

Stochastic Poisson Surface Reconstruction

SILVIA SELLÁN, University of Toronto

ALEC JACOBSON, University of Toronto and Adobe Research

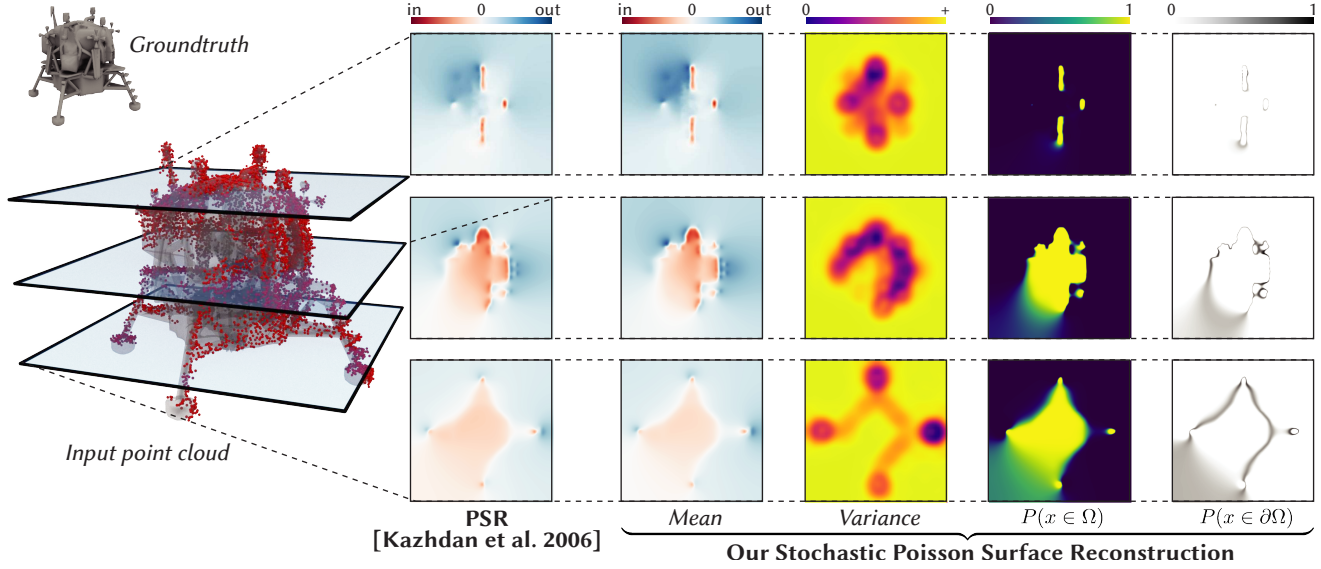


Fig. 1. Left to right: given an input point cloud, Poisson Surface Reconstruction (PSR) recovers the surface as the zero levelset of an implicit function. We propose a novel statistical derivation of PSR that exchanges the function for a distribution, allowing us to answer many statistical queries.

We introduce a statistical extension of the classic Poisson Surface Reconstruction algorithm for recovering shapes from 3D point clouds. Instead of outputting an implicit function, we represent the reconstructed shape as a modified Gaussian Process, which allows us to conduct statistical queries (e.g., the likelihood of a point in space being on the surface or inside a solid). We show that this perspective: improves PSR’s integration into the online scanning process, broadens its application realm, and opens the door to other lines of research such as applying task-specific priors.

1 INTRODUCTION

Surface reconstruction refers to the process of converting a point cloud (the most common real-world raw 3D capture format) into another shape representation, such as a mesh or an implicit function, for use in downstream applications. This is an *underdetermined* process filled with *uncertainty*, not just due to a point cloud’s discrete nature and lack of topological information but also because of real-world challenges like scan occlusions or measurement error.

The *de facto* standard geometry processing algorithm for this task is *Poisson Surface Reconstruction* (PSR) [Kazhdan et al. 2006], which solves a partial differential equation to reconstruct a function f_{PSR} whose zero levelset $f_{PSR} = 0$ defines the desired surface. Due to its speed, quality and simplicity, PSR remains relevant and has seen uses in fields as varied as digital heritage preservation [Andrade et al. 2012], topography [Gupta and Shukla 2017], medicine [Palomar et al. 2016] and autonomous driving [Vizzo et al. 2021].

Unfortunately, PSR lacks the statistical formalism to quantify the uncertainties of the surface reconstruction process. The magnitudes of f_{PSR} outside of the zero levelset are arbitrary (see Fig. 2) and f_{PSR} alone cannot provide an answer to statistical questions crucial to the reconstruction process like “how confident can one be of the values of f_{PSR} ?” or “where should one aim the scanner next to optimize information gain?” Similarly, it cannot respond to queries like “what is the probability of a point p being contained in the shape?”, critical for collision detection or ray casting applications.

In this paper, we introduce *Stochastic Poisson Surface Reconstruction* (SPSR), a statistical derivation of PSR as conditional probability distributions in a Gaussian Process (GP). Instead of just an implicit function value, we endow every point in space with a Gaussian distribution of possible values. We propose an algorithm to compute the mean and variance that fully determine this distribution (see Fig. 1), allowing us to answer any statistical queries.

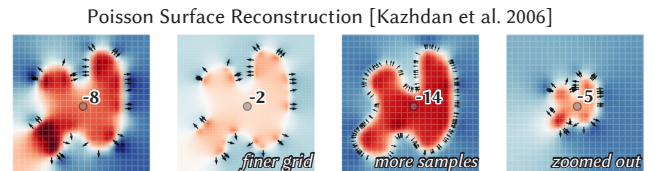


Fig. 2. PSR values outside of zero are arbitrary (e.g., they depend on grid size, sampling rate and scale) and contain no direct statistical information.

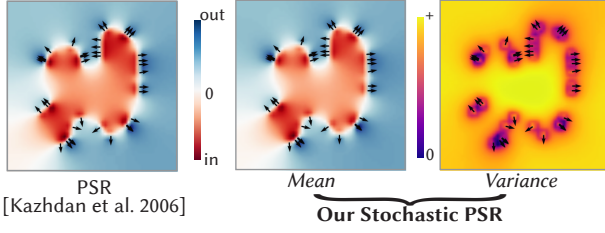


Fig. 3. Our Stochastic PSR extends the traditional PSR into a statistical distribution whose mean is nearly identical to the PSR output.

This extension vastly broadens the use cases of Poisson Surface Reconstruction, as we highlight with prototypical examples of surface point cloud repair, ray casting, next-view planning and collision detection. Furthermore, we show that by understanding PSR from this new perspective, we can borrow from the Gaussian Process literature to modify it by incorporating task-specific priors, opening several promising lines of future research.

2 RELATED WORK

A complete survey of vast research areas like uncertainty quantification or surface reconstruction is beyond the scope of this work. Instead, we focus this section on setting our Stochastic PSR in its context of bridging the gap between PSR and Gaussian Processes.

2.1 Surface Reconstruction from Point Clouds

Point clouds are a common raw format for 3D geometry acquired from the real world. However, most applications in fields like rendering, simulation and geometry processing require more structured representations like triangle meshes. Thus, reconstructing surfaces from point clouds is a well-studied, fundamental problem in Computer Graphics (see [Berger et al. 2017] for an exhaustive survey). While some methods convert point clouds directly to meshes (e.g., by dictionary learning [Xiong et al. 2014]) or simple primitives (e.g., [Monszpart et al. 2015; Nan et al. 2010]), we focus on those that extract the shape as the zero levelset of a reconstructed function f .

Within these, a common separation is made between *local* and *global* algorithms. Local algorithms prioritize performance in speed and memory; for example, by fitting linear [Hoppe et al. 1992], polynomial [Alexa et al. 2001] or higher-order [Fuhrmann and Goesele 2014] functions to restricted point subsets. By their nature, these are more susceptible to oscillations far from the sample points. To cope, global algorithms (e.g., [Jacobson et al. 2013]) allow f to be influenced by every point in the cloud (hierarchical fast summation structures can help reduce computation [Barill et al. 2018]).

Poisson Surface Reconstruction (PSR) [Kazhdan et al. 2006] captures the best features of the global (robustness) and local (performance) methods by computing f in two steps. First, a vector field \vec{V} is interpolated from the point cloud using only local information. Then, f is obtained from \vec{V} via a global sparse PDE solve, which is discretized as a linear system and can be solved very efficiently using an adaptive grid structure. PSR has been improved since its publication; for example, Kazhdan and Hoppe [2013] combine both steps to improve noise robustness, Kazhdan et al. [2020] include

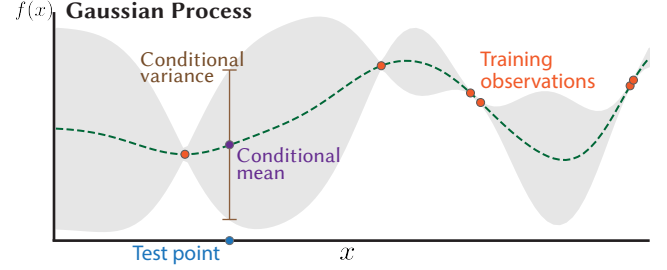


Fig. 4. A sample Gaussian Process applied to a supervised learning task. Given some training observations (orange), any unobserved test point is given a conditional distribution with a mean (purple) and variance (brown).

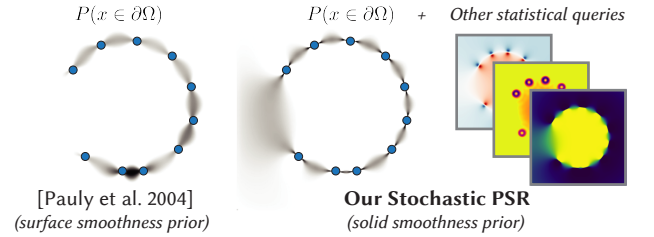


Fig. 5. Pauly et al. [2004] use a surface smoothness prior to quantify uncertainty in reconstruction. We use a *solid* smoothness prior to compute a full statistical distribution, from which we can also query surface quantities.

envelope constraints and Peng et al. [2021] formulate the Poisson solve in a differentiable way. Nonetheless, the original 2006 publication is still one of the few showing robustness in every metric considered by Berger et al. [2017] more than a decade later.

Our Stochastic PSR inherits all the benefits of the original PSR, supplying it with a complete statistical formalism that extends it and its application realm (see Fig. 3). Our contributions are orthogonal to the specific grid structure used (see [Kazhdan and Hoppe 2019]).

Our algorithm can output the probability of any point in space being inside the sampled domain (see Fig. 1). While this resembles the shape representations proposed by *occupancy networks* [Mescheder et al. 2019] and *neural radiance fields* [Mildenhall et al. 2020], we note that our algorithm produces this quantity as a direct byproduct of a fully determined statistical distribution which can answer many other statistical queries, like boundary probabilities (see Fig. 5) or regional probabilities (see Fig. 12).

2.2 Gaussian Processes

A Gaussian Process (GP) is an infinite collection of joint normal distributions [Doob 1944; Dudley 2018], usually parametrized by a continuous parameter like time or space [Kac and Siebert 1947]. Recently, Gaussian processes have been used as a tool in unsupervised learning (see [Williams and Rasmussen 2006] for an introduction, [Engel et al. 2005; Raissi et al. 2017] for examples), even suggested initially as an alternative to neural networks by MacKay [1997].

Closer to our application are Gaussian Process Implicit Surfaces (GPIS) [Williams and Fitzgibbon 2006]. These algorithms exchange the $f(x)$ function that implicitly defines a surface for a Gaussian distribution $f(x) \sim \mathcal{N}(\mu(x), \sigma(x))$, and compute $\mu(x)$ and $\sigma(x)$ by

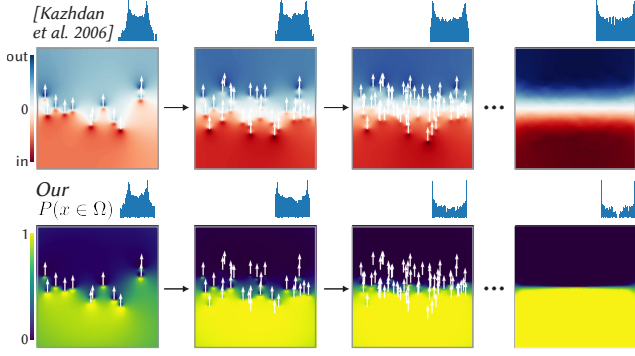


Fig. 6. Our SPSR provides a probability that accounts for sampling and cannot be recovered from the PSR values. Histograms in logarithmic scale.

studying the posterior GP distribution given observed points. One can recover a surface by extracting the zero levelset of μ ; however, the information contained in σ also finds use in tasks like robotic grasping [Dragiev et al. 2011], next view planning [Hollinger et al. 2012] and segmentation [Ramon Soria et al. 2017; Shin et al. 2017].

GPIS present the same global/local dilemma as other surface reconstruction algorithms. If the interpolation is done using all the point cloud information (this is encoded in the support of the GP covariance function), the method suffers in performance; if not, in robustness. By using the insights of Poisson Surface Reconstruction, our Stochastic PSR instead uses a local Gaussian Process to build the distribution of the gradient field $\nabla f(x)$, and then recover the full distribution of $f(x)$ with a global PDE solve. While using a GP to interpolate vector-valued data is unorthodox, we note it has been done before; e.g., for fluid velocity information [Lee et al. 2019].

Gaussian processes are a more traditional approach to quantifying the uncertainty of a regression model, a field which has seen significant growth since recent advances in deep learning (see [Abdar et al. 2021] for a survey). Often, the posterior distribution of a given neural network is approximated by another Bayesian network whose parameters minimize a chosen distribution loss. (see e.g., [Xue et al. 2019]). Alternatively, the posterior may be learned directly from the observed data (see e.g., [Shen et al. 2021]).

2.3 Stochastic Geometry Processing

Our algorithm stands among many similar works that add statistical formalism to standard geometry processing techniques, like point cloud registration. Specifically, a lot of work has been dedicated to computing the covariances in the pairwise iterative closest point [Bosse and Zlot 2008; Landry et al. 2019] and, most recently, for general multi-scan registration [Cao et al. 2018; Huang et al. 2020].

Closer to our application, Curless and Levoy [1996] compute truncated signed distance fields for each sensor position and use a model for their individual uncertainty to weigh their contributions to the final implicit reconstruction. Pöthkow et al. [2011] model grid samples of an implicit function as normal distributions and calculate the individual probabilities of each voxel marching cubes configuration to quantify the contouring uncertainty. Sharf et al.

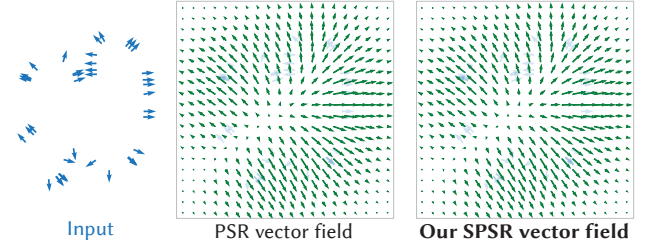


Fig. 7. Our SPSR vector field \vec{V}_{SPSR} (right), which uses a symmetrized version k_{SPSR} of the traditional PSR covariance k_{PSR} , is visually identical to the PSR vector field \vec{V}_{PSR} (center) for a representative input point cloud (left). In the language of Section 4, the right-most subfigure shows the mean of our vector field Gaussian Process $\vec{V}(q)$ after covariance lumping.

[2007] similarly measure topological reconstruction uncertainty to identify where user-provided disambiguation is most needed.

Pauly et al. [2004] work is the most similar to ours. It takes a point cloud as input and uses a surface smoothness prior to compute the likelihood of any point in space lying on it (see Fig. 5, left). Instead, our proposed algorithm assumes samples to lay on the boundary of a solid, and impose solid smoothness prior (examined further in Section 5). Further, while our algorithm can also output a surface likelihood quantity (see Fig. 5, center), it is only part of a full statistical distribution of the solid (see Fig. 5, right).

3 BACKGROUND

By posing PSR as a Gaussian Process, our work combines these two concepts. We begin by reviewing them individually.

3.1 Gaussian Processes

Intuitively, a Gaussian Process is an extension of the multivariate normal distribution to an infinite number of dimensions. Formally, let $\mathcal{A} = \{A(x)\}_{x \in D}$ be a collection of random variables parametrized by some continuous parameter x . \mathcal{A} is said to be a Gaussian Process if any finite subset of \mathcal{A} follows a multivariate Gaussian distribution. Equivalently, \mathcal{A} is a Gaussian Process if for any two $x, x' \in \Omega$,

$$A(x), A(x') \sim \mathcal{N} \left(\begin{bmatrix} m(x) \\ m(x') \end{bmatrix}, \begin{bmatrix} k(x, x) & k(x, x') \\ k(x', x) & k(x', x') \end{bmatrix} \right) \quad (1)$$

for some *mean* and *covariance* functions $m : \Omega \rightarrow \mathbb{R}, k : \Omega \times \Omega \rightarrow \mathbb{R}$. These two functions uniquely determine the Gaussian Process \mathcal{A} .

Gaussian Processes are a particularly useful tool for supervised learning tasks (see Fig. 4). Assume training observations $\mathcal{T} = \{(x_i, a_i)\}_{i=1}^n$, where each a_i is an observation of the distribution $A(x_i)$, and consider an unobserved (test) point x . By definition of GP, the joint distribution of $\{A(x), A(x_1), \dots, A(x_n)\}$ is

$$\mathcal{N} \left(\begin{bmatrix} m(x) \\ m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x, x) & k(x, x_1) & \dots & k(x, x_n) \\ k(x_1, x) & k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x) & k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix} \right) \quad (2)$$

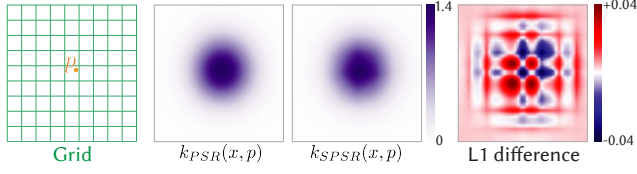


Fig. 8. To interpret PSR as a Gaussian Process, we define k_{SPSR} , a minor modification of the PSR semicovariance k_{PSR} . These are visually similar and their difference is small (a maximum of 2%).

which we write as

$$A(x), A(x_1), \dots, A(x_n) \sim \mathcal{N} \left(\begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \begin{bmatrix} k_1 & k_2^\top \\ k_2 & K_3 \end{bmatrix} \right), \quad (3)$$

where it is relevant to note that K_3 depends only on the training data and k_2 depends on both training and test sets. By Bayes' theorem, this means the distribution of $A(x)$ conditioned on the observations $\{(x_i, a_i)\}_{i=1}^n$ is

$$A(x) | \mathcal{T} \sim \mathcal{N}(m_1 + k_2^\top K_3^{-1} (a - m_2), k_1 - k_2^\top K_3^{-1} k_2) \quad (4)$$

One usually assumes $m = 0$ (otherwise, the Gaussian process $\mathcal{A} - m$ is considered), and writes

$$A(x) | \mathcal{T} \sim \mathcal{N}(k_2^\top K_3^{-1} a, k_1 - k_2^\top K_3^{-1} k_2). \quad (5)$$

In the case of noisy observations, a term $\sigma_n^2 I$ (where σ_n^2 is the *noise variance*) is added to K_3 .

3.2 Poisson Surface Reconstruction

Any input oriented point cloud of the surface of a solid shape Ω can be written as a set of observations $s \in \mathcal{S}$, each of them storing a position p_s and a (normalized) orientation \vec{N}_s . Poisson Surface Reconstruction [Kazhdan et al. 2006] aims to build a function $f_{PSR} : \mathbb{R}^3 \rightarrow \mathbb{R}$ which takes positive values inside Ω and negative values outside of it, thus making the zero levelset $f_{PSR} = 0$ the reconstructed surface $\partial\Omega$.

PSR begins by building a grid \mathcal{O} . Then, they define the following vector field for any $q \in \mathbb{R}^3$ by using the grid structure to interpolate the orientation observations:

$$\vec{V}_{PSR}(q) = \sum_{s \in \mathcal{S}} \frac{1}{W(p_s)} \sum_{o \in B(s)} \alpha_{o,p_s} F_o(q) \vec{N}_s \quad (6)$$

where $B(s)$ are the eight closest grid nodes to s , α_{o,p_s} is the trilinear interpolation weight for p_s at o , W is a measure of volumetric sampling density and F_o is a compactly-supported approximation of a Gaussian kernel centered at the o -th node.

The fundamental observation of PSR is that once \vec{V} has been constructed, the desired implicit function f satisfies

$$\Delta f_{PSR}(x) = \nabla \cdot \vec{V}_{PSR}(x), \quad \forall x \in \mathcal{O} \quad (7)$$

Eq. (7) is underdetermined, as wildly different functions can have the same Laplacian. This ambiguity is resolved in part during discretization. Kazhdan et al. [2006] suggest building \mathcal{O} as an adaptive grid of a bounding box of the point cloud. Then, they use the finite element method to build discrete Laplacian and divergence operators L and Z and solve

$$L f_{PSR} = Z v_{PSR}. \quad (8)$$

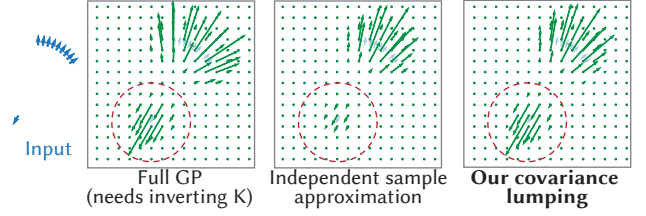


Fig. 9. One can avoid the GP sample covariance matrix inversion (center left) by assuming samples to be independent (center right). This makes magnitudes proportional to sampling density (see highlight). Our *covariance lumping* approximates the full GP with invariant magnitudes (right).

This discretization imposes zero Neumann $\nabla f \cdot \vec{n} = 0$ boundary conditions on the boundary of \mathcal{O} . Even so, Eq. (8) only determines f up to translation. To account for this, one valid f_{PSR} is computed and then shifted so its values near the observation points are zero on average. The solution f_{PSR} contains the implicit function f_{PSR} evaluated at each grid node, and its zero levelset can be extracted using contouring algorithms like Marching Cubes [Lorensen and Cline 1987] or Dual Contouring [Ju et al. 2002].

4 STOCHASTIC POISSON SURFACE RECONSTRUCTION

We introduce our *Stochastic PSR*, which extends the traditional PSR from Kazhdan et al. [2006] with the statistical formalism of a Gaussian Process. We begin by defining the *PSR semicovariance* as

$$k_{PSR}(x, y) = \sigma_g \sum_{o \in B(x)} \alpha_{o,x} F_o(y), \quad (9)$$

where σ_g is a scalar parameter. Then, the vector field interpolation in Eq. (6) can be written as

$$\vec{V}_{PSR}(q) = \sum_{s \in \mathcal{S}} k_{PSR}(p_s, q) \frac{1}{\sigma_g W(p_s)} \vec{N}_s. \quad (10)$$

By term identification, it may be tempting to interpret this interpolation directly as the posterior mean of a Gaussian Process $k_2^\top K_3^{-1} y$ (see Eq. (5)), with normals as the observed values y , $k_{PSR}(p_s, q)$ as the entries of k_2 and division by $\sigma_g W(p_s)$ playing the role of multiplication by K_3^{-1} . However, a valid Gaussian distribution must have a symmetric covariance, and $k_{PSR}(p_s, q) \neq k_{PSR}(q, p_s)$ in general (hence the *semi* in “PSR semicovariance”). To circumvent this, we define the *Stochastic PSR covariance* to be its symmetrized version

$$k_{SPSR}(x, y) = k_{SPSR}(y, x) = \frac{1}{2} (k_{PSR}(x, y) + k_{PSR}(y, x)), \quad (11)$$

which analogously defines a *Stochastic PSR vector field*

$$\vec{V}_{SPSR}(q) = \sum_{s \in \mathcal{S}} k_{SPSR}(p_s, q) \frac{1}{\sigma_g W(p_s)} \vec{N}_s \quad (12)$$

This variance symmetrization will be our only deviation from the traditional Poisson Surface Reconstruction by Kazhdan et al. [2006] (see Fig. 8). Strictly speaking, all the observations that follow in this paper can only be said to apply to \vec{V}_{SPSR} ; however, note that \vec{V}_{PSR} and \vec{V}_{SPSR} are visually indistinguishable (see Fig. 7) and their difference vanishes under refinement (see proof in Appendix A).

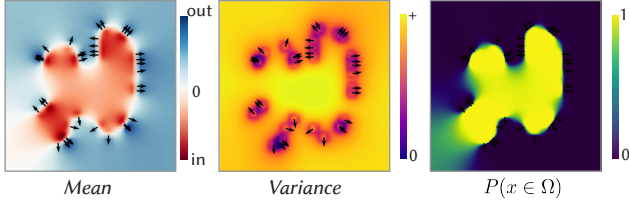


Fig. 10. By adding a variance, we can consistently compute the probability of any point in space being in the sampled object.

Our critical observation is that the interpolation in Eq. (12) can be interpreted as the mean of a supervised learning task under the assumptions of a Gaussian Process with $m = 0$ and covariance k_{SPSR} . Indeed, one needs only to exchange A for the vector-valued \vec{V} and the observations $\{(x_i, a_i)\}$ for $\{(p_s, \vec{N}_s)\}$ in Eq. (5) to obtain the conditional probability distribution

$$\vec{V}(q) | \mathcal{S} \sim \mathcal{N}(k_2^\top K_3^{-1} \vec{N}_s, k_1 - k_2^\top K_3^{-1} k_2) \quad (13)$$

where

$$k_1 = k(q, q), \quad k_2 = (k(q, p_s))_{s \in \mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}, \quad (14)$$

$$K_3 = (k(p_s, p_{s'}))_{s, s' \in \mathcal{S}} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}. \quad (15)$$

The main computational cost in computing the mean and variance of the conditional distribution is the inversion of K_3 , a matrix whose size scales with the number of points in the cloud. One could avoid this by assuming that the samples \mathcal{S} are independent, which would result in approximating K_3 by $\sigma_g I$. However, completely discarding the sample interdependency could have adverse effects: in practice, it would make densely sampled regions have an outsized effect over more sparsely sampled ones (see Fig. 9, center right).

We introduce a better approximation: we assume each sample is independent, but with variance proportional to sampling density. Intuitively, this means we account for the interdependence by trusting each individual sample in a densely sampled region less than in a sparsely sampled one (see Fig. 9, right). We justify this choice mathematically in the general GP context in Appendix B.

We are not aware of any previous work that proposes this approximation, which addresses one of the main performance limitations of general Gaussian Processes. Numerically, this resembles the mass matrix lumping step often carried out in the Finite Element literature (see e.g., [Zienkiewicz et al. 2005] Chap. 16.2.4.), so we call it the (diagonal) *lumped covariance matrix*

$$D := \text{diag}(\sigma_g w) \approx K_3 \quad (16)$$

where w is the local sampling density at each sample point, which we compute as described by Kazhdan et al. [2006]. As is usual with Gaussian processes, we sum $\sigma_n I$ if the sampled point cloud is assumed to contain noise with variance σ_n in the normal vector.

Then, the conditional distribution becomes

$$\vec{V}(q) | \mathcal{S} \sim \mathcal{N}(k_2^\top D^{-1} \vec{N}_s, k_1 - k_2^\top D^{-1} k_2) \quad (17)$$

Importantly, note that the mean of this distribution is *exactly* the Stochastic PSR vector field we introduced in Eq. (12), i.e.,

$$\vec{V}(q) | \mathcal{S} \sim \mathcal{N}(\vec{V}_{SPSR}(q), k_1 - k_2^\top D^{-1} k_2). \quad (18)$$

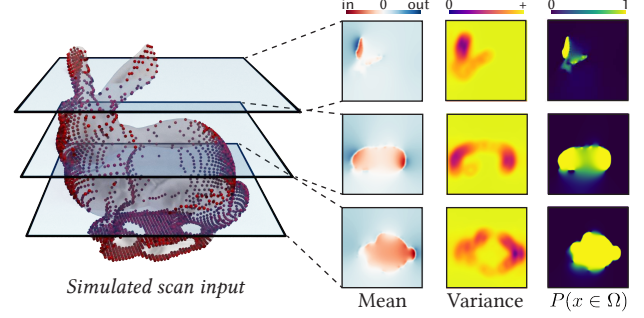


Fig. 11. Our extension of PSR provides a mean and variance, which can be used to compute probabilities.

In other words, this critical reinterpretation of PSR has allowed us to extend the interpolated PSR vector field into a complete statistical distribution whose mean \vec{V}_{SPSR} is nearly identical to \vec{V}_{PSR} (see Fig. 7, which we can now understand as comparing the traditional PSR vector field to the mean of our lumped Gaussian Process).

Ideally, however, we would want a similar statistical formalism not for \vec{V} but for the implicit function f , which would allow us to formulate statistical queries meaningful to the reconstruction task, e.g., $P(f(q) < 0)$.

Fortunately, transferring our statistical understanding of \vec{V} to f is just a matter of Gaussian arithmetic. First, note that a reasoning identical to the one above leads to the joint conditional probability of \vec{V} at all grid nodes being

$$\vec{V}(o_1), \dots, \vec{V}(o_{|O|}) | \mathcal{S} \sim \mathcal{N}(V_{SPSR}, K_1 - K_2^\top D^{-1} K_2), \quad (19)$$

where

$$K_1 = (k(o, o')) \in \mathbb{R}^{|O| \times |O|}, \quad K_2 = (k(o, p_s)) \in \mathbb{R}^{|O| \times |\mathcal{S}|} \quad (20)$$

and

$$V_{SPSR} = (\vec{V}_{SPSR}(o)) \in \mathbb{R}^{|O| \times 3}. \quad (21)$$

For simplicity, denote $K_V = K_1 - K_2^\top D^{-1} K_2$, and let v be the concatenated vector of \vec{V} values as in the previous section. Then, Eq. (19) becomes

$$v | \mathcal{S} \sim \mathcal{N}(V_{SPSR}, K_V). \quad (22)$$

By linearity and bilinearity of the mean and covariance, respectively, we know

$$Zv | \mathcal{S} \sim \mathcal{N}(ZV_{SPSR}, ZK_V Z^\top). \quad (23)$$

and thus, since the vector f of evaluations of f satisfies $Lf = Zv$,

$$f | \mathcal{S} \sim \mathcal{N}\left(L^{-1} ZV_{SPSR}, L^{-1} ZK_V Z^\top (L^\top)^{-1}\right), \quad (24)$$

Similarly to the traditional PSR, we shift the mean values to be zero near the sample points and shift the variance values to have a minimum diagonal entry of zero. We write the above distribution as

$$f | \mathcal{S} \sim \mathcal{N}(f_{SPSR}, K_f). \quad (25)$$

This statement is our principal contribution, as it extends the traditional PSR implicit function f_{PSR} into a full statistical distribution whose mean is almost identical to f_{PSR} . Of all the information contained in K_f , a particularly useful quantity is its diagonal σ_f^2 , which contains the variances of each entry of f (see Fig. 3).

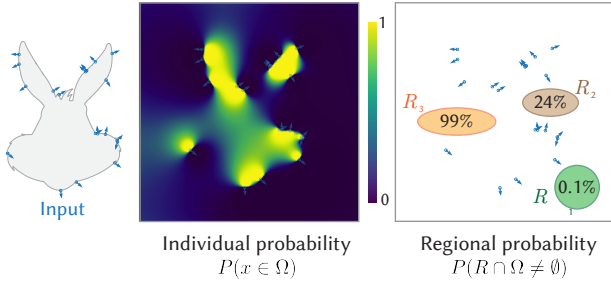


Fig. 12. Our algorithm not only allows us to compute individual spatial probabilities (center), but also joint regional probabilities (right).

4.1 Statistical queries

Eq. (25) contains all the information we need to respond to statistical queries fundamental to the reconstruction process; e.g., the probability of a given point being inside the shape (see Fig. 10)

$$p(o_i \in \Omega) = p(f_i \leq 0) = CDF_{f_{SPSR,i}, \sigma_i^2}(0) \quad (26)$$

or the probability density of being on the surface (see Fig. 13)

$$p(o_i \in \Omega) = p(f_i = 0) = PDF_{f_{SPSR,i}, \sigma_i^2}(0) \quad (27)$$

where CDF_{μ, σ^2} and PDF_{μ, σ^2} are the cumulative distribution and probability density functions, respectively, of a Gaussian distribution with mean μ and variance σ^2 . Similar Gaussian arithmetic lets us compute other quantities like confidence intervals (see Fig. 14).

All these queries would be impossible to answer from PSR's f_{PSR} alone. We show this in the didactic example in Fig. 6, where vectors pointing upward are sampled from a normal distribution centered at the middle horizontal line. As more samples get added, f_{PSR} converges to a smooth transition between positive and negative values; however, our variances are progressively decreasing, making $P(x \in \Omega)$ converge in certainty as expected.

We formalize this observation further by defining an integrated statistical quantity, the *total uncertainty* U_{SPSR} , which we define as

$$U_{SPSR} = \int_B (0.5 - |P(x \in \Omega) - 0.5|) dx \quad (28)$$

where B is a bounding box around the point cloud. U_{SPSR} can be intuitively interpreted as a global measure of reconstruction quality which converges to zero when the surface has been reconstructed with absolute certainty (see Fig. 15). We pose that this quantity can be used as a stopping criteria to guide the scanning process (see Fig. 17). We know of no analogous quantity that can be formally defined from the traditional PSR understanding alone.

The off-diagonal entries of K_f quantify the correlation between the values of f at different points in \mathcal{O} . This allows for *joint probability queries*; for example, computing the likelihood of a whole region in space R intersecting Ω (see Fig. 12), a quantity of immediate relevance to collision detection applications.

To compute it, we can randomly sample $R_s = \{r_1, \dots, r_s\} \subset R$. If W is the linear interpolation matrix from \mathcal{O} to R_s , then distribution arithmetic says

$$f(r_1, \dots, r_s) | S \sim \mathcal{N}(W f_{SPSR}, W^T K_f W). \quad (29)$$

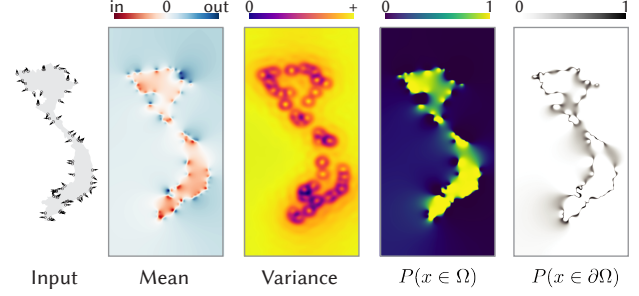


Fig. 13. Our computed mean and variance (center left) completely determine the implicit function's distribution, allowing us to respond to statistical queries like the likelihood of a point being in the volume defined by the point cloud (center right) or on its surface (right).

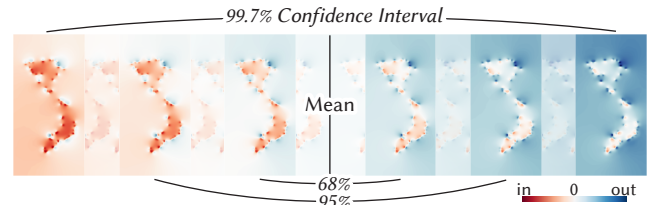


Fig. 14. The 68-95-99.7 rule from normal distributions lets us compute confidence intervals for the implicit function values. Regions with the same color (red or blue) at both extremes of an interval can be said to be inside or outside with at least the interval's confidence.

The collision probability, i.e., the probability that any r_i is contained in the shape, is opposite to the joint probability that all r_i are outside the shape, $p(f(r_1) > 0, \dots, f(r_s) > 0)$. This quantity is also the same as $p(-f(r_1) < 0, \dots, -f(r_s) < 0)$, known as the *multivariate Gaussian cumulative distribution* of $-f$, which can be estimated with numerical integration (see Figs. 12 and 20).

4.2 Space reduction with eigenspace analysis

All this additional information comes at a computational cost. Specifically, the main performance bottleneck is the computation of the covariance matrix

$$K_f = L^{-1} Z K_V Z^T (L^T)^{-1}, \quad (30)$$

a matrix equation that amounts to solving $2 \times |\mathcal{O}|$ linear systems. Even making use of efficient prefactorizations of L , this step is impracticable for large 3D grids with sizes in the millions of cells.

To circumvent this, we will work in the reduced space of Laplacian eigenvectors. For a cuboid with lengths ℓ_1, ℓ_2, ℓ_3 and zero Neumann boundary conditions, these are known [Gottlieb 1985] to respond to the analytic function

$$\psi_{M,N,\tilde{N}}(x, y, z) = \cos\left(\frac{M\pi x}{\ell_1}\right) \cos\left(\frac{N\pi y}{\ell_2}\right) \cos\left(\frac{\tilde{N}\pi z}{\ell_3}\right), \quad (31)$$

with associated eigenvalues

$$\lambda_{M,N,\tilde{N}} = \pi^2 \left[\left(\frac{M}{\ell_1}\right)^2 + \left(\frac{N}{\ell_2}\right)^2 + \left(\frac{\tilde{N}}{\ell_3}\right)^2 \right]. \quad (32)$$

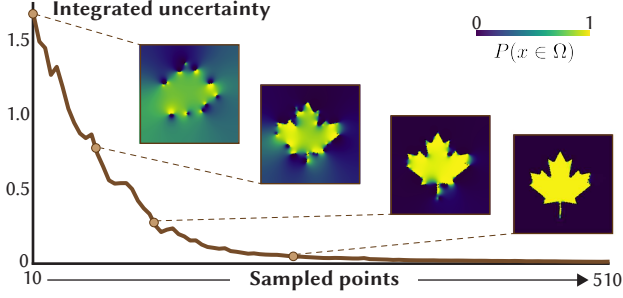


Fig. 15. Our statistical formalism means we can define quantities like the integrated uncertainty, which converges to zero as the probability is collapsed to zero and one. This measure can serve as a scanning stopping criterion.

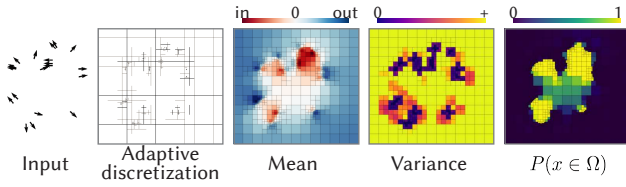


Fig. 16. Our contribution is orthogonal to the choice of discretization, as we show by answering statistical queries on a graded quadtree.

Specifically, we will evaluate these analytical expressions into $E \in \mathbb{R}^{|O| \times k}$, the matrix of the k lowest-magnitude eigenvectors of L , and D_e , the diagonal eigenvalue matrix. Then, we approximate K_f by projecting $ZK_V Z^T$ to the eigenspace, solving the matrix equation in this reduced space, and reprojecting; i.e.,

$$K_f \approx E D_e^{-1} E^T (Z K_V Z^T) E D_e^{-1} E^T. \quad (33)$$

In all our examples, we fix $k = 3000$.

Due to the dimensionality of the above matrices, the mostly computationally expensive step in Eq. (33) is the product $E C E^T$, where $C \in \mathbb{R}^{k \times k}$ results from multiplying all middle matrices in Eq. (33). We avoid this step by noting that it is rare that one is interested in all entries of K_f . As seen in Section 4.1, most statistical queries require only the diagonal σ_f^2 , which we can compute directly and efficiently as $(E C) \cdot E$, where \cdot denotes row-wise dot product. In the case of joint probability queries that require off-diagonal knowledge of K_f , we first construct the selection matrices S, S^T that extract the covariance rows and columns relevant to the specific application, and directly compute $E' C E'^T$, where $E' = S E$. Thus, we eventually avoid ever computing (or storing) the full $K_f \in \mathbb{R}^{|O| \times |O|}$.

Dimension reduction has been used for implicit reconstruction before; e.g., the Fourier coefficients computed by Kazhdan [2005]. We limit our approximation to the covariance computation only.

4.3 Adaptive discretization

The statistical formalism we contribute is agnostic to where the values of f and \vec{V} are stored and to the discretization scheme used to solve the Poisson equation. Kazhdan et al. [2006] suggest building an adaptive grid which is finer near the sampled point cloud. This choice is justified through their sole goal of accurately recovering the zero levelset of f .

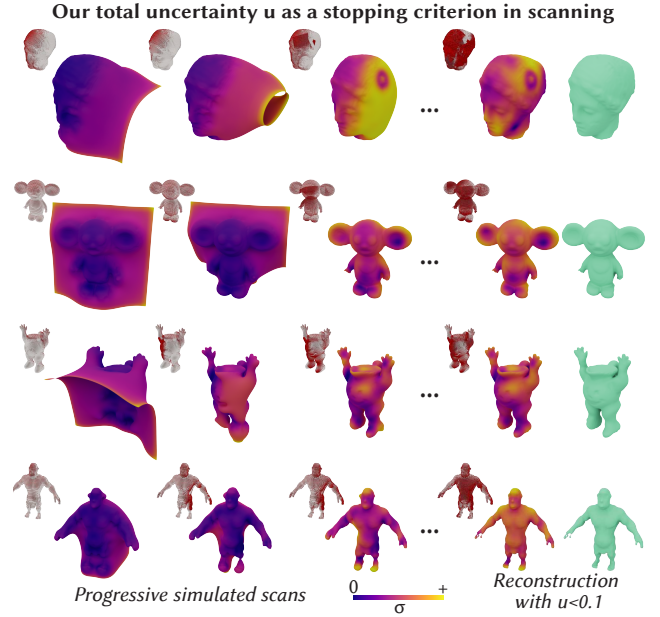


Fig. 17. Like PSR, we can extract isosurfaces of the mean of our Stochastic PSR. Unlike PSR, our statistical formalism provides a reliable stopping criterion by thresholding integrated uncertainty u . Meshes colored by variance.

Our Stochastic PSR extends the traditional PSR by providing volumetric statistical information, which is also relevant and non-trivial in regions away from the zero levelset of our mean function f_{SPSR} or the point cloud (see, e.g., the probability in Fig. 10). Further, computing statistical queries accurately away from the mean-zero levelset can be critical for applications that our formalism newly allows like collision detection or ray casting (see Fig. 24). Thus, the choice of an adaptive grid structure is less obvious for our algorithm. Nonetheless, we show in Fig. 16 our algorithm working as expected on a graded quadtree grid structure, following the finite difference discretization proposed by Bickel et al. [2006]. While orthogonal to our contribution, we believe exploring different adaptive discretization strategies (e.g., narrowing in on regions of higher uncertainty) to be a promising avenue for future work.

5 BEYOND POISSON SURFACE RECONSTRUCTION

A significant benefit of our interpretation of Poisson Surface Reconstruction as a Gaussian Process is that we may borrow from the vast literature on the latter to study variations of the traditional PSR. A prototypical example of this is incorporating task-specific priors.

In Section 3.1, we mentioned that most Gaussian Process consider $m = 0$ and are thus fully determined by the covariance k (this is often known as *kriging* in the GP literature). Indeed, it was by assuming $m = 0$ that we recovered the PSR vector field interpolation as the mean of the GP posterior distribution. In fact, the choice of m is a prior imposed on the joint distribution before any data is observed. Thus, PSR can be understood to have a zero-gradient prior.

Incidentally, this observation also provides an answer to an often asked question about PSR; namely, why PSR, which is posed as recovering a function which is zero on the surface and satisfies certain

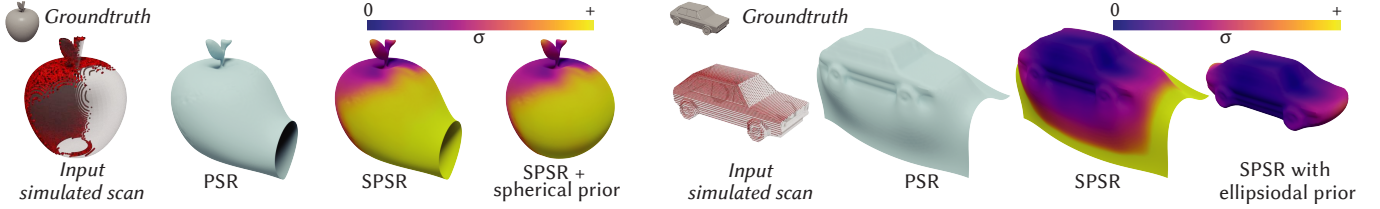


Fig. 18. Taking inspiration from Figs. 6 and 7 in [Martens et al. 2016], we show the result of reconstructing an apple and a car from a partial scan. PSR fails to produce even a closed surface, as does our vanilla SPSR, albeit also providing variance information signaling the less confident regions. When combined with a task-specific simple primitive (spherical or ellipsoidal) prior, SPSR provides a significantly better reconstruction.

norm-one gradients, produces an output so different from a Signed Distance Field (SDF). From this novel Gaussian Process perspective, we can understand that PSR is, via its $m = 0$ prior, encouraging zero-norm gradients away from the sample positions, while an SDF requires Eikonal norm one gradients almost everywhere.

Primitive geometric priors. Martens et al. [2016] suggest using simple geometric primitives as priors in the context of GP implicit surfaces. Since the authors are directly learning the surface’s implicit representation, their suggested priors m are scalar, SDF-like functions. In our case, where learning is carried out in the gradient space and then transferred to an implicit function via a PDE solve, we can use gradient vector-valued priors.

In Fig. 19, we exemplify the effect of a simple spherical prior

$$m(x) = \alpha \frac{x - c}{\|x - c\|}, \quad (34)$$

which we pose can be a useful tool for favouring closed reconstructions over open ones. α is a parameter tuning the strength of the prior and c is the center of the sphere, which we set to be the average of all point cloud positions. In Fig. 18, we show less didactical uses of this same prior, along with a similar ellipsoidal one.

6 IMPLEMENTATION DETAILS

We implemented our algorithm in PYTHON, using LIBIGL [Jacobson et al. 2018] for common geometry processing subroutines. We rendered all our figures in BLENDER, using BLENDERTOOLBOX [Liu 2022]. All our results were produced on our machine with Intel Xeon CPU E5-2637 v3 3.50Hz (16 cores) with 64GB of RAM.

For parameter standardization, we normalized all our examples to fit a length-one cube as a preprocessing step. Unless specified otherwise, we use a 100^3 uniform grid (100^2 in the two-dimensional examples) such that L and Z become the usual finite difference Laplacian and divergence matrices. We fix $\sigma_g = 0.02$, $\alpha = 0.05$ and $k = 3000$. We use the same function F_0 as Kazhdan et al. [2006], obtained by convolving a box filter with itself three times.

For memory efficiency, we use SciPy’s biconjugate gradient method to solve for the distribution mean f_{SPSR} , which accounts for approximately 5% of our runtime (around 10 seconds in all 3D examples). Our main computational bottleneck is Eq. (33) (specifically, the projection of K_V into the reduced space), covering 90% of our runtime (around two minutes in all 3D examples).

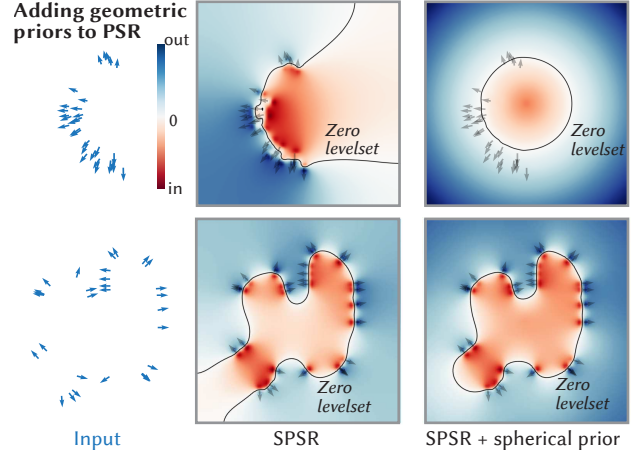


Fig. 19. Adding a spherical prior to our Stochastic PSR reconstruction helps us recover spherical objects and also avoid open surfaces in partial scans.

7 EXPERIMENTS

7.1 Deviation from Kazhdan et al. [2006]

Our sole deviation from traditional Poisson Surface Reconstruction as introduced by Kazhdan et al. [2006] is the symmetrization of the covariance in Eq. (37). Theoretically, it is clear that both k_{PSR} and k_{SPSR} converge to the same symmetric $F_x(y)$ in the limit of grid refinement. However, we also justify the step by showing that the difference between these two is small even at a very coarse resolution in Fig. 8. In PSR, k_{PSR} is used to reconstruct a vector field \vec{V}_{PSR} . In Fig. 7, we show that the interpolated vector field \vec{V}_{SPSR} obtained with the symmetrized covariance is visually identical. In Fig. 3, we show that this applies also to the implicit function f obtained through the Poisson solve.

7.2 Statistical quantities

Our Stochastic PSR exchanges the functional PSR output for a fully defined statistical distribution. To explore just how much more information is contained in this distribution, we show different statistical quantities meaningful to the shape reconstruction process.

In Fig. 10, we use our computed mean and variance to evaluate each point’s cumulative distribution functions at zero, which returns the probability of any point in space being contained in Ω . Similarly, evaluating the probabilistic density function at zero returns a measure of surface probability, as we show in Fig. 13. We



Fig. 20. We can use our statistical formalism to compute statistical quantities like collision chances (bold), unlike traditional PSR (grey).

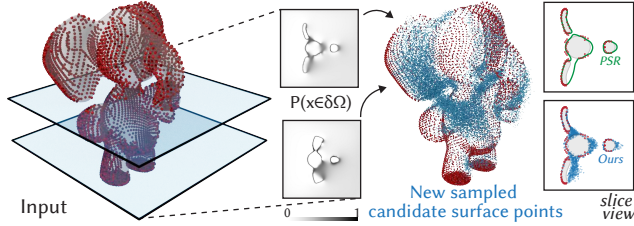


Fig. 21. We can use our computed surface likelihood to repair the input point cloud, naturally drawing points from less certain regions.

use slice planes to visualize these functions for a 3D reconstruction problem in Fig. 1 and Fig. 11. In Fig. 14, we show our statistical formalism applied to the computation of confidence intervals, which serve as an intuitive tool for visualizing variances.

8 APPLICATIONS

Our main contribution is a novel theoretical understanding of Poisson Surface Reconstruction that expands it beyond its traditional application realm. We explore this with prototypical results.

8.1 Collision detection

Statistical information is critical for robots or autonomous vehicles, which may capture their surroundings as point clouds with partial occlusions. When a car is deciding whether to swerve, it does not want to know only if the manoeuvre is secure *for the most likely scenario of its surroundings* (the traditional PSR output or the SPSR mean); rather, it must account for uncertainty and conclude whether the manoeuvre is safe *in the vast majority of possible scenarios*.

We exemplify this in Fig. 20, where we simulate a 3D scan of a street and use joint probability queries to compute the collision chance of an automated vehicle along a given trajectory. We compare this chance against a baseline (in grey), which only checks for intersections between the car mesh and the traditional PSR zero isosurface extracted with Marching Cubes [Lorensen and Cline 1987]. We include synthetic Gaussian noise both on the input point positions and normal vectors. In Fig. 22, we show how one can threshold our computed probabilities $p(x \in \Omega)$ to obtain isosurfaces for use in collision detection applications.

8.2 Volume rendering

The arbitrary units in the traditional PSR output mean that it cannot be interpreted as a volume rendering occupancy. Fortunately, our Stochastic Poisson Surface Reconstruction allows us to compute $p(x \in \Omega)$, values between 0 and 1 which can be interpreted as occupancies (this reinterpretation is common in the Neural Radiance Field literature, e.g., [Mildenhall et al. 2020]). In Fig. 24, we use

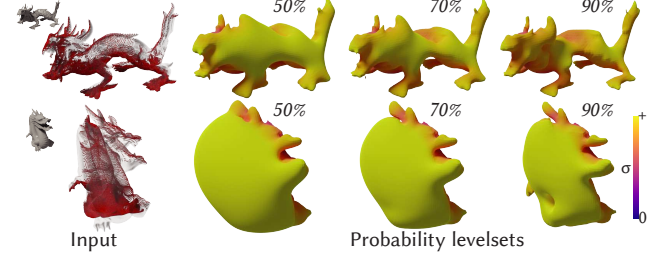


Fig. 22. We can threshold our computed probability $p(x \in \Omega)$ to obtain meshes that can be used for, e.g., collision detection.

standard volume rendering techniques (see [Pharr et al. 2016], Chap. 11) to sample the free path of a ray aimed at a partial reconstruction of a Space Wizard vehicle from two different directions. As expected, rays intersecting regions with higher uncertainty produce a wider spread of paths, as evidenced by the orange and blue histograms.

8.3 Reconstruction

Scanning integration. Surface reconstruction is the flagship application of PSR. Despite this, PSR cannot be fully integrated into the scanning pipeline, as it fails to provide basic feedback like when a point cloud is dense enough or where to scan next. In Fig. 15, we show that our introduced *total uncertainty* u converges to zero as points are added, giving a clear measure of scan quality. We show how one can use thresholds on u as a scanning stopping criteria in Fig. 17, obtaining reconstructions of similar quality for the same threshold value. In Fig. 23, we show our algorithm’s output on a real-world depth scan obtained with a smartphone app.

The ability to simulate ray casting on our reconstructed shapes also allows us to score different potential scan positions, as we show in Figs. 25 and 26. We begin by simulating a scan on a given groundtruth object using traditional ray tracing in a cone around some predetermined camera positions (see Fig. 25) placed on the left side of a race car. We include synthetic Gaussian noise both on the input point positions and normal vectors. This lets us obtain a point cloud P which we can input to our algorithm and compute its total uncertainty $U_{SPSR}(P)$. For each potential new scan position, we cast a ray and add the resulting intersection p as a new point to the input point cloud, defining the *camera score* as the change in total uncertainty from adding the new point (see Fig. 26),

$$s = |U_{SPSR}(P) - U_{SPSR}(P \cup p)| \quad (35)$$

We repeat this process ten times for each potential camera position, to account for the statistical variability in the ray casting. A fundamental part of any scan feedback pipeline, this process helps us decide on a next best camera view. As expected, scores are highest for hypothetical cameras on the right side of the car.

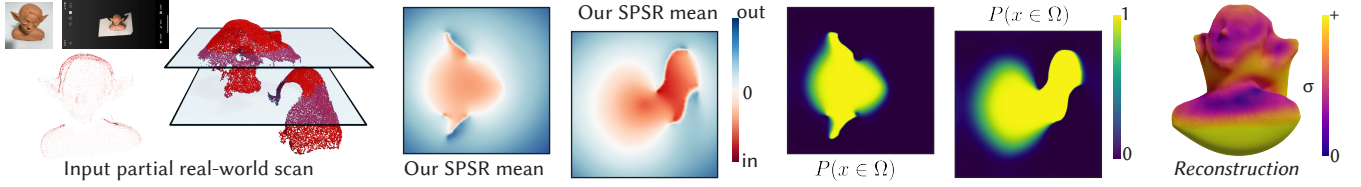


Fig. 23. Our SPSR on real-world scan data. The geometry of our reconstructed surface is the same as PSR, but we provide variances and volumetric quantities.

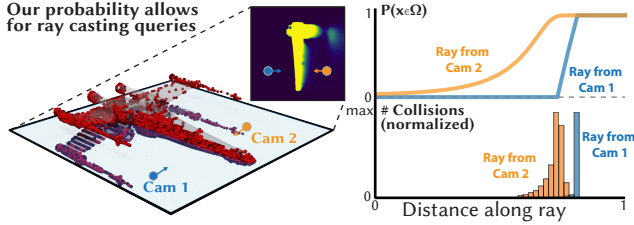


Fig. 24. Our probability can be interpreted as a density for ray casting applications that have no analogue in the traditional PSR formulation.

Point cloud repair. Converting to an implicit function or a triangular mesh is not needed for some downstream applications that can work directly on point clouds. However, raw scan clouds often contain holes due to occlusions or missing camera angles. In Fig. 21, we show how our algorithm can be used to produce possible surface points in the occluded regions by using a Metropolis-Hastings scheme to sample from our surface probability $p(x \in \Omega)$.

Incorporating priors. Our novel statistical interpretation of PSR allows us to build on it even as it applies to its most direct purpose of recovering surfaces from point clouds; for example, by incorporating geometric priors as we exemplify in Fig. 19. In Fig. 18, we show how even very simple primitive geometric priors can be useful for recovering complex three-dimensional geometry.

9 LIMITATIONS & CONCLUSION

Our work merges the renowned Poisson Surface Reconstruction with the field of Gaussian Process Implicit Surfaces, both providing a statistical formalism for PSR and a novel GPIS that can be used for supervised learning of implicit functions.

By using covariance functions with compact support like the original PSR, we sample the interpolated vector field \vec{V} efficiently. However, unlike other GPIS, we require a global Poisson solve to evaluate even a single implicit function f mean or variance query. When seen as a GPIS, this is a limitation that makes our algorithm less efficient for applications that require a limited amount of samples. Also like the original PSR, our algorithm requires an oriented point cloud as input, and extending it to unoriented point clouds (e.g., those representing thin sheets [Chi and Song 2021]) would require orienting them as a preprocessing step (see, e.g., [Alliez et al. 2007; Hornung and Kobbelt 2006; Metzner et al. 2021]).

While computing the mean of our distribution x_{SPSR} is just as computationally expensive as the original PSR output, our covariance matrix computation requires solving a full matrix equation. We alleviate this by precomputing the Laplacian eigenpairs on a bounding box of any input. This introduces a trade-off between memory, runtime and precision when computing this covariance.

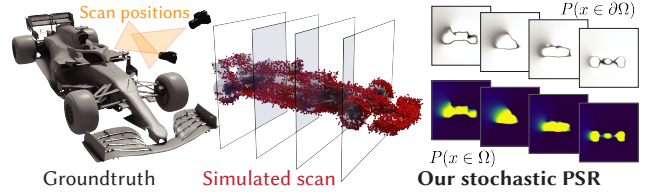


Fig. 25. Many of our results follow a similar pipeline: we use a groundtruth object (left) and simulate scanning it from different directions to obtain an oriented point cloud (middle), which we then pass as input to our Stochastic PSR algorithm (right) to compute the desired statistical quantity.

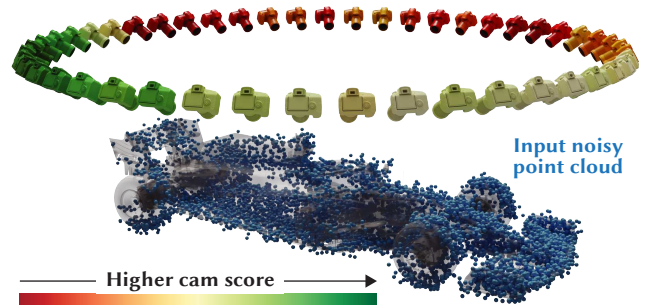


Fig. 26. Interpreting our probability as a density and simulating rays from different cameras, we can score them according to the simulated change in total uncertainty to aid next view planning.

Our algorithm extends the traditional PSR by Kazhdan et al. [2006] by separating it into a GP vector field reconstruction and a PDE solve. It is unclear how our approach could be applied to the *Screened PSR* by Kazhdan and Hoppe [2013], which combines both steps to improve robustness in unsampled regions. We conjecture that the answer may lie in works on incorporating gradient observations to Gaussian processes (see 9.4 in [Williams and Rasmussen 2006]).

Our work not only extends our knowledge of Poisson Surface Reconstruction from the perspective of a Gaussian Process. Indeed, we have also used PSR to expand our knowledge of Gaussian Processes, by introducing a covariance lumping technique which avoids costly matrix inversions at test time. We believe applying this in the broader GP context to be a promising avenue for future work.

As 3D acquisition becomes more accessible and attractive to consumers and the lines between real and virtual geometry get blurred, we believe quantifying and transmitting the uncertainties of the capture process to be one of the fundamental problems of this era. We hope our work inspires our geometry processing colleagues, whose workflow often involves PSR, to study how this statistical formalism carries over further down the shape processing pipeline.

ACKNOWLEDGMENTS

This project is funded in part by NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, the DSI Catalyst Grant program and gifts by Adobe Inc. The first author is funded in part by an NSERC Vanier Scholarship and an Adobe Research Fellowship.

We acknowledge the authors of the 3D models used throughout this paper and thank them for making them available for academic use: ShaggyDude (Fig. 1, CC BY 4.0), Perry Engel (Fig. 17 third row, CC BY-NC 4.0), Joshua Poh (Fig. 18 left, CC BY-SA 3.0), Agustin Flowalistik (Fig. 18 right, CC BY-NC-SA 4.0), Will McKay (Fig. 20, CC BY-NC 4.0), Karl (Fig. 24, CC BY-SA 3.0) and Rafael Rodrigues (Figs. 25 and 26, CC BY-NC-SA 4.0).

We thank Kirill Serkh, Kiriakos Kutulakos, Eitan Grinspun, David I.W. Levin, Oded Stein, Nicholas Sharp, Otman Benchechroun and Towaki Takikawa for insightful conversations that inspired and guided us throughout our work; Abhishek Madan, Aravind Ramakrishnan and Jonathan Panuelos for proofreading; Hsueh-Ti Derek Liu for help rendering our results; Xuan Dam, John Hancock and all the University of Toronto Department of Computer Science research, administrative and maintenance staff that literally kept our lab running during very hard years.

REFERENCES

- Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76 (2021), 243–297.
- Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. 2001. Point set surfaces. In *Proceedings Visualization*.
- Pierre Alliez, David Cohen-Steiner, Yiyi Tong, and Mathieu Desbrun. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Eurographics SGP*.
- Beatriz Trincão Andrade, Caroline Mazetto Mendes, Jurandir de Oliveira Santos Jr, Olga Regina Pereira Bellon, and Luciano Silva. 2012. 3D preserving XVIII century baroque masterpiece: Challenges and results on the digital preservation of Aleijadinho's sculpture of the Prophet Joel. *Journal of Cultural Heritage* 13, 2 (2012), 210–214.
- Gavin Barill, Neil G Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM ToG* (2018).
- Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. 2017. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*.
- Bernd Bickel, Martin Wicke, and Markus Gross. 2006. Adaptive simulation of electrical discharges. In *Proceedings of Vision, Modeling, and Visualization*. 209–216.
- Michael Bosse and Robert Zlot. 2008. Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research* 27, 6 (2008), 667–691.
- Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. 2018. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 1–16.
- Cheng Chi and Shuran Song. 2021. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3324–3333.
- Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 303–312.
- J Li Doob. 1944. The elementary Gaussian processes. *The Annals of Mathematical Statistics* 15, 3 (1944), 229–282.
- Stanimir Dragiev, Marc Toussaint, and Michael Gienger. 2011. Gaussian process implicit surfaces for shape estimation and grasping. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2845–2850.
- Richard M Dudley. 2018. *Real analysis and probability*. CRC Press.
- Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*.
- Simon Fuhrmann and Michael Goesele. 2014. Floating scale surface reconstruction. *ACM Transactions on Graphics (ToG)* 33, 4 (2014), 1–11.
- HPW Gottlieb. 1985. Eigenvalues of the Laplacian with Neumann boundary conditions. *The ANZIAM Journal* 26, 3 (1985), 293–309.
- Sharad Kumar Gupta and Dericks P Shukla. 2017. 3D reconstruction of a landslide by application of UAV and structure from motion. In *20th AGILE conference on geographic information science*. 9–12.
- Geoffrey A Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S Sukhatme. 2012. Uncertainty-driven view planning for underwater inspection. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 4884–4891.
- Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques*. 71–78.
- Alexander Hornung and Leif Kobbelt. 2006. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing*. 41–50.
- Xiangru Huang, Zhenxiao Liang, and Qixing Huang. 2020. Uncertainty quantification for multi-scan registration. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 130–1.
- Alec Jacobson, Ladislav Kavan, and Olga Sorkine. 2013. Robust Inside-Outside Segmentation using Generalized Winding Numbers. *ACM Trans. Graph.* 32, 4 (2013).
- Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. <https://libigl.github.io/>.
- Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 339–346.
- Mark Kac and Arnold JF Siegert. 1947. On the theory of noise in radio receivers with square law detectors. *Journal of Applied Physics* 18, 4 (1947), 383–397.
- Michael Kazhdan. 2005. Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing*. 73–es.
- Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics SGP*.
- Misha Kazhdan, Ming Chuang, Szymon Rusinkiewicz, and Hugues Hoppe. 2020. Poisson surface reconstruction with envelope constraints. In *Computer graphics forum*, Vol. 39. Wiley Online Library, 173–182.
- Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.
- Misha Kazhdan and Hugues Hoppe. 2019. An Adaptive Multi-Grid Solver for Applications in Computer Graphics. In *Computer graphics forum*.
- David Landry, François Pomerleau, and Philippe Giguere. 2019. CELLO-3D: Estimating the Covariance of ICP in the Real World. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 8190–8196.
- Ki Myung Brian Lee, Chanyeol Yoo, Ben Hollings, Stuart Anstee, Shoudong Huang, and Robert Fitch. 2019. Online estimation of ocean current from sparse GPS data for underwater vehicles. In *2019 ICRA*. IEEE, 3443–3449.
- Hsueh-Ti Derek Liu. 2022. Blender Toolbox: A set of Python scripts for rendering 3D shapes in Blender 3.2. <https://github.com/HTDerekLiu/BlenderToolbox>.
- William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987).
- David JC MacKay. 1997. Gaussian processes—a replacement for supervised neural networks? (1997).
- Wolfram Martens, Yannick Poffet, Pablo Ramón Soria, Robert Fitch, and Salah Sukkarieh. 2016. Geometric priors for gaussian process implicit surfaces. *IEEE Robotics and Automation Letters* 2, 2 (2016), 373–380.
- Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Gal Metzger, Rana Hanocka, Denis Zorin, Raja Giryes, Daniele Panozzo, and Daniel Cohen-Or. 2021. Orienting point clouds with dipole propagation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.
- 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*. Springer, 405–421.
- Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. 2015. RAPter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.* 34, 4 (2015), 103–1.
- Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2010. Smartboxes for interactive urban reconstruction. In *ACM SIGGRAPH 2010 papers*.
- Rafael Palomar, Faouzi A Cheikh, Bjørn Edwin, Azeddine Beghdadhi, and Ole J Elle. 2016. Surface reconstruction for planning and navigation of liver resections. *Computerized Medical Imaging and Graphics* 53 (2016), 30–42.
- Mark Pauly, Niloy J Mitra, and Leonidas J Guibas. 2004. Uncertainty and variability in point cloud surface data. In *PBG*. 77–84.
- Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. 2021. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems* 34 (2021).
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

- Kai Pöthkow, Britta Weber, and Hans-Christian Hege. 2011. Probabilistic marching cubes. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 931–940.
- Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2017. Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.* 348 (2017).
- Pablo Ramon Soria, Fouad Sukkar, Wolfram Martens, Begoña C Arrue, and Robert Fitch. 2017. Multi-view probabilistic segmentation of pome fruit with a low-cost RGB-D camera. In *Iberian Robotics Conference*. Springer, 320–331.
- Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. 2007. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 43–es.
- Jianxiong Shen, Adria Ruiz, Antonio Agudo, and Francesc Moreno-Noguer. 2021. Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 972–981.
- Myung-Ok Shin, Gyu-Min Oh, Seong-Woo Kim, and Seung-Woo Seo. 2017. Real-time and accurate segmentation of 3-D point clouds based on Gaussian process regression. *IEEE Transactions on Intelligent Transportation Systems* 18, 12 (2017), 3363–3377.
- Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. 2021. Poisson surface reconstruction for LiDAR odometry and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5624–5630.
- Christopher K Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.
- Oliver Williams and Andrew Fitzgibbon. 2006. Gaussian process implicit surfaces. In *Gaussian Processes in Practice*.
- Shiyao Xiong, Juyong Zhang, Jianmin Zheng, Jianfei Cai, and Ligang Liu. 2014. Robust surface reconstruction via dictionary learning. *ACM ToG* (2014).
- Yujia Xue, Shiyi Cheng, Yunzhe Li, and Lei Tian. 2019. Reliable deep-learning-based phase imaging with uncertainty quantification. *Optica* 6, 5 (2019), 618–629.
- Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. 2005. *The finite element method: its basis and fundamentals*. Elsevier.

A SPSR COVARIANCE CONVERGENCE PROOF

We will prove that the difference between the PSR semicovariance

$$k_{PSR}(x, y) = \sigma_g \sum_{o \in N(x)} \alpha_{o,x} F_o(y) \quad (36)$$

and our Stochastic PSR covariance

$$k_{SPSR}(x, y) = k_{SPSR}(y, x) = \frac{1}{2}(k_{PSR}(x, y) + k_{PSR}(y, x)), \quad (37)$$

vanishes at the limit of grid refinement. We can do so by showing that $k_{PSR}(x, y)$ converges to the gridless covariance function

$$k^*(x, y) = \sigma_g F_y(x) \quad (38)$$

Let λ be the Lipschitz constant of F . Then, since F is symmetric,

$$|F_o(y) - F_y(x)| = |F_y(o) - F_y(x)| \leq \lambda \|o - x\| \quad (39)$$

Since trilinear interpolation weights sum up to one, this means

$$|k_{PSR}(x, y) - k^*(x, y)| \leq \sigma_g \lambda \|o - x\|, \quad (40)$$

Symmetrically, we get

$$|k_{PSR}(y, x) - k^*(x, y)| \leq \sigma_g \lambda \|o - x\|, \quad (41)$$

which, by definition of k_{SPSR} , means

$$|k_{SPSR}(x, y) - k_{PSR}(x, y)| \leq \sigma_g \lambda \|o - x\|. \quad \square \quad (42)$$

B JUSTIFICATION FOR COVARIANCE LUMPING

Consider a simple Gaussian Process $\mathcal{A} = \{A(x)\}_{x \in D}$ with zero mean and covariance function $k : D \times D \rightarrow \mathbb{R}$, with two different training observations $\{(x_1, a_1), (x_2, a_2)\}$ and one test point x_3 . For clarity, let $k_{ij} = k(x_i, x_j)$. Then, since $k_{11} = k_{22}$, $k_{12} = k_{21}$, we have

$$K_3 = \begin{pmatrix} k_{11} & k_{12} \\ k_{12} & k_{11} \end{pmatrix}, \quad k_2 = \begin{pmatrix} k_{13} \\ k_{23} \end{pmatrix} \quad (43)$$

Asymptotic justification for covariance matrix lumping

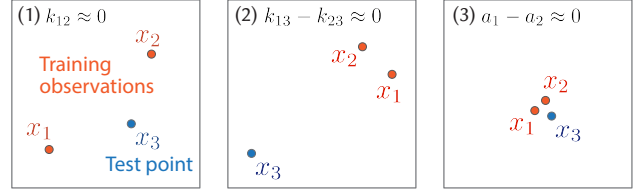


Fig. 27. Cases where our lumping has correct asymptotics: (1) when training data are far from one another, (2) when test points are far from training data and (3) when training data are so close that their features are similar.

Traditionally, one uses these matrices to compute the GP posterior mean $k_2^\top K_3^{-1} a$ and covariance $k_2^\top K_3^{-1} k_2$. Let us write both of these as $k_2^\top K_3^{-1} c$ for some generic feature vector c . Then,

$$k_2^\top K_3^{-1} c = \frac{1}{k_{11}^2 - k_{12}^2} ((k_{11}k_{13} - k_{12}k_{23})c_1 + (k_{11}k_{23} - k_{12}k_{13})c_2)$$

which is

$$\frac{1}{k_{11}^2 - k_{12}^2} ((k_{11} - k_{21})k_{13}c_1 + (k_{11} - k_{21})k_{23}c_2 + k_{12}(k_{23} - k_{13})(c_1 - c_2)),$$

or, equivalently,

$$k_2^\top K_3^{-1} c = \frac{k_{13}c_1 + k_{23}c_2}{k_{11} + k_{12}} + \frac{k_{12}(k_{23} - k_{13})(c_1 - c_2)}{k_{11}^2 - k_{12}^2}, \quad (44)$$

Note that the denominator $k_{11} + k_{12}$ is (assuming k is monotonically decreasing with distance) nothing but a measure of sampling density, meaning that we write the left term in the sum as

$$k_2^\top K_3^{-1} c = k_2^\top D^{-1} c + \frac{k_{12}(k_{23} - k_{13})(c_1 - c_2)}{k_{11}^2 - k_{12}^2}, \quad (45)$$

where D is our lumped covariance matrix. Therefore,

$$|k_2^\top K_3^{-1} c - k_2^\top D^{-1} c| \leq \frac{k_{12}}{k_{11}^2 - k_{12}^2} |k_{23} - k_{13}| |c_1 - c_2| \quad (46)$$

Thus, while our lumping introduces error, it is bounded and has the correct asymptotic behaviour (see Fig. 27 when the training samples are far from one another ($k_{12} \rightarrow 0$), when the test samples are far enough from the training samples ($k_{23} - k_{13} \rightarrow 0$), and when the training samples are close enough that the training features are close ($c_1 - c_2 \rightarrow 0$). Assuming independent samples *without* accounting for sampling density would have meant approximating $k_{11} + k_{12}$ in Eq. (44) by a constant factor, instead of recognizing it as a measure of the relative positions of x_1 and x_2 . This would have introduced further error and abandoned the asymptotic convergence.