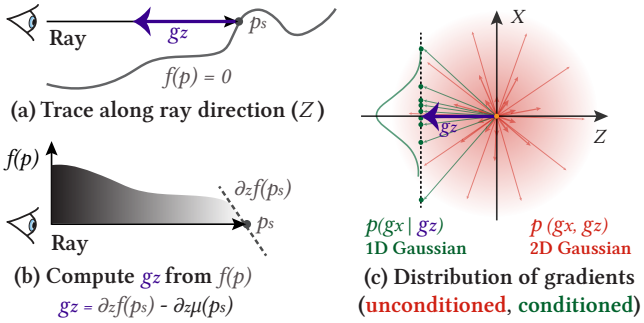


Fig. 9. We visualize the gradient distribution $\mathbf{g}$ of an isotropic stationary GP unconditioned (c, red) and conditioned on the directional derivative g_z along the ray (c, green). For clarity, we display only the 2D distribution $\mathbf{g}_{xz}$, with g_y coming out of the page. The unconditioned $\mathbf{g}_{xz}$ distribution is a 2D isotropic Gaussian. When a ray intersects a GPIS realization f at $\boldsymbol{p}_s$ (a), g_z can be computed from the value observations along the ray (b). This conditioning reduces $\mathbf{g}_{xz}$ to follow a 1D Gaussian distribution. For rays intersecting the GPIS from outside, g_z will be negative. The GPIS normal distribution $p(\mathbf{n} \mid \zeta)$ is the hemispherical projection of the gradient distribution (Eq. (20)); with g_z fixed, the GPIS normal distribution is a generalization of a Beckmann NDF, which itself can be expressed as a 2D Gaussian distribution in slope space.

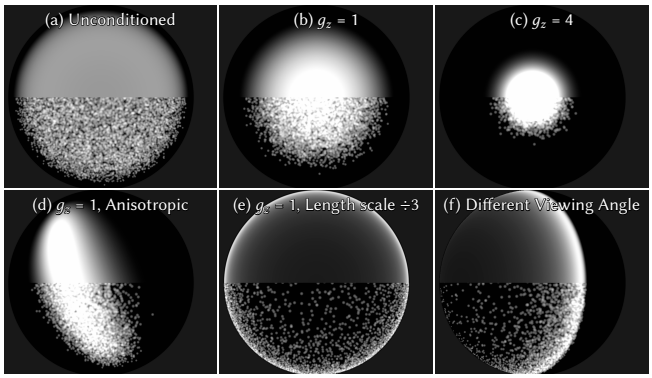


Fig. 10. We visualize the hemispherical distribution of normals from a top-down view to validate that our derived analytic distribution (Eq. (24), top half) matches the gradients sampled from sparse convolution noise realizations (bottom half). (a), (b), (c) show unconditioned and conditioned distributions, corresponding to the gradient distribution depicted in Fig. 9 (c). Larger absolute values of g_z lead to more concentrated normal distributions (c vs. b), whereas smaller covariance length scales (e vs. b) lead to broader distributions. Interestingly, both variables can be interpreted as adjusting the Beckmann surface roughness. Our interactive system also allows us to verify the correctness of anisotropic covariance kernels (d) and allows observing the distribution from arbitrary viewing directions (f). A demonstration video is provided in the supplemental materials.

Figure 9 illustrates the distribution of $\mathbf{g}$ either unconditioned or conditioned on a sampled g_z. Figure 10 validates and analyzes the probability distribution derived in Eq. (24).

This lets us rewrite Eq. (21) as a nested integral over g_z and $\mathbf{g}_{xy}$:

$$\left\langle L_o(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o) \right\rangle_\zeta = \int_{\mathbb{R}} p(g_z \mid \zeta) \left(\int_{\mathbb{R}^2} \rho(\boldsymbol{p}_s^f) \left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s) \right\rangle_{\zeta'} p(\mathbf{g}_{xy}) \, \mathrm{d}\mathbf{g}_{xy} \right) \mathrm{d}g_z. \tag{25}$$

All preceding formulas exclusively address the gradient of the zero-mean component $\mathbf{g} = \nabla\psi$. To obtain ∇f, we must simply offset the derived gradient by the mean gradient $\nabla\mu$.

6.3 Monte Carlo Estimation

We now have a path forward to perform MC estimation with NEE. While we cannot easily evaluate $p(g_z \mid \zeta)$, we can sample from it by generating a random realization f (via sparse convolution noise), and finding the intersection to obtain $\boldsymbol{p}_s^f$. The directional derivative $g_z(\boldsymbol{p}_s^f)$ is distributed exactly according to $p(g_z \mid \zeta)$. Estimating the outer integral with this sampling PDF cancels out this factor, leaving us with a 2D integral containing our now closed-form $p(\mathbf{g}_{xy})$. We can hence form the Monte Carlo estimator:

$$\widehat{L}_o(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o) = \frac{\cancel{p(g_z \mid \zeta)} \, \rho(\boldsymbol{p}_s^f, \boldsymbol{\omega}_o, \boldsymbol{\omega}_s) \left\langle L^{f'}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s) \right\rangle_{\zeta'} p(\mathbf{g}_{xy})}{\cancel{p(g_z \mid \zeta)} \, p_{\mathrm{mc}}(\mathbf{g}_{xy})}, \tag{26}$$

where we use an MC sampling PDF $p_{\mathrm{mc}}(\mathbf{g}_{xy})$ of our choosing.

Choosing to sample $\mathbf{g}_{xy}$ by looking up the directional derivatives $g_x(\boldsymbol{p}_s^f)$ and $g_y(\boldsymbol{p}_s^f)$ at the hit point $\boldsymbol{p}_s^f$ would set $p_{\mathrm{mc}}(\mathbf{g}_{xy}) = p(\mathbf{g}_{xy})$, reverting to the purely unidirectional strategy we had before. To perform next-event estimation, we can instead design $p_{\mathrm{mc}}(\mathbf{g}_{xy})$ so that the reflected direction $\boldsymbol{\omega}_s$ points towards a light source. We can also combine the two strategies using MIS as outlined in Algorithm 4. This requires (a) computing $\mathbf{g}_{xy}$ from $\boldsymbol{\omega}_s$ and g_z, as illustrated in Fig. 11, (b) converting the light sampling PDF $p^\omega(\boldsymbol{\omega}_s)$ from the solid angle measure to gradient measure $p(\mathbf{g}_{xy})$. We demonstrate the effectiveness of our MIS strategy in Fig. 12 on a reproduction of Veach's classic MIS scene with metal plates of different roughnesses modeled via stationary isotropic GPISes.

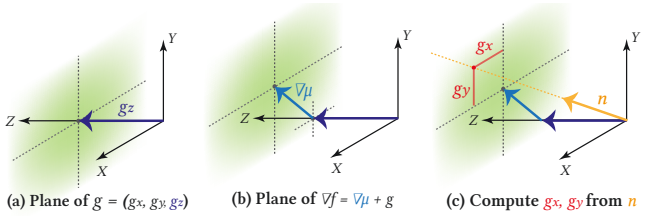
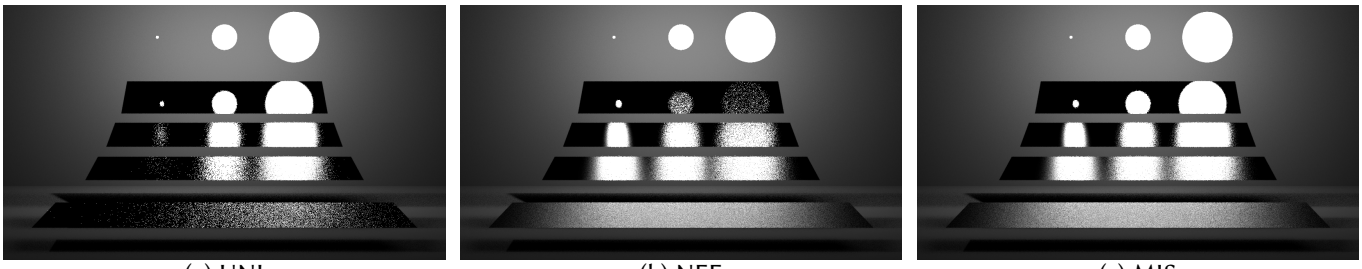


Fig. 11. Once g_z is sampled, the set of possible 3D gradients $\mathbf{g}$ spans (a) an infinite plane centered at $(0, 0, g_z)$ parallel to the X, Y axes. The shading indicates the 2D probability density $p(\mathbf{g}_{xy})$. The GPIS gradient distribution ∇f is this planar Gaussian distribution translated by the mean gradient $\nabla\mu$ (b). Emitter sampling requires computing the gradient which, when normalized, equals a given $\mathbf{n} = \frac{\boldsymbol{\omega}_o + \boldsymbol{\omega}_s}{\|\boldsymbol{\omega}_o + \boldsymbol{\omega}_s\|}$. This gradient is at the intersection of the ray along $\mathbf{n}$ with the plane of ∇f.



(a) UNI (b) NEE (c) MIS

Fig. 12. We show the Veach MIS scene composed of GPIS-represented planes with perfectly specular micro-surfaces and varying covariance function length scales. Smaller length scales produce rougher aggregated appearances. Our equal-time (1 min) rendering comparison evaluates three approaches: (a) unidirectional sampling (UNI), (b) our next-event estimation (NEE) strategy, and (c) multiple importance sampling (MIS) combining both strategies.

Algorithm 4 Renewal+ ensemble-average, delta BSDF, MIS

function $L(\boldsymbol{p}, \boldsymbol{\omega}, \boldsymbol{v}', d, \text{seed})$
  ▷ *Lines 2–10 of Algorithm 3* ◁
  if $f \leq 0$ **then**
    $\boldsymbol{p}_s^f \leftarrow \boldsymbol{p} + s\boldsymbol{\omega}$
    $L_o \leftarrow L_e(\boldsymbol{p}_s^f)$
    $\mathbf{g}^{\text{gp}} \leftarrow \nabla\text{SCNoise}(\boldsymbol{p}_s^f, \text{seed})$
    $g_z \leftarrow \mathbf{g}_z^{\text{gp}}$ ▷ *Shared by UNI and NEE*
    ▷ *Next-event estimation* ◁
    $\boldsymbol{\omega}_s^{\text{nee}}, p_\omega^{\text{nee}}, L_e^{\text{nee}} \leftarrow \text{sampleLightDir}(\boldsymbol{p}_s^f)$
    $\mathbf{g}_{xy}^{\text{nee}} \leftarrow \text{toGxy}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s^{\text{nee}}, g_z)$ ▷ *Fig. 11*
    $\mathbf{g}^{\text{nee}} \leftarrow (\mathbf{g}_{xy}^{\text{nee}}, g_z)$
    $p_{xy}^{\text{nee}} \leftarrow \text{toGxyMeasure}(p_\omega^{\text{nee}}, \boldsymbol{\omega}_s^{\text{nee}}, \mathbf{g}^{\text{nee}})$
    $p_{xy}^{\text{gp}} \leftarrow \text{evalGxyPDF}(\boldsymbol{p}_s^f, \mathbf{g}_{xy}^{\text{nee}})$
    $\mathbf{n}_s^{\text{nee}} \leftarrow \text{normalize}(\nabla\mu(\boldsymbol{p}_s^f) + \mathbf{g}^{\text{nee}})$
    $\rho^{\text{nee}} \leftarrow \text{evalBSDFCosine}(\boldsymbol{p}_s^f, \boldsymbol{\omega}, \boldsymbol{\omega}_s^{\text{nee}}, \mathbf{n}_s^{\text{nee}})$
    $L_o \leftarrow L_o + \frac{p_{xy}^{\text{nee}}}{p_{xy}^{\text{nee}} + p_{xy}^{\text{gp}}} \rho^{\text{nee}} L_e^{\text{nee}}$
    ▷ *Unidirectional sampling* ◁
    $\mathbf{n}_s^{\text{gp}} \leftarrow \text{normalize}(\nabla\mu(\boldsymbol{p}_s^f) + \mathbf{g}^{\text{gp}})$
    $\boldsymbol{\omega}_s^{\text{gp}}, \_ \leftarrow \text{sampleScatteredDirection}(\mathbf{n}_s^{\text{gp}}, \boldsymbol{\omega})$
    $p_\omega^{\text{nee}} \leftarrow \text{evalLightDirPDF}(\boldsymbol{\omega}_s^{\text{gp}})$
    $p_{xy}^{\text{nee}} \leftarrow \text{toGxyMeasure}(p_\omega^{\text{nee}}, \boldsymbol{\omega}_s^{\text{gp}}, \mathbf{g}^{\text{gp}})$
    $p_{xy}^{\text{gp}} \leftarrow \text{evalGxyPDF}(\boldsymbol{p}_s^f, \mathbf{g}_{xy}^{\text{gp}})$
    $\rho^{\text{gp}} \leftarrow \text{evalBSDFCosine}(\boldsymbol{p}_s^f, \boldsymbol{\omega}, \boldsymbol{\omega}_s^{\text{gp}}, \mathbf{n}_s^{\text{gp}})$
    $L_o \leftarrow L_o + \frac{p_{xy}^{\text{gp}}}{p_{xy}^{\text{nee}} + p_{xy}^{\text{gp}}} \rho^{\text{gp}} \text{L}(\boldsymbol{p}_s^f, \boldsymbol{\omega}_s^{\text{uni}}, \nabla f(\boldsymbol{p}_s^f), d+1, \text{seed})$
    return L_o
function $\text{toGxyMeasure}(p^\omega(\omega_s), \omega_s, \mathbf{g})$
  $\mathbf{n}_s \leftarrow \text{normalize}(\mathbf{g})$
  $p^{\mathbf{n}}(\mathbf{n}_s) \leftarrow 4(\mathbf{n}_s \cdot \omega_s) p^\omega(\omega_s)$
  return $p^{\mathbf{n}}(\mathbf{n}_s) |\mathbf{n}_s.z| / \text{length}(\mathbf{g})^2$

6.4 Generalizations and Optimizations

So far, Eq. (26) is only valid for camera rays and stationary isotropic covariance functions, but we can extend this to the Renewal and Renewal+ models, and more general forms of covariance.

Memory models. In Eq. (26), ζ' defines the conditioning that is passed to the recursive ray tracing call via a memory model. The Renewal model (remembering only the value $f(\boldsymbol{p}_s^f) = 0$ but not the gradient $\nabla f(\boldsymbol{p}_s^f)$) conditions the value $\psi(\boldsymbol{p}_1)$ at the start of the next ray. Since Eq. (24) already conditions $\mathbf{g}$ on all value observations along the ray, it is also valid for secondary rays under the Renewal model. Sampling $g_z \sim p(g_z|\zeta)$ simply requires generating a random realization with a single pathwise conditioning splat placed at $\boldsymbol{p}_1$.

The Renewal+ memory model additionally conditions on the gradient (call this $\mathbf{g}'$) at the start of the ray. Since $\mathbf{g}'$ varies in all three dimensions, it will turn our previous 3D Gaussian distribution for $\mathbf{g}$ (which was isotropic and zero-mean in $x - y$) into a more general anisotropic 3D Gaussian. Nevertheless, it is still possible to factor the distribution as $p(\mathbf{g}) = p(g_z \mid \zeta, \mathbf{g}')p(\mathbf{g}_{xy} \mid g_z, \zeta, \mathbf{g}')$ and sample a $g_z \sim p(g_z \mid \zeta, \mathbf{g}')$ for Eq. (26): we simply need to intersect the ray with a realization that additionally places a single gradient conditioning splat to constrain $\mathbf{g}'$ at $\boldsymbol{p}_1$. This leaves $p(\mathbf{g}_{xy} \mid g_z, \zeta, \mathbf{g}')$ in the numerator of Eq. (26). This will now be a more general anisotropic 2D normal distribution, but since it only depends on one additional gradient observation, we can still compute the mean and covariance of this distribution in closed form efficiently.

In practice, we found that sampling $\mathbf{g}_{xy}$ from the 2D isotropic Gaussian, i.e. $p(\mathbf{g}_{xy} \mid g_z, \zeta)$ instead of $p(\mathbf{g}_{xy} \mid g_z, \zeta, \mathbf{g}')$, can benefit more from our NEE and MIS strategy. Ignoring the correlation between $\mathbf{g}_{xy}$ and $\mathbf{g}'$ defines a new memory model, which we call *Renewal Half+*. The top two rows of Fig. 13 demonstrate that the renderings of Renewal Half+ (right) and Renewal+ (left) are visually similar. The last row shows that, while both memory models render to similar noise levels under unidirectional sampling, Renewal+ benefits significantly less from NEE and consequently MIS. This is because conditioning on $\mathbf{g}'$ shrinks the distribution of $\mathbf{g}_{xy}$, making the light sampling strategy less effective. In Sec. 7.3, we use Renewal Half+ for all ensemble-averaged light transport results.

Non-stationarity and anisotropy. If the GP is stationary but *globally* anisotropic, then its stochastic component can be written $\psi_{\text{ani}}(\boldsymbol{p}) = \psi_{\text{iso}}(\mathbf{M}\boldsymbol{p})$ for some 3×3 matrix $\mathbf{M}$. We can hence transform the ray by $\mathbf{M}$ to move into an isotropic space, and then transform $\boldsymbol{p}_s^f$ and $\mathbf{g}$ back via $\mathbf{M}^{-1}$ and $\mathbf{M}^T$ respectively when evaluating Eq. (26).

We use multi-resolution sparse convolution noise [Lagae et al. 2011] to efficiently evaluate GPs with non-stationary length scale.

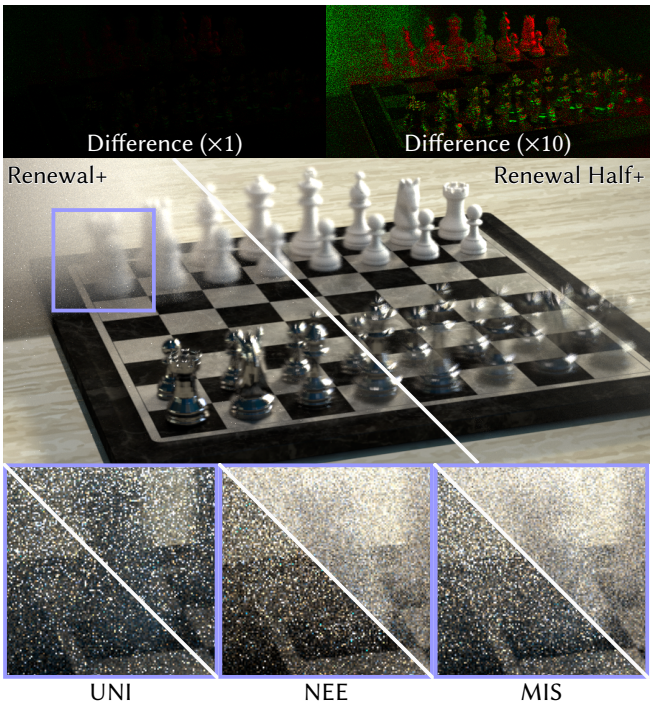


Fig. 13. The middle row shows the converged renderings for the Renewal+ (left) and Renewal Half+ (right) memory models. They are visually identical, as confirmed by the near-black difference image (top row, left). For clarity, we also show the difference magnified by 10× (top row, right). The third row compares the performance of the two memory models under UNI (left), NEE (middle) and MIS (right). While UNI produces a comparable noise level for both models, our NEE strategy (and consequently, our MIS) is significantly more efficient under Renewal Half+. This gain in efficiency is due to the forward sampling $\mathbf{g}_{xy}$ distribution being broader and better aligned with the gradient distribution derived from light sampling.

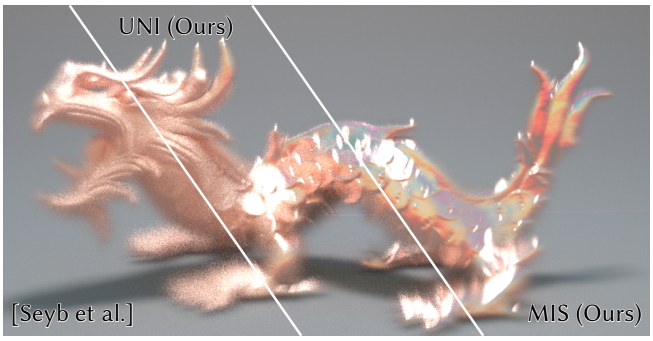


Fig. 14. We render the Dragon scene featuring a GPIS with non-stationary length scales. We compare three methods under the Renewal+ memory model: (1) function space sampling from Seyb et al. [2024], (2) our sparse convolution noise method with unidirectional path sampling, and (3) our approach incorporating multiple importance sampling (MIS). All three methods, despite their respective approximations, yield renderings that are visually similar. This demonstrates that the approximation errors introduced by both the sparse convolution noise formulation and next-event estimation are minor. All three renderings use 4096 samples per pixel.

This makes $\psi(\boldsymbol{p})$ a linear combination of two independent stationary realizations of noise of different scales, $\psi_{\text{multi}}(\boldsymbol{p}) = w_1(\boldsymbol{p})\psi_1(\boldsymbol{p}) + w_2(\boldsymbol{p})\psi_2(\boldsymbol{p})$, blended via spatially varying weight functions w_1, w_2. The gradient covariance and value-gradient cross-variance blocks of Eq. (23) can still be written exactly in terms of the isotropic stationary case by a simple application of the chain rule. The distribution $p(\mathbf{g} \mid \boldsymbol{\zeta})$ in this case becomes a combination of two 3D normal distributions.

In the case of general non-stationary anisotropy, the gradient covariance and value-gradient cross-variance blocks of Eq. (23) will not simplify exactly to Eq. (24). However, if the anisotropy changes *slowly* then Eq. (24) should remain a good approximation. More precisely, if the anisotropy remains approximately constant on the order of the length-scale from an intersection point $\boldsymbol{p}_s^f$, then (much like the globally anisotropic case above) we can transform into a *local* isotropic space at $\boldsymbol{p}_s^f$ where the assumptions of Eq. (24) will remain approximately true.

From 3D to 1D noise. We can enable a key optimization for Eq. (26) under the Renewal, Renewal Half+, or Renewal+ memory models: since sampling along a ray only requires evaluating noise in one dimension, we can replace the 3D sparse convolution noise with a 1D equivalent. This reduces the cost of noise evaluation, which scales with 3^d for dimension d. We demonstrate this and evaluate its impact on performance in the results section.

7 IMPLEMENTATION AND RESULTS

7.1 Implementation

We implemented[2] our sparse convolution GPIS approach in the Tungsten renderer [Bitterli 2018] for comparison against the implementation provided by Seyb et al. [2024]. We ran all equal-time experiments on Intel Xeon Platinum 8488C CPUs, with 48 cores and 96 threads. Furthermore, we developed[3] a simplified GPU implementation in Shadertoy. The supplementary materials include all renderings presented in the paper's figures and several videos demonstrating our GPU implementation.

We use mean squared error (MSE), and since this scales linearly with render time, we estimate speedup by taking the ratio of MSE values between methods. All assessed methods – ours and Seyb et al. [2024]'s – introduce *bias* due to their respective approximations, further detailed in Sec. 7.3. Nevertheless, Fig. 14 renders these methods at high SPP counts, empirically showing their visual similarity. For computing error metrics, we rendered ground truth images using our MIS approach, as other techniques require considerably more samples to converge on the designed scenes. The errors calculated using this reference provide a fair comparison across all methods, in particular for Seyb et al. [2024]'s approach, as the errors primarily stem from Monte Carlo sampling variance rather than fundamental image differences.

All the GPISes in our experiments use a squared exponential convolution kernel. To render GPISes with non-stationary covariance functions, we employ multi-resolution sparse convolution noise [Lagae et al. 2011] to eliminate the need for high impulse densities in

[2]Code available at https://github.com/dartmouth-vcl/sparse-conv-gpis-tungsten.
[3]Code available at https://github.com/dartmouth-vcl/sparse-conv-gpis-shadertoy.

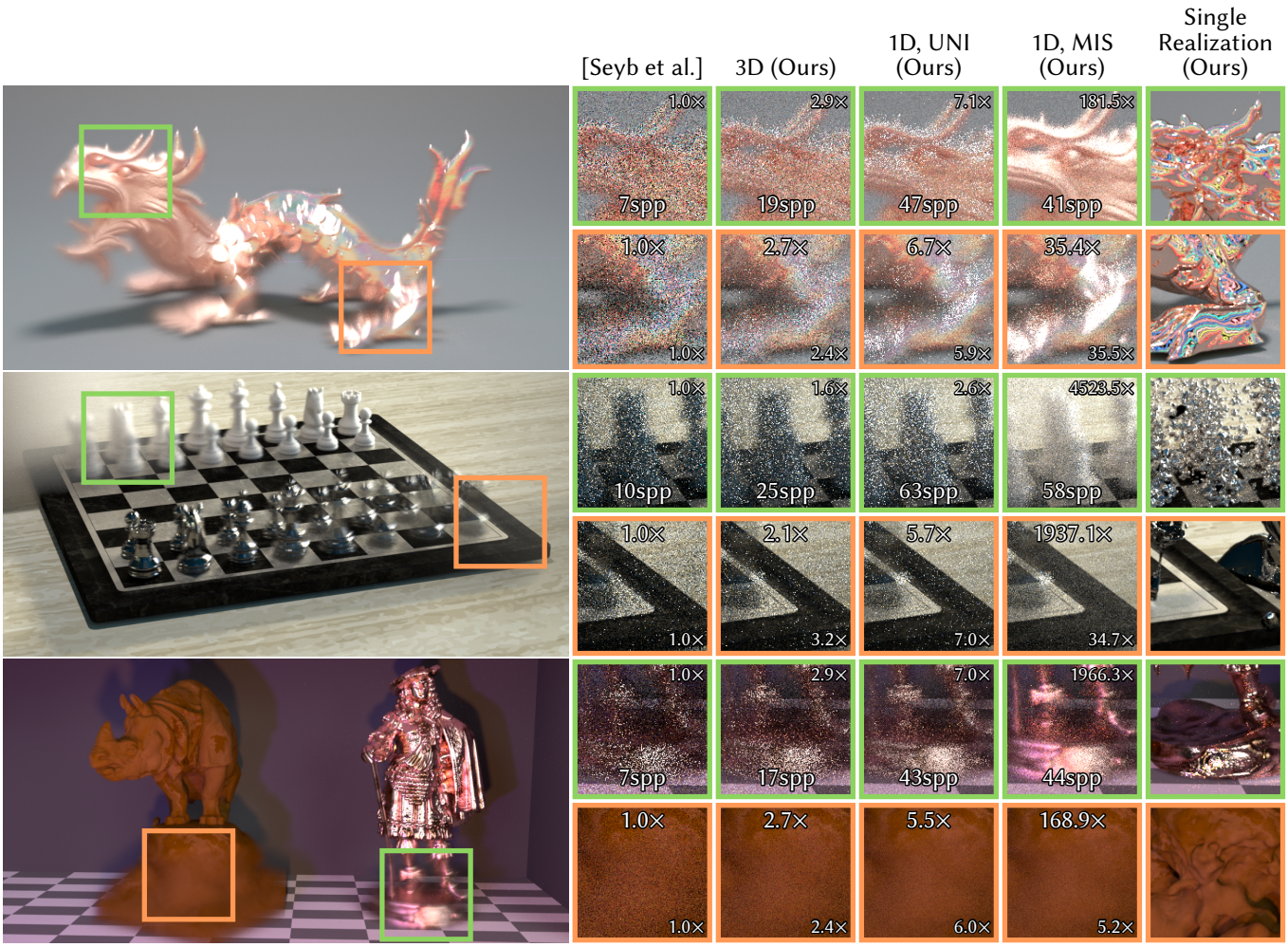


Fig. 15. We evaluate the function space method from Seyb et al. [2024] and our sparse convolution approach at equal time (20 min), on multiple scenes composed of non-stationary GPISes with perfectly specular micro-surface materials. We present several variants of our method (middle three columns of insets), including the 3D noise implementation, 1D noise with unidirectional sampling, and 1D noise with MIS. The techniques are added progressively to demonstrate the improvement brought by each. Additionally, in the rightmost column, we visualize a single realization of the GPIS, offering insight into the underlying structure that composes the aggregate appearance. The SPP and the speedup over the whole image are shown in the bottom and top inset centers. The speedup for each image patch is shown in the corner.

regions with large variations in kernel scale. Following Tavernier et al. [2019], we splat a fixed number of kernels per cell (10) rather than sampling this number from a Poisson distribution. We provide an analysis on this number in Sec. 7.3. For all ensemble-averaged light transport results, we use the Renewal Half+ memory model.

7.2 Scene Representation and Memory Usage

Evaluating a GPIS at a point requires being able to (1) query the mean and covariance fields, and (2) synthesize the zero-mean stochastic component using sparse convolution noise. The stochastic component is generated procedurally on the fly and does not require stored samples.

Persistent memory usage is therefore determined by how the mean and covariance fields are represented. This could be procedural (no storage) or tabulated (as e.g. a voxel grid or octree). For instance, we could express a mean function derived from a mesh procedurally by directly computing the nearest distance of the query point to the mesh triangles – in $O(\log N)$ time for N triangles via a BVH. Alternatively, we can achieve $O(1)$ lookup complexity at the expense of increased memory usage by precomputing the nearest distance over a voxel grid. We use the latter approach in our results and store the fields in OpenVDB grids. Covariance parameters can likewise be procedural or stored in an OpenVDB grid. These storage requirements are identical to that of Seyb et al. [2024].

7.3 Results

Sparse Convolution Noise vs. Random Fourier Features. We compared the two weight-space approaches for 2D noise generation in Fig. 4 and for rendering 3D stationary GPISes in Fig. 7 (left and middle). Both techniques can efficiently sample global realizations, thoughrandom Fourier features (RFF) are fundamentally limited to stationary covariance functions. Furthermore, both figures illustrate that sparse convolution noise more closely approximates the ground truth Gaussian process than RFF under equal computational cost.

Sparse Convolution Noise vs. Function Space Sampling. Seyb et al. [2024] primarily proposed rendering GPISes with the function-space approach. We compare our sparse convolution approach with theirs on rendering ensemble light transport under the Renewal+ memory model, as their method is impractical for sampling single realizations (whereas our method can, shown in Fig. 15 rightmost column).

As illustrated in Fig. 14 (left and middle), both function space sampling and our approach yield visually comparable renderings. While both methods introduce approximations for practicality (Seyb et al. evaluates the GPIS in batches of ray march locations and propagates correlations only to neighboring batches, and our method utilizes a multi-resolution grid), we did not find these approximations to result in significant visual differences in practice.

In Fig. 15, we present an equal-time comparison of our method's variants against Seyb et al.'s approach. Each subsequent variant of our method progressively incorporates additional techniques, illustrating their individual contributions to performance improvement. For all scenes, we evaluated the function space method with a batch size of 8 ray marching points.

Notably, by simply replacing function-space sampling (Fig. 15, first column of insets) with 3D sparse convolution noise sampling (second column of insets), our method outperforms Seyb et al.'s by an average speedup of 2.4×.

From 3D to 1D Noise. Under Renewal, Renewal Half+, or Renewal+ memory models, when sampling a hit point $\boldsymbol{p}_s^f$ and $g_z \sim p(g_z \mid \zeta)$ via ray marching we only ever need to look up values (and the directional derivative) along the ray. This means we can get away with synthesizing 1D instead of 3D noise, allowing a 9× reduction in noise evaluation cost. The two remaining gradient components, $\mathbf{g}_{xy}$, can then be explicitly sampled from the normal distribution $p(\mathbf{g}_{xy})$ in Eq. (24).

The third column of insets in Fig. 15 illustrates the performance improvement of transitioning from a 3D to a 1D sparse convolution noise. After accounting for other overheads (e.g., VDB queries, emitter sampling), this yields an average render speedup of 2.3×.

Next-event Estimation. The technique we derived in Sec. 6 allows us to apply NEE even in the case where the micro-surface uses a perfectly specular BRDF. Figure 12 compares the effectiveness of unidirectional sampling, our NEE, and their MIS combination in the classic Veach scene where each plane is represented as a GPIS.

Our method with and without MIS will converge to the same result even for non-stationary GPISes (as long as only the scale, and not the anisotropy, of the covariance varies spatially). We confirm this visually in the non-stationary Dragon scene (Fig. 14 middle and right).

(a) Single Realization (b) Ensemble Average (c) Microfacet Surface

Fig. 16. Our sparse convolution GPIS framework unlocks the ability to visualize and interactively explore (a) individual realizations of micro-geometry, which, when ensemble averaged (b), reproduce Beckmann microfacet appearance (c). This would be infeasible with the expensive linear system solves of prior function-space methods, but is now possible to do interactively even in a simple GPU-based Shadertoy implementation.

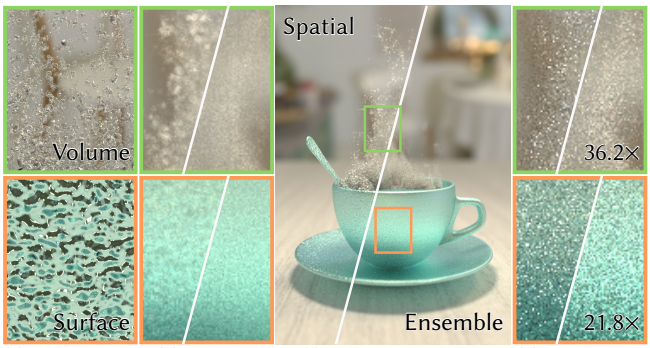
Figure 15 contains two other more complex scenes, all featuring GPISes with spatially varying length scales, specular micro-surface GPISes and finite-area light sources. The ability to perform NEE significantly enhances rendering efficiency (fourth column of insets), with speedups varying considerably across different scene settings.

GPU-Accelerated Prototype. The implementation of GPIS is significantly simplified through sparse convolution noise, enabling broad compatibility across different platforms. As a proof of concept, we demonstrate real-time GPU rendering of GPISes with simplified light transport in Shadertoy. This is possible because we avoid the expensive linear system solves needed by function space sampling. Figure 16 presents a screenshot of our GPU-accelerated prototype. A demonstration video is available in the supplementary materials.

Spatial vs. Ensemble Average. Here, we explore the connection between spatial and ensemble averages of GPISes. When the covariance scale is on a small scale, a spatial average over realizations within a single pixel closely resembles an ensemble average over realizations across samples. This resemblance is the fundamental rationale behind Seyb et al.'s initial exploration of ensemble light transport; however, they lacked a direct validation, partly because their function-space approach did not support rendering individual GPIS realizations.

In Fig. 17, we validate this theory using a coffee cup and smoke, both represented by a single GPIS with a non-stationary covariance function. The spatial and ensemble-averaged results in (b) and (c) demonstrate strong visual similarity. Furthermore, zooming into the spatial averages allows us to visualize in (a) the micro-structures simulated by GPIS, which give rise to the macro-scale appearances of both the surface and the volume. The volume consists of small particles, while the surface exhibits bumpy structures, both consistent with classic appearance models.

While spatial averaging yields an appearance similar to ensemble averaging and uniquely enables the visualization of micro-structures,



(a) Micro-struct. (b) Zoom-in (c) Full-size Rendering (d) UNI MIS

Fig. 17. We validate the connection between spatial and ensemble averages using a coffee cup and smoke, modeled with a single non-stationary GPIS. With sub-pixel covariance scales, the spatial average yields similar macro-scale appearances as the ensemble average in (b) and (c). Our sparse convolution noise approach allows us to visualize the intricate surface and volumetric micro-structures (a) by zooming into the spatial average. Conversely, the flexibility of ensemble light transport allows us to employ next-event estimation for improved importance sampling. (d) illustrates this improvement both qualitatively and quantitatively, comparing unidirectional path tracing and MIS at equal time (20 min).

its nature as a single realization prevents applying importance sampling techniques when the micro-surface is a delta BSDF. On the other hand, the geometric randomness in the ensemble average provides the flexibility to perform next-event estimation. Figure 17 (d) demonstrates the performance gain of using MIS compared to unidirectional path tracing at equal time.

Number of Kernels per Cell. The number of kernels per cell in sparse convolution noise critically influences its accuracy in approximating a Gaussian process. Insufficient density could result in bias in the effective transmittance, NDF and final rendering. Luckily, we found that stationary GPISes, even anisotropic ones, can achieve accurate results with very few kernels. As shown in Fig. 7 (middle top), even a single kernel per cell (27 overlapping kernels or "features") produces realizations resembling higher-density counterparts. Similarly, non-stationary isotropic scaling, when implemented with multi-resolution noise, does not increase the required kernel density.

Because the procedural grid cell size is based on a conservative maximum kernel radius, kernels fill space more sparsely when anisotropy can vary rapidly across space. In these situations, we found that higher impulse densities are required for visually unbiased results. Figure 18 shows a non-stationary anisotropic GPIS with varying kernel densities, demonstrating how the bias diminishes as the number of kernels per cell increases.

8 LIMITATION AND FUTURE WORK

Convolution Kernel-Covariance Function Pairs. No universal method exists to derive analytic convolution kernels for arbitrary positive semi-definite covariance functions, nor vice versa. For instance, we were unable to derive a corresponding convolution kernel for the

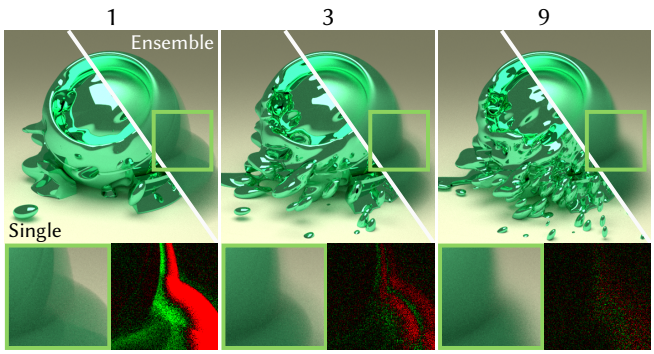


Fig. 18. As the number of kernels per cell increases from left to right, the sparse convolution noise more closely approximates a Gaussian process (split screen, left), and the ensemble light transport converges toward the ground truth (split screen, right). Insets highlight the artifacts at low kernel densities and visualize the diminishing bias (scale by 10× for better visibility).

rational quadratic covariance function. Nevertheless, we emphasize that sparse convolution GPISes only require knowledge of the convolution kernel itself. This affords significant flexibility, as we may select any once-differentiable kernel that provides the desired spatial correlation properties.

Efficient Directional Non-stationarity. While multi-scale sparse convolution noise effectively handles length scale variations, representing extreme directional non-stationarity remains challenging. Our current method achieves close approximations only by inefficiently increasing the impulse density, necessitating global oversampling for localized variations. Future work could focus on adaptive strategies for efficiently handling complex, spatially varying anisotropy without this uniform oversampling.

Ray Marching Acceleration. Despite our advancements, GPIS rendering speed remains constrained by ray intersection cost, which scales with ray length due to our use of brute-force ray marching. Recent work on sublinear ray marching or implicit surface acceleration structures [Moinet and Neyret 2025; Tokuyoshi and Harada 2019] may provide substantial further improvement.

Improving Next-event Estimation. Our current next-event estimation strategy is formulated for 3D GPISes. For a 2D GPIS (e.g. a heightfield), the gradient distribution at an intersection, conditioned on ray observations, has only one degree of freedom instead of two. Consequently, possible outgoing directions form a 1D distribution, so randomly sampling a 2D area light source has zero chance of providing a contribution. Specialized sampling techniques are therefore required for this scenario.

Our NEE derivation for perfectly reflective materials easily extends to refractive ones. While our current strategy yields no improvement for non-delta BSDFs (the last row of Fig. 15), the analytic gradient distribution presents a further opportunity to enhance their importance sampling techniques. For instance, if the micro-BSDF (i.e. the BSDF on the GPIS surface) follows a Gaussian distribution, it can be convolved with the gradient distribution (another Gaussian) to produce an analytic macro-BSDF. Otherwise, the terms in the

integrand can be individually importance sampled, with different strategies combined via MIS.

GPIS Editing Tools. As a novel geometry representation, GPISes offer intuitive editing capabilities, as we illustrated conceptually in Fig. 8. Developing artistic tools for GPIS, encompassing both modeling and texturing, is a desirable future direction. This process presents not only engineering challenges but also intriguing research problems, such as establishing mappings between GPIS parameters and traditional appearance models, exploring mechanisms for uncertainty manipulation, and adapting established 3D noise texturing techniques to GPIS.

Reciprocity. To enable bidirectional rendering algorithms, such as bidirectional path tracing or photon mapping, we must rigorously prove the reciprocity of the ensemble light transport integral. Assuming deterministic camera and light sources (where only intermediate path vertices reside on GPISes), we foresee no fundamental obstacles to the ensemble light transport being reciprocal. However, a more challenging scenario would involve stochastic light sources, i.e. emissive GPISes.

9 CONCLUSION

We present significant advances in the practical application of Gaussian Process Implicit Surfaces. Our core contribution is a novel sparse convolution noise formulation, which offers a more intuitive representation than prior function-space methods. This new representation enables practical sampling of global non-stationary GPIS realizations and improves rendering efficiency for ensemble light transport. We further introduce pathwise update for flexible value and gradients conditioning during both modeling and rendering. Crucially, we enable next-event estimation for specular micro-surfaces by deriving the analytic normal distributions as Gaussian distributions. This further boosts rendering efficiency and bridges the gap between GPIS rendering and importance sampling of traditional appearance models. The resulting noise-based approach is easy to implement on either a CPU or GPU platform.

ACKNOWLEDGMENTS

We thank Dario Seyb for insightful discussions on GPIS and Jeffrey Liu for providing helpful feedback. This work was partially supported by NSF awards 1844538 and 2440472. We used assets licensed under CC0 in many figures of this paper and thank the original creators for providing them: shader ball [Takikawa et al. 2022], chess set (polyhaven.com, by Riley Queen), dragon (OpenVDB sample models), two statues (threedscans.com), and coffee mug (Sketchfab). We pair-programmed with an LLM to refine the code and layout of several programmatically generated figures: Figs. 2, 3, 5, 6 and 19.

REFERENCES

Ken Anjyo, J. P. Lewis, and Frédéric Pighin. 2014. Scattered Data Interpolation for Computer Graphics. In *ACM SIGGRAPH Courses*. ACM Press, New York, NY, USA, 1–69. https://doi.org/10/gt2jcm

Petr Beckmann and André Spizzichino. 1963. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon Press, NY.

Benedikt Bitterli. 2018. Tungsten Renderer. https://github.com/tunabrain/tungsten/

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. A Radiative Transfer Framework for Non-Exponential Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Nov. 2018), 225:1–225:17. https://doi.org/10/gfz2cm

Victor Charpenay, Bernhard Steiner, and Przemyslaw Musialski. 2014. Sampling Gabor Noise in the Spatial Domain. In *Proceedings of the 30th Spring Conference on Computer Graphics (SCCG '14)*. Association for Computing Machinery, New York, NY, USA, 79–82. https://doi.org/10/f3s4cw

Eugene d'Eon. 2018. A Reciprocal Formulation of Non-Exponential Radiative Transfer. 1: Sketch and Motivation. *arXiv:1803.03259 [nucl-th, physics:physics]* (March 2018). arXiv:1803.03259 [nucl-th, physics:physics]

David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Kenneth Perlin, and Steven Worley. 2003. *Texturing and modeling: a procedural approach* (3rd ed.). Morgan Kaufmann, San Francisco, CA, USA.

Dennis Gabor. 1946. Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering* 93, 26 (1946), 429–441.

Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. 2016. Multiple-Scattering Microfacet BSDFs with the Smith Model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 35, 4 (July 2016). https://doi.org/10/f89kkm

David Higdon, Jenise Swall, and John Kern. 1999. Non-Stationary Spatial Modeling. In *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, J M Bernardo, J O Berger, A P Dawid, and A F M Smith (Eds.). Oxford University Press. https://doi.org/10.1093/oso/9780198504856.003.0036

Yehuda Hoffman and Erez Ribak. 1991. Constrained Realizations of Gaussian Fields: A Simple Algorithm. *The Astrophysical Journal* 380 (Oct. 1991), L5. https://doi.org/10/dngzjw

David S. Immel, Michael F. Cohen, and Donald P. Greenberg. 1986. A Radiosity Method for Non-Diffuse Environments. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (Aug. 1986), 133–142. https://doi.org/10/dmjm9t

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete Stochastic Microfacet Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33, 4 (July 2014), 115:1–115:10. https://doi.org/10/f6cnzx

Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A Radiative Transfer Framework for Spatially-Correlated Materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 37, 4 (July 2018), 83:1–83:13. https://doi.org/10/gd52qq

A. G. Journel and Ch J. Huijbregts. 1978. *Mining Geostatistics*. Academic Press.

James T. Kajiya. 1986. The Rendering Equation. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (Aug. 1986), 143–150. https://doi.org/10/cvf53j

Olav Kallenberg and Olav Kallenberg. 1997. *Foundations of modern probability*. Vol. 2. Springer.

Juš Kocijan. 2016. *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-21021-6

Ares Lagae and George Drettakis. 2011. Filtering Solid Gabor Noise. *ACM Trans. Graph.* 30, 4 (July 2011), 51:1–51:6. https://doi.org/10/bdm39b

Ares Lagae, S. Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S. Ebert, J. P. Lewis, Ken Perlin, and Matthias Zwicker. 2010. State of the Art in Procedural Noise Functions. In *Eurographics 2010 - State of the Art Reports*. The Eurographics Association. https://doi.org/10/gtxh9f

Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural Noise Using Sparse Gabor Convolution. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 28, 3 (July 2009), 1–10. https://doi.org/10/cvv5s7

Ares Lagae, Sylvain Lefebvre, and Philip Dutre. 2011. Improving Gabor Noise. *IEEE Transactions on Visualization and Computer Graphics* 17, 8 (Aug. 2011), 1096–1107. https://doi.org/10/cr3qwx

J. P. Lewis. 1986. Methods for Stochastic Spectral Synthesis. In *Proceedings of Graphics Interface and Vision Interface '86 (Gi '86)*. Canadian Man-Computer Communications Society, Toronto, Ontario, Canada, 173–179. http://graphicsinterface.org/wp-content/uploads/gi1986-30.pdf

J. P. Lewis. 1989. Algorithms for Solid Noise Synthesis. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '89)*. Association for Computing Machinery, New York, NY, USA, 263–270. https://doi.org/10/cw7wzj

Wolfram Martens, Yannick Poffet, Pablo Ramon Soria, Robert Fitch, and Salah Sukkarieh. 2017. Geometric Priors for Gaussian Process Implicit Surfaces. *IEEE Robotics and Automation Letters* 2, 2 (April 2017), 373–380. https://doi.org/10/ghk78w

Bertil Matérn. 1960. *Spatial variation*. PhD thesis. Stockholm University, Example City, CA.

Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-Scale Modeling and Rendering of Granular Materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 34, 4 (July 2015), 49:1–49:13. https://doi.org/10/gfzndr

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*. https://doi.org/10/gn9vj9

Bailey Miller, Hanyu Chen, Alice Lai, and Ioannis Gkioulekas. 2024. Objects as Volumes: A Stochastic Geometry View of Opaque Solids. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*. 87–97. https://doi.org/10/gt258p
Mathéo Moinet and Fabrice Neyret. 2025. Fast sphere tracing of procedural volumetric noise for very large and detailed scenes. In *Computer Graphics Forum*. Wiley Online Library, e70072.
Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. 2007. Rendering Discrete Random Media Using Precomputed Scattering Solutions. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*. Eurographics Association, 231–242. https://doi.org/10/gfzp5n
Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (Nov. 2016), 168:1–168:14. https://doi.org/10/f9cm65
Daisuke Murakami, Yoshiki Yamagata, and Toshihiro Hirano. 2020. Geostatistics and Gaussian Process Models. In *Spatial Analysis Using Big Data*, Yoshiki Yamagata and Hajime Seya (Eds.). Academic Press, 57–112. https://doi.org/10.1016/B978-0-12-813127-5.00004-7
Christopher J. Paciorek and Mark J. Schervish. 2006. Spatial Modelling Using a New Class of Nonstationary Covariance Functions. *Environmetrics* 17, 5 (Aug. 2006), 483–506. https://doi.org/10/bt6vvk
Ken Perlin. 1985. An Image Synthesizer. *Computer Graphics (Proceedings of SIGGRAPH)* 19, 3 (July 1985), 287–296. https://doi.org/10/bbsdxj
Gerald C Pomraning. 1991. *Linear Kinetic Theory and Particle Transport in Stochastic Mixtures*. Vol. 7. World Scientific Press, Singapore. https://doi.org/10/mr49
Ali Rahimi and Benjamin Recht. 2007. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates Inc., Red Hook, NY, USA, 1177–1184.
Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Mass. https://gaussianprocess.org/gpml/
Luigi M. Ricciardi and Shunsuke Sato. 1986. On the Evaluation of First Passage Time Densities for Gaussian Processes. *Signal Processing* 11, 4 (Dec. 1986). https://doi.org/10/cqq2hg
Silvia Sellán and Alec Jacobson. 2022. Stochastic Poisson Surface Reconstruction. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 6 (Dec. 2022). https://doi.org/10/mr5b
Silvia Sellán and Alec Jacobson. 2023. Neural Stochastic Screened Poisson Reconstruction. *ACM SIGGRAPH Asia Conference Papers* (2023). https://doi.org/10/mr5c
Dario Seyb, Eugene d'Eon, Benedikt Bitterli, and Wojciech Jarosz. 2024. Microfacets to Participating Media: A Unified Theory of Light Transport with Stochastic Geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 43, 4 (July 2024). https://doi.org/10/gt5nh9
Towaki Takikawa, Andrew Glassner, and Morgan McGuire. 2022. A Dataset and Explorer for 3D Signed Distance Functions. *Journal of Computer Graphics Techniques (JCGT)* 11, 2 (April 2022), 1–29. http://jcgt.org/published/0011/02/01/
Vincent Tavernier, Fabrice Neyret, Romain Vergne, and Joëlle Thollot. 2019. Making Gabor Noise Fast and Normalized. (2019). https://doi.org/10/g825mg
Yusuke Tokuyoshi and Takahiro Harada. 2019. Hierarchical russian roulette for vertex connections. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.
Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Annual Conference Series (Proceedings of SIGGRAPH)*, Vol. 29. ACM Press, 419–428. https://doi.org/10/d7b6n4
James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. 2020. Efficiently Sampling Functions from Gaussian Process Posteriors. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 10292–10302. https://doi.org/10/gt2hjn
James T. Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. 2021. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research* 22, 5 (July 2021). https://doi.org/10/gt2hjr
Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33, 4 (July 2014), 116:1–116:9. https://doi.org/10/f6cprr
Cheng Zhang and Shuang Zhao. 2020. Multi-Scale Appearance Modeling of Granular Materials with Continuously Varying Grain Properties. In *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 13 pages. https://doi.org/10/ghcp26

A GAUSSIAN PROCESSES

A Gaussian process $f(\boldsymbol{p}) \sim \mathbf{GP}(\mu(\boldsymbol{p}), \kappa(\boldsymbol{p}, \boldsymbol{p}'))$ is a random function over a d-dimensional input space, such that its evaluations at any finite number of n locations, $P \coloneqq \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, follow an n-dimensional Gaussian distribution:

$$\underbrace{\begin{bmatrix} f(\boldsymbol{p}_1) \\ \vdots \\ f(\boldsymbol{p}_n) \end{bmatrix}}_{f(P)\in\mathbb{R}^n} \sim \mathcal{N}\Bigg(\underbrace{\begin{bmatrix} \mu(\boldsymbol{p}_1) \\ \vdots \\ \mu(\boldsymbol{p}_n) \end{bmatrix}}_{\boldsymbol{\mu}(P)\in\mathbb{R}^n}, \underbrace{\begin{bmatrix} \kappa(\boldsymbol{p}_1,\boldsymbol{p}_1) & \cdots & \kappa(\boldsymbol{p}_1,\boldsymbol{p}_n) \\ \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{p}_n,\boldsymbol{p}_1) & \cdots & \kappa(\boldsymbol{p}_n,\boldsymbol{p}_n) \end{bmatrix}}_{\boldsymbol{\kappa}(P,P)\in\mathbb{R}^{n\times n}} \Bigg), \tag{27}$$

where $\mu(\boldsymbol{p}) \coloneqq \mathbb{E}[f(\boldsymbol{p})]$ is the deterministic mean function and $\kappa(\boldsymbol{p}, \boldsymbol{p}') \coloneqq \mathbb{E}[(f(\boldsymbol{p}) - \mu(\boldsymbol{p}))(f(\boldsymbol{p}') - \mu(\boldsymbol{p}'))]$ is the covariance function (kernel). A Gaussian process is *stationary* iff $\kappa(\boldsymbol{p}, \boldsymbol{p}') = \kappa(\boldsymbol{p} - \boldsymbol{p}')$, and *isotropic* iff the covariance depends only on distance $\kappa(\boldsymbol{p}, \boldsymbol{p}') = \kappa(\|\boldsymbol{p} - \boldsymbol{p}'\|)$. We use $\|\boldsymbol{p}\|$ to denote the L_2 norm of a d-dimensional vector, and $|P|$ to denote the cardinality of a set. We are concerned mostly with Gaussian processes with input dimensions $d = 1, 2, 3$. Common covariance functions in machine learning include the squared exponential, rational quadratic, and Matern kernels.

Due to the linearity of the derivative operator, the gradient $\nabla f(\boldsymbol{p})$ of a Gaussian process f is again a Gaussian process, which is jointly Gaussian distributed with the value process:

$$\begin{bmatrix} f(\boldsymbol{p}) \\ \nabla f(\boldsymbol{p}) \end{bmatrix} \sim \mathcal{N}\Bigg(\begin{bmatrix} \mu(\boldsymbol{p}) \\ \nabla\mu(\boldsymbol{p}) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{p},\boldsymbol{q}) & \nabla_{\boldsymbol{q}}^\intercal \kappa(\boldsymbol{p},\boldsymbol{q}) \\ \nabla_{\boldsymbol{p}}\kappa(\boldsymbol{p},\boldsymbol{q}) & \nabla_{\boldsymbol{p}}\nabla_{\boldsymbol{q}}^\intercal \kappa(\boldsymbol{p},\boldsymbol{q}) \end{bmatrix} \Bigg), \tag{28}$$

where $\nabla_{\boldsymbol{p}} \coloneqq \left[\partial/\partial\boldsymbol{p}_x, \partial/\partial\boldsymbol{p}_y, \partial/\partial\boldsymbol{p}_z\right]^\intercal$ denotes the gradient operator with respect to $\boldsymbol{p}$. The covariance of the gradient process is hence defined by the second derivative of the covariance.

A.1 Sampling GP priors

We can create GP realizations ("sample GP priors") in either the "function space" or "weight space" views.

Function-space. Given a set of n desired evaluation locations, we can sample their values from the joint distribution (27) as

$$f(P) = \boldsymbol{\mu}(P) + \boldsymbol{\kappa}^{\frac{1}{2}}(P,P)\,\boldsymbol{\xi}, \quad \text{with} \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{29}$$

This requires a matrix "square root" $\boldsymbol{\kappa}^{\frac{1}{2}}$, like the Cholesky decomposition, making it $O(|P|^3)$.

Weight-space. Weight-space approximations first assume that a realization f can be expressed as a linear combination of b basis functions $\Phi(\boldsymbol{p}) \coloneqq (\phi_1(\boldsymbol{p}) \ldots \phi_b(\boldsymbol{p}))^\intercal$:

$$f(\cdot) = \mu(\cdot) + \sum_{i=1}^{b} \phi_i(\cdot) w_i = \mu(\cdot) + \Phi(\cdot)^\intercal \mathbf{w}, \tag{30}$$

where $\mathbf{w}$ is a weight vector. Sampling a random weight vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ results in a realization that can be evaluated at any point, and evaluation is now $O(b|P|)$ – *linear* in the number of evaluation points $|P|$.

The covariance of realizations sampled via Eq. (30) is $\boldsymbol{\kappa}(P,P) = \boldsymbol{\Phi}(P)\,\boldsymbol{\Phi}^\intercal(P)$ where $\boldsymbol{\Phi}(P)$ is the $|P| \times b$ matrix with rows $\Phi^\intercal(\boldsymbol{p})$ for each $\boldsymbol{p} \in P$. The basis functions are typically chosen to obtain a desired covariance. A common choice for stationary kernels is random Fourier features [Rahimi and Recht 2007]. Seyb et al. [2024] dismissed weight-space methods as being largely limited to stationary processes. We show that this is not true, and that some procedural noise techniques from graphics can be cast as weight-space GPs.

A.2 Conditioning

A powerful feature of Gaussian processes is that they can be *conditioned* on a set of observations $(C, \boldsymbol{v})$, where C is a set of points on the input domain and $\boldsymbol{v}$ are the observed values at those locations, each with Gaussian uncertainty parameter σ to quantify the uncertainty in model predictions. In the applications of Gaussian process implicit surface, no prediction task is involved, so σ is commonly set to 0.

Function-space. After observing $(C, \boldsymbol{v})$, the "posterior" distribution at a set of evaluation points P is also Gaussian, with mean and covariance:

$$\begin{aligned} \mu_{|C}(P) &= \mu(P) + \kappa(P, C)[\kappa(C, C) + \sigma^2 \mathbf{I}]^{-1}(\boldsymbol{v} - \mu(C)), \\ \kappa_{|C}(P, P) &= \kappa(P, P) - \kappa(P, C)\kappa(C, C)^{-1}\kappa(C, P). \end{aligned} \tag{31}$$

Forming this posterior mean and covariance takes $O(|C|^3 + |C||P| + |P|^2)$ and then sampling – by inserting into Eq. (29) – takes $O(|P|^3)$.

Weight-space. In weight-space, we can condition by modifying the distribution of the weights in Eq. (30) to $\mathbf{w}_{|C} \sim \mathcal{N}(\mathbf{m}_{|C}, \boldsymbol{\Sigma}_{|C})$ with

$$\begin{aligned} \mathbf{m}_{|C} &= (\boldsymbol{\Phi}^\intercal(C)\boldsymbol{\Phi}(C) + \sigma^2 \mathbf{I})^{-1}\boldsymbol{\Phi}^\intercal(C)\boldsymbol{v} \\ \boldsymbol{\Sigma}_{|C} &= (\boldsymbol{\Phi}^\intercal(C)\boldsymbol{\Phi}(C) + \sigma^2 \mathbf{I})^{-1}\sigma^2. \end{aligned} \tag{32}$$

This is $O(b^3 + b|C|)$.

B FREQUENCY DOMAIN PROOF OF POSITIVE SEMI-DEFINITENESS

If we assume, for simplicity, that the kernel $h(\boldsymbol{s}, \boldsymbol{p}) = h(\boldsymbol{p} - \boldsymbol{s})$ is shift-invariant/stationary,

$$\psi_{\text{dense}}(\boldsymbol{p}) \coloneqq \int_{\mathbb{R}^d} h(\boldsymbol{p} - \boldsymbol{s})\, W(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}, \quad \text{and} \tag{33}$$

$$\kappa(\boldsymbol{p}, \boldsymbol{p}') = \sigma^2 \int h(\boldsymbol{p} - \boldsymbol{s})\, h(\boldsymbol{p}' - \boldsymbol{s})\, \mathrm{d}\boldsymbol{s}, \tag{34}$$

then we can easily show, via the Fourier transform, that Eq. (34) is a valid covariance. According to Bochner's theorem, a function k is positive semi-definite iff its Fourier transform $\mathcal{F}\{k\}$ is a nonnegative function. We recognize Eq. (34) as (up to the scale factor σ^2) the autocorrelation of h: $(h \star h)(\boldsymbol{p} - \boldsymbol{p}')$. By the convolution – product theorem, we have

$$\mathcal{F}\{\kappa\} = \sigma^2\, \mathcal{F}\{h \star h\} = \sigma^2\, \mathcal{F}\{h\}\, \overline{\mathcal{F}\{h\}} = \sigma^2\, |\mathcal{F}\{h\}|^2 \geq 0, \tag{35}$$

where $\overline{\mathcal{F}}$ is the complex conjugate. Equation (35) states that the power spectrum of h, which must be non-negative, gives the Fourier transform of the covariance κ, which therefore must be positive semi-definite. Figure 19 provides a visualization of the discussed quantities for a 2D Gabor kernel. It is possible to extend this proof to the non-stationary version (4) by using locally centered Fourier transforms.

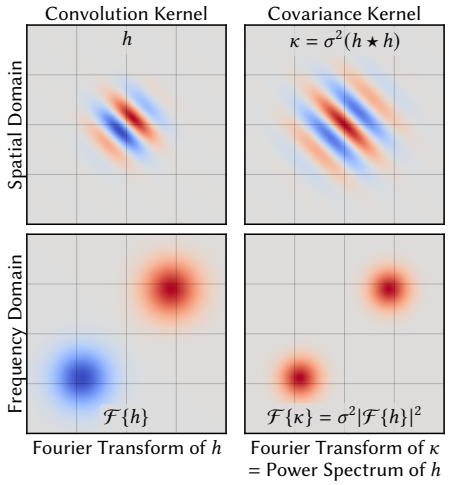


Fig. 19. We visualize the spatial and frequency domain quantities discussed in Appendix B of the 2D Gabor kernel [Gabor 1946] as the convolution kernel h. Red and blue indicate positive and negative values, respectively. Notably, while the kernel's Fourier transform $\mathcal{F}\{h\}$ may exhibit both positive and negative components, its power spectrum $\mathcal{F}\{\kappa\} = \sigma^2|\mathcal{F}\{h\}|^2$ is inherently non-negative, ensuring the positive semi-definiteness of the covariance function κ.

C COVARIANCE OF SPARSE CONVOLUTION NOISE

Inserting Eq. (9) into the definition of covariance and expanding yields

$$\kappa_\lambda(\boldsymbol{p}, \boldsymbol{p}') = \mathbb{E}\left[\psi_\lambda(\boldsymbol{p})\, \psi_\lambda(\boldsymbol{p}')\right] \tag{36a}$$

$$= \mathbb{E}\left[\left(\sum_{i=1}^{N} w_i\, h(\boldsymbol{s}_i, \boldsymbol{p})\right)\left(\sum_{i=1}^{N} w_i\, h(\boldsymbol{s}_i, \boldsymbol{p}')\right)\right] \tag{36b}$$

$$= \mathbb{E}[N]\mathbb{E}[W_\lambda^2]\mathbb{E}[h(\boldsymbol{s}, \boldsymbol{p})h(\boldsymbol{s}, \boldsymbol{p}')] \tag{36c}$$

$$= \underbrace{\lambda\mathbb{E}[W_\lambda^2]}_{\sigma^2} \int h(\boldsymbol{s}, \boldsymbol{p})\, h(\boldsymbol{s}, \boldsymbol{p}')\, \mathrm{d}\boldsymbol{s}. \tag{36d}$$

Going from Eq. (36b) to Eq. (36c) the cross-terms vanish in expectation by Campbell's theorem, and we obtain Eq. (36d) by assuming, without loss of generality, a domain volume of 1.