Stochastic Poisson Surface Reconstruction

SILVIA SELLÁN, University of Toronto
ALEC JACOBSON, University of Toronto and Adobe Research

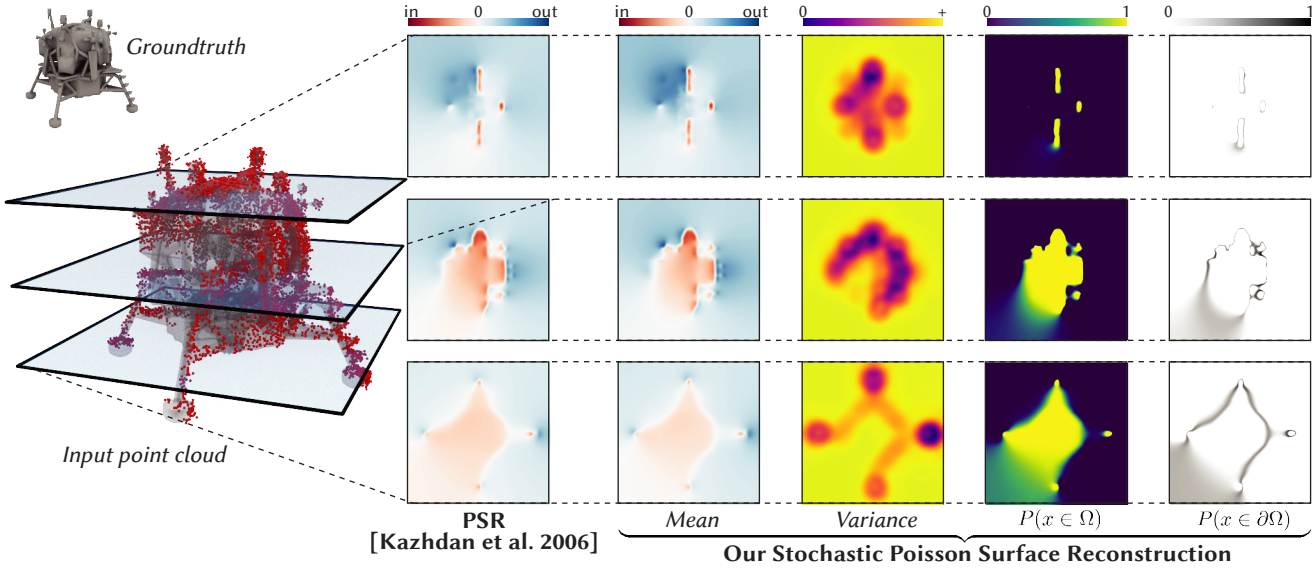


Fig. 1. Left to right: given an input point cloud, Poisson Surface Reconstruction (PSR) recovers the surface as the zero levelset of an implicit function. We propose a novel statistical derivation of PSR that exchanges the function for a distribution, allowing us to answer many statistical queries.

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.

ACM Reference Format:
Silvia Sellán and Alec Jacobson. 2022. Stochastic Poisson Surface Reconstruction. *ACM Trans. Graph.* 41, 6, Article 227 (December 2022), 12 pages. https://doi.org/10.1145/3550454.3555441

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.

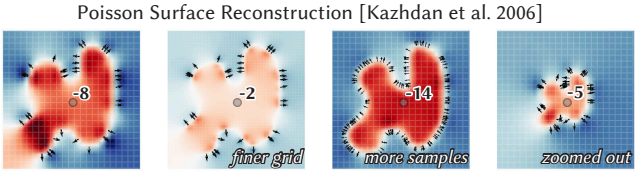


Fig. 2. PSR values outside of zero are arbitrary (e.g., they depend on grid size, sampling rate and scale) and contain no direct statistical information.

Authors' addresses: Silvia Sellán, sgsellan@cs.toronto.edu, University of Toronto; Alec Jacobson, jacobson@cs.toronto.edu, University of Toronto, Adobe Research.

0730-0301/2022/12-ART227 $15.00
https://doi.org/10.1145/3550454.3555441

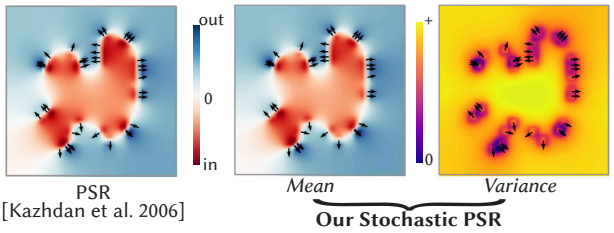


Fig. 3. Our Stochastic PSR extends the traditional PSR into a statistical distribution whose mean is nearly identical to the PSR output.

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.

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 form 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.

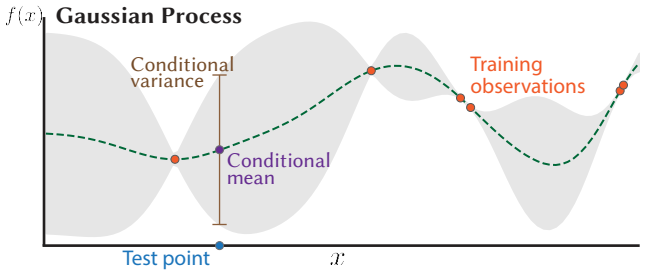


Fig. 4. A sample Gaussian Process applied to a supervised learning task. Given some training observations (orange), any unobserved test point is given a conditional distribution with a mean (purple) and variance (brown).

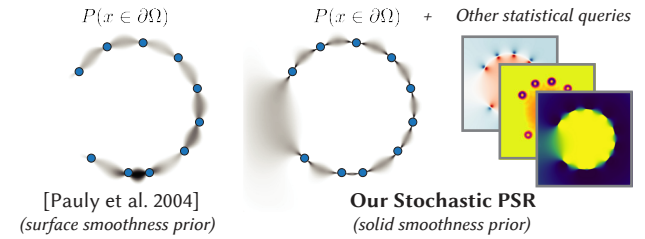


Fig. 5. Pauly et al. [2004] use a surface smoothness prior to quantify uncertainty in reconstruction. We use a *solid* smoothness prior to compute a full statistical distribution, from which we can also query surface quantities.

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 adaptative 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 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 Siegert 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].

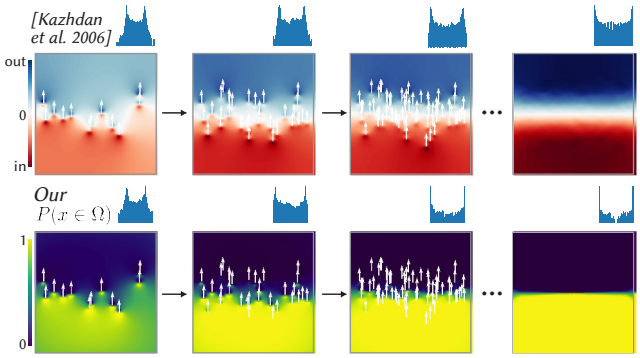


Fig. 6. Our SPSR provides a probability that accounts for sampling and cannot be recovered from the PSR values. Histograms in logarithmic scale.

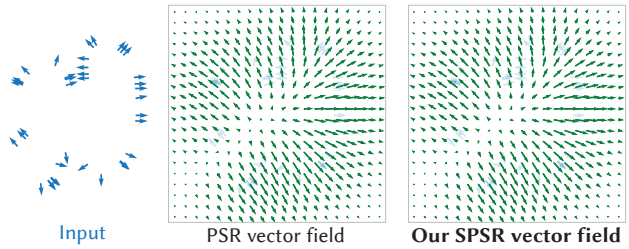


Fig. 7. Our SPSR vector field $\vec{V}_{SPSR}$ (right), which uses a symmetrized version k_{SPSR} of the traditional PSR covariance k_{PSR}, is visually identical to the PSR vector field $\vec{V}_{PSR}$ (center) for a representative input point cloud (left). In the language of Section 4, the right-most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

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 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. [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 $x, x' \in \Omega$,

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

for some *mean* and *covariance* functions $m : \Omega \to \mathbb{R}, k : \Omega \times \Omega \to \mathbb{R}$. 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 $\{A(x), A(x_1), \ldots, A(x_n)\}$ is

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

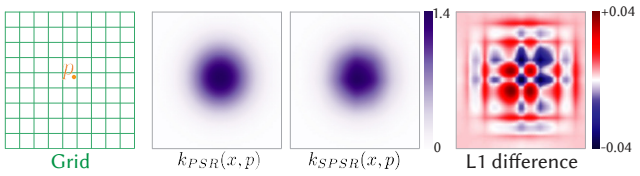


Fig. 8. To interpret PSR as a Gaussian Process, we define k_{SPSR}, a minor modification of the PSR semicovariance k_{PSR}. These are visually similar and their difference is small (a maximum of 2%).

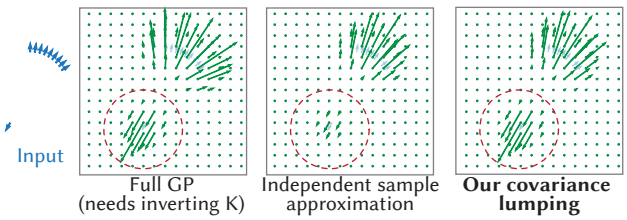


Fig. 9. One can avoid the GP sample covariance matrix inversion (center left) by assuming samples to be independent (center right). This makes magnitudes proportional to sampling density (see highlight). Our *covariance lumping* approximates the full GP with invariant magnitudes (right).

which we write as

$$A(x), A(x_1), \ldots, A(x_n) \sim \mathcal{N}\left(\begin{bmatrix} m_1 \\ \mathbf{m}_2 \end{bmatrix}, \begin{bmatrix} k_1 & \mathbf{k}_2^\top \\ \mathbf{k}_2 & \mathbf{K}_3 \end{bmatrix}\right), \tag{3}$$

where it is relevant to note that $\mathbf{K}_3$ depends only on the training data and $\mathbf{k}_2$ 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

$$A(x) \,|\, \mathcal{T} \sim \mathcal{N}(m_1 + \mathbf{k}_2^\top \mathbf{K}_3^{-1}(\mathbf{a} - \mathbf{m}_2), k_1 - \mathbf{k}_2^\top \mathbf{K}_3^{-1}\mathbf{k}_2) \tag{4}$$

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

$$A(x) \,|\, \mathcal{T} \sim \mathcal{N}(\mathbf{k}_2^\top \mathbf{K}_3^{-1}\mathbf{a}, k_1 - \mathbf{k}_2^\top \mathbf{K}_3^{-1}\mathbf{k}_2)\,. \tag{5}$$

In the case of noisy observations, a term $\sigma_n^2\mathbf{I}$ (where σ_n^2 is the *noise variance*) is added to $\mathbf{K}_3$.

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 $f_{PSR} : \mathbb{R}^3 \to \mathbb{R}$ 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:

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

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

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

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 $\mathbf{L}$ and $\mathbf{Z}$ and solve

$$\mathbf{L}\mathbf{f}_{PSR} = \mathbf{Z}\mathbf{v}_{PSR}. \tag{8}$$

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 $\mathbf{f}$ up to translation. To account for this, one valid $\mathbf{f}_{PSR}$ is computed and then shifted so its values near the observation points are zero on average. The solution $\mathbf{f}_{PSR}$ 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

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

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

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

By term identification, it may be tempting to interpret this interpolation directly as the posterior mean of a Gaussian Process $\mathbf{k}_2^\top \mathbf{K}_3^{-1}\mathbf{y}$ (see Eq. (5)), with normals as the observed values $\mathbf{y}$, $k_{PSR}(p_s, q)$ as the entries of $\mathbf{k}_2$ and division by $\sigma_g W(p_s)$ playing the role of multiplication by $\mathbf{K}_3^{-1}$. 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 *semi*covariance"). To circumvent this, we define the *Stochastic PSR covariance* to be its symmetrized version

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

which analagously defines a *Stochastic PSR vector field*

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

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).

Our critical observation is that the interpolation in Eq. (12) can be interpreted as the mean of a supervised learning task under the

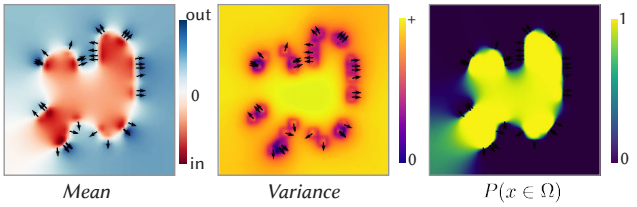


Fig. 10. By adding a variance, we can consistently compute the probability of any point in space being in the sampled object.

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

$$\vec{V}(q) \,|\, \mathcal{S} \sim \mathcal{N}(\mathbf{k}_2^\top \mathbf{K}_3^{-1} \vec{N}_s, k_1 - \mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{k}_2) \tag{13}$$

where

$$k_1 = k(q,q)\,, \quad \mathbf{k}_2 = (k(q,p_s))_{s\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}\,, \tag{14}$$

$$\mathbf{K}_3 = (k(p_s,p_{s'}))_{s,s'\in\mathcal{S}} \in \mathbb{R}^{|S|\times|S|}\,. \tag{15}$$

The main computational cost in computing the mean and variance of the conditional distribution is the inversion of $\mathbf{K}_3$, 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 $\mathbf{K}_3$ by $\sigma_g \mathbf{I}$. 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*

$$\mathbf{D} := \mathrm{diag}(\sigma_g w) \approx \mathbf{K}_3 \tag{16}$$

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 $\sigma_n \mathbf{I}$ if the sampled point cloud is assumed to contain noise with variance σ_n in the normal vector.

Then, the conditional distribution becomes

$$\vec{V}(q) \,|\, \mathcal{S} \sim \mathcal{N}(\mathbf{k}_2^\top \mathbf{D}^{-1} \vec{N}_s, k_1 - \mathbf{k}_2^\top \mathbf{D}^{-1} \mathbf{k}_2) \tag{17}$$

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

$$\vec{V}(q) \,|\, \mathcal{S} \sim \mathcal{N}(\vec{V}_{SPSR}(q), k_1 - \mathbf{k}_2^\top \mathbf{D}^{-1} \mathbf{k}_2)\,. \tag{18}$$

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).

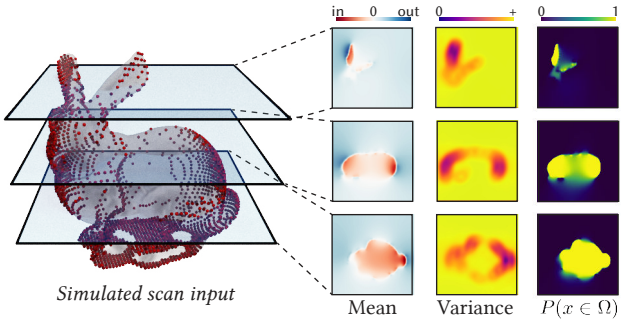


Fig. 11. Our extension of PSR provides a mean and variance, which can be used to compute probabilities.

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, transfering 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

$$\vec{V}(o_1), \ldots, \vec{V}(o_{|O|}) \,|\, \mathcal{S} \sim \mathcal{N}(\mathbf{V}_{SPSR}, \mathbf{K}_1 - \mathbf{K}_2^\top \mathbf{D}^{-1} \mathbf{K}_2)\,, \tag{19}$$

where

$$\mathbf{K}_1 = (k(o,o')) \in \mathbb{R}^{|O|\times|O|}\,, \quad \mathbf{K}_2 = (k(o,p_s)) \in \mathbb{R}^{|O|\times|\mathcal{S}|} \tag{20}$$

and

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

For simplicity, denote $\mathbf{K_V} = \mathbf{K}_1 - \mathbf{K}_2^\top \mathbf{D}^{-1} \mathbf{K}_2$, and let $\mathbf{v}$ be the concatenated vector of $\vec{V}$ values as in the previous section. Then, Eq. (19) becomes

$$\mathbf{v} \,|\, \mathcal{S} \sim \mathcal{N}(\mathbf{V}_{SPSR}, \mathbf{K_V})\,. \tag{22}$$

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

$$\mathbf{Zv} \,|\, \mathcal{S} \sim \mathcal{N}(\mathbf{ZV}_{SPSR}, \mathbf{ZK_V Z}^\top)\,. \tag{23}$$

and thus, since the vector $\mathbf{f}$ of evaluations of f satisfies $\mathbf{Lf} = \mathbf{Zv}$,

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

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

$$\mathbf{f} \,|\, \mathcal{S} \sim \mathcal{N}\left(\mathbf{f}_{SPSR}, \mathbf{K_f}\right)\,. \tag{25}$$

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

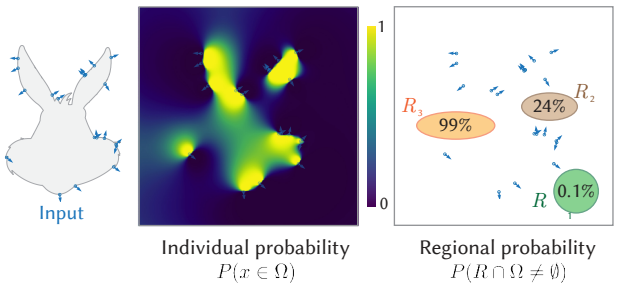


Fig. 12. Our algorithm not only allows us to compute individual spatial probabilities (center), but also joint regional probabilities (right).

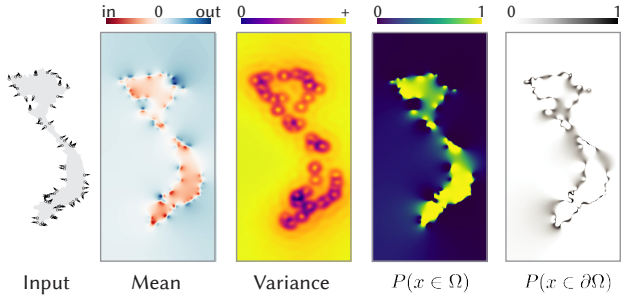


Fig. 13. Our computed mean and variance (center left) completely determine the implicit function's distribution, allowing us to respond to statistical queries like the likelihood of a point being in the volume defined by the point cloud (center right) or on its surface (right).

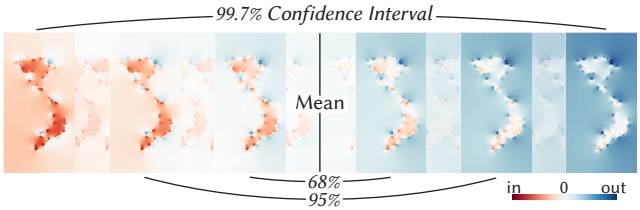


Fig. 14. The *68-95-99.7 rule* from normal distributions lets us compute confidence intervals for the implicit function values. Regions with the same color (red or blue) at both extremes of an interval can be said to be inside or outside with at least the interval's confidence.

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)

$$p(o_i \in \Omega) = p(\mathbf{f}_i \leq 0) = CDF_{\mathbf{f}_{SPSR,i},\sigma_i^2}(0) \tag{26}$$

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

$$p(o_i \in \Omega) = p(\mathbf{f}_i = 0) = PDF_{\mathbf{f}_{SPSR,i},\sigma_i^2}(0) \tag{27}$$

where CDF_{μ,σ^2} and PDF_{μ,σ^2} are the cummulative 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

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

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 $\mathbf{K_f}$ quantify the correlation between the values of f at different points in 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 $R_s = \{r_1, \ldots, r_s\} \subset R$. If $\mathbf{W}$ is the linear interpolation matrix from O to R_s, then distribution arithmetic says

$$\mathbf{f}(r_1, \ldots, r_s)\,|\mathcal{S} \sim \mathcal{N}(\mathbf{W}\mathbf{f}_{SPSR}, \mathbf{W}^\top \mathbf{K_f} \mathbf{W})\,. \tag{29}$$

The collision probability, i.e., the probablity that any r_i is contained in the shape, is opposite to the joint probability that all r_i are outside the shape, $p(\mathbf{f}(r_1) > 0, \ldots, \mathbf{f}(r_s) > 0))$. This quantity is also the same as $p(-\mathbf{f}(r_1) < 0, \ldots, -\mathbf{f}(r_s) < 0))$, known as the *multivariate Gaussian cummulative distribution* of $-\mathbf{f}$, 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

$$\mathbf{K_f} = \mathbf{L}^{-1}\mathbf{Z}\mathbf{K_V}\mathbf{Z}^\top(\mathbf{L}^\top)^{-1}, \tag{30}$$

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

To circunvent 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

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

with associated eigenvalues

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

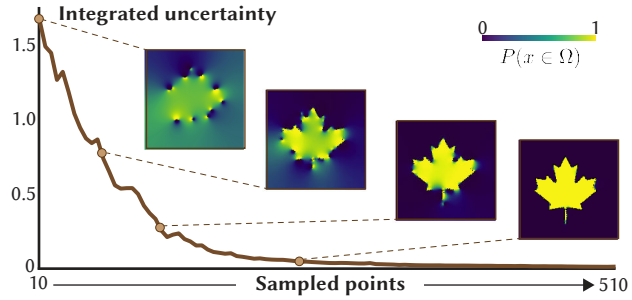


Fig. 15. Our statistical formalism means we can define quantities like the integrated uncertainty, which converges to zero as the probability is collapsed to zero and one. This measure can serve as a scanning stopping criterion.

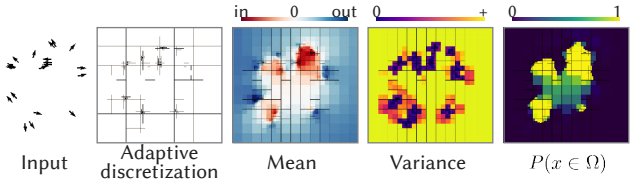


Fig. 16. Our contribution is orthogonal to the choice of discretization, as we show by answering statistical queries on a graded quadtree.

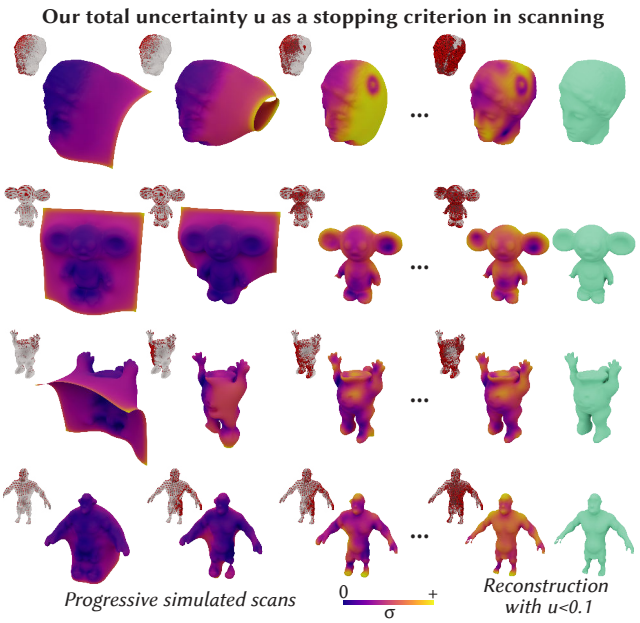


Fig. 17. Like PSR, we can extract isosurfaces of the mean of our Stochastic PSR. Unlike PSR, our statistical formalism provides a reliable stopping criterion by thresholding integrated uncertainty u. Meshes colored by variance.

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

$$\mathbf{K_f} \approx \mathbf{E}\mathbf{D}_e^{-1}\mathbf{E}^\top \left(\mathbf{Z}\mathbf{K_V}\mathbf{Z}^\top\right) \mathbf{E}\mathbf{D}_e^{-1}\mathbf{E}^\top . \tag{33}$$

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 $\mathbf{E}\mathbf{C}\mathbf{E}^\top$, where $\mathbf{C} \in \mathbb{R}^{k \times k}$ 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 $\mathbf{K_f}$. As seen in Section 4.1, most statistical queries require only the diagonal $\sigma_\mathbf{f}^2$, which we can compute directly and efficiently as $(\mathbf{EC}) \cdot \mathbf{E}$, where $\cdot$ denotes row-wise dot product. In the case of joint probability queries that require off-diagonal knowledge of $\mathbf{K_f}$, we first construct the selection matrices $\mathbf{S}, \mathbf{S}^\top$ that extract the covariance rows and columns relevant to the specific application, and directly compute $\mathbf{E'}\mathbf{C}\mathbf{E'}^\top$, where $\mathbf{E'} = \mathbf{S}\mathbf{E}$. Thus, we eventually avoid ever computing (or storing) the full $\mathbf{K_f} \in \mathbb{R}^{|O| \times |O|}$.

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.

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 $\mathbf{f}_{SPSR}$ 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

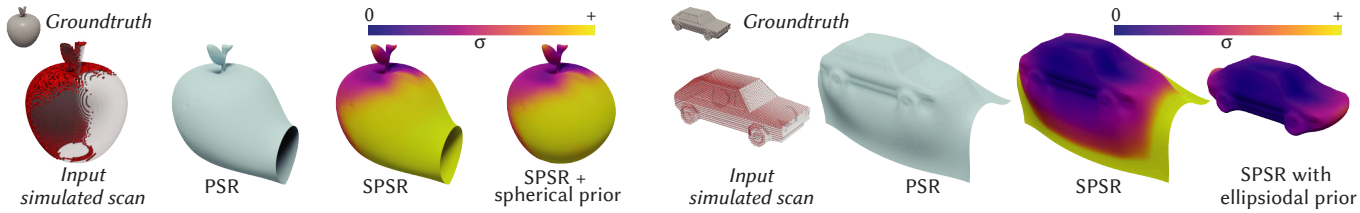


Fig. 18. Taking inspiration from Figs. 6 and 7 in [Martens et al. 2016], we show the result of reconstructing an apple and a car from a partial scan. PSR fails to produce even a closed surface, as does our vanilla SPSR, albeit also providing variance information signaling the less confident regions. 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 transfered 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

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

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 $\mathbf{c}$ 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.

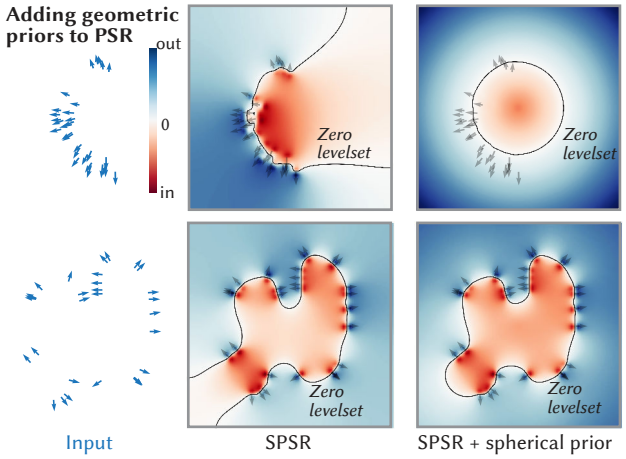


Fig. 19. Adding a spherical prior to our Stochastic PSR reconstruction helps us recover spherical objects and also avoid open surfaces in partial scans.

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 $\mathbf{L}$ and $\mathbf{Z}$ 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_o 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 $\mathbf{K}_\mathbf{Y}$ into the reduced space), covering 90% of our runtime (around two minutes in all 3D examples).

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 cummulative 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 it to compute statistical quantities like collision chances (bold), unlike traditional PSR (grey).

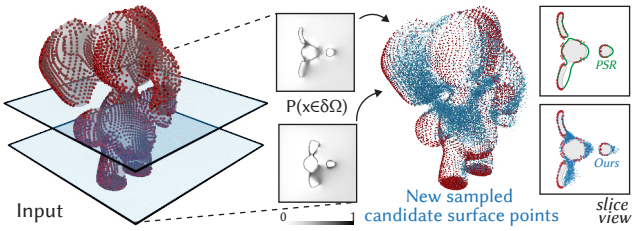


Fig. 21. We can use our computed surface likelihood to repair the input point cloud, naturally drawing points from less certain regions.

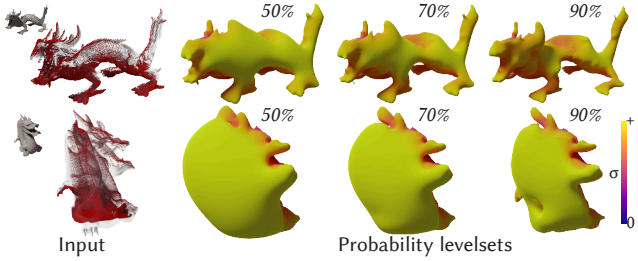


Fig. 22. We can threshold our computed probability $p(x \in \Omega)$ to obtain meshes that can be used for, e.g., collision detection.

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 maneuvre 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 maneuvre 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 isolevel 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 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),

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

We repeat this process ten times for each potential camera position, to account for the statistical variabilty 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.

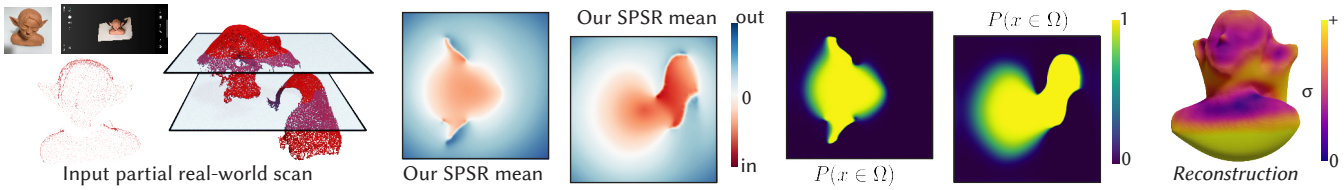


Fig. 23. Our SPSR on real-world scan data. The geometry of our reconstructed surface is the same as PSR, but we provide variances and volumetric quantities.

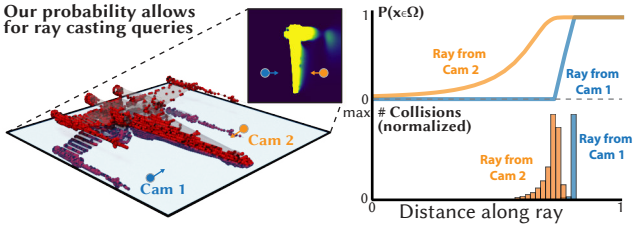


Fig. 24. Our probability can be interpreted as a density for ray casting applications that have no analogue in the traditional PSR formulation.

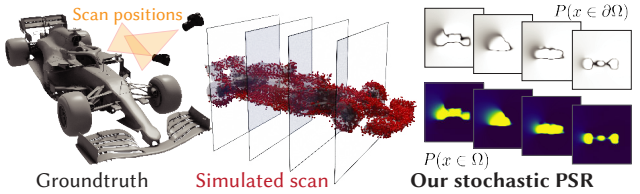


Fig. 25. Many of our results follow a similar pipeline: we use a groundtruth object (left) and simulate scanning it from different directions to obtain an oriented point cloud (middle), which we then pass as input to our Stochastic PSR algorithm (right) to compute the desired statistical quantity.

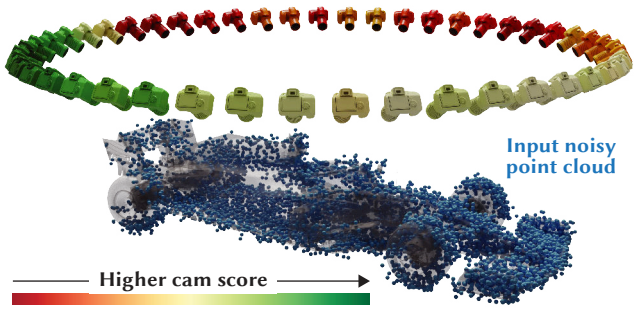


Fig. 26. Interpreting our probability as a density and simulating rays from different cameras, we can score them according to the simulated change in total uncertainty to aid next view planning.

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; Metzer 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.

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 Benchekroun 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, Yiying Tong, and Mathieu Desbrun. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Eurographics SGP*.

Beatriz Trinchão Andrade, Caroline Mazetto Mendes, Jurandir de Oliveira Santos Jr, Olga Regina Pereira Bellon, and Luciano Silva. 2012. 3D preserving XVIII century barroque 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 IW 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 Metzer, 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

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

and our Stochastic PSR covariance

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

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

$$k^\star(x,y) = \sigma_g F_y(x) \tag{38}$$

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

$$|F_o(y) - F_y(x)| = |F_y(o) - F_y(x)| \leq \lambda \|o - x\| \tag{39}$$

Since trilinear interpolation weights sum up to one, this means

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

Symmetrically, we get

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

which, by definition of k_{SPSR}, means

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

B JUSTIFICATION FOR COVARIANCE LUMPING

Consider a simple Gaussian Process $\mathcal{A} = \{A(x)\}_{x \in D}$ with zero mean and covariance function $k : D \times D \to \mathbb{R}$, 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

$$\mathbf{K}_3 = \begin{pmatrix} k_{11} & k_{12} \\ k_{12} & k_{11} \end{pmatrix} \quad , \quad \mathbf{k}_2 = \begin{pmatrix} k_{13} \\ k_{23} \end{pmatrix} \tag{43}$$

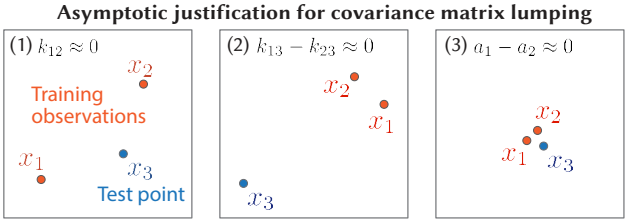


Fig. 27. Cases where our lumping has correct asymptotics: (1) when training data are far from one another, (2) when test points are far from training data and (3) when training data are so close that their features are similar.

Traditionally, one uses these matrices to compute the GP posterior mean $\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{a}$ and covariance $\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{k}_2$. Let us write both of these as $\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{c}$ for some generic feature vector $\mathbf{c}$. Then,

$$\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{c} = \frac{1}{k_{11}^2 - k_{12}^2}((k_{11}k_{13} - k_{12}k_{23})c_1 + (k_{11}k_{23} - k_{12}k_{13})c_2)$$

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,

$$\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{c} = \frac{k_{13}c_1 + k_{23}c_2}{k_{11} + k_{12}} + \frac{k_{12}(k_{23} - k_{13})(c_1 - c_2)}{k_{11}^2 - k_{12}^2}, \tag{44}$$

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

$$\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{c} = \mathbf{k}_2^\top \mathbf{D}^{-1} \mathbf{c} + \frac{k_{12}(k_{23} - k_{13})(c_1 - c_2)}{k_{11}^2 - k_{12}^2}, \tag{45}$$

where $\mathbf{D}$ is our lumped covariance matrix. Therefore,

$$\left|\mathbf{k}_2^\top \mathbf{K}_3^{-1} \mathbf{c} - \mathbf{k}_2^\top \mathbf{D}^{-1} \mathbf{c}\right| \leq \frac{k_{12}}{k_{11}^2 - k_{12}^2} |k_{23} - k_{13}| \, |c_1 - c_2| \tag{46}$$

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 ($k_{12} \to 0$), when the test samples are far enough from the training samples ($k_{23} - k_{13} \to 0$), and when the training samples are close enough that the training features are close ($c_1 - c_2 \to 0$). 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.

Reach For the Spheres: Tangency-Aware Surface Reconstruction of SDFs

Silvia Sellán
University of Toronto
Canada
sgsellan@cs.toronto.edu

Christopher Batty
University of Waterloo
Canada
christopher.batty@uwaterloo.ca

Oded Stein
University of Southern California
United States of America
ostein@usc.edu

Figure 1: Reconstructing a mesh from the discrete signed distance field (SDF) of a koala (source, *rightmost*). By using global information from all sample points at once, our method recovers the shape even in low resolutions where methods like Marching Cubes [Lorensen and Cline 1987] and Neural Dual Contouring (NDCx) [Chen et al. 2022a] produce very coarse shapes (*left trio*), and it recovers surface detail at higher resolutions that Marching Cubes and NDCx miss (*middle trio*). Our method is purely geometric, and does not require any training or storing of weights (unlike NDCx).

ABSTRACT

Signed distance fields (SDFs) are a widely used implicit surface representation, with broad applications in computer graphics, computer vision, and applied mathematics. To reconstruct an explicit triangle mesh surface corresponding to an SDF, traditional isosurfacing methods, such as Marching Cubes and and its variants, are typically used. However, these methods overlook fundamental properties of SDFs, resulting in reconstructions that exhibit severe oversmoothing and feature loss. To address this shortcoming, we propose a novel method based on a key insight: each SDF sample corresponds to a spherical region that must lie fully inside or outside the surface, depending on its sign, and that must be tangent to the surface at some point. Leveraging this understanding, we formulate an energy that gauges the degree of violation of tangency constraints by a proposed surface. We then employ a gradient flow that minimizes our energy, starting from an initial triangle mesh that encapsulates the surface. This algorithm yields superior reconstructions to previous methods, even with sparsely sampled SDFs. Our approach provides a more nuanced understanding of SDFs and offers significant improvements in surface reconstruction.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; **Mesh models**; *Mesh geometry models.*

KEYWORDS

surface reconstruction, signed distance function, geometric flow

1 INTRODUCTION

Signed distance fields (SDFs) are a classical implicit surface representation that finds diverse applications in computer graphics, computer vision, and applied mathematics, among other domains [Frisken et al. 2000; Jones et al. 2006; Sethian 1999]. A continuous SDF is a scalar function $\phi(\mathbf{x})$ that, given a query point $\mathbf{x}$ in $\mathbb{R}^n$, returns the Euclidean distance to the closest point on the surface it represents, augmented with a sign indicating whether the point is on the interior or exterior. A discrete SDF samples this function at a finite set of points in space, such as a grid, octree, or point cloud. The task we consider is the reconstruction of an explicit triangle mesh corresponding to the zero isosurface of such a discrete SDF.

Perhaps the most familiar such isosurfacing approach is Marching Cubes [Lorensen and Cline 1987] and its variants. They use sign changes between adjacent SDF samples (e.g., along grid edges) to approximately locate the zero isosurface and apply per-cell templates and linear interpolation of the function values to fill in local

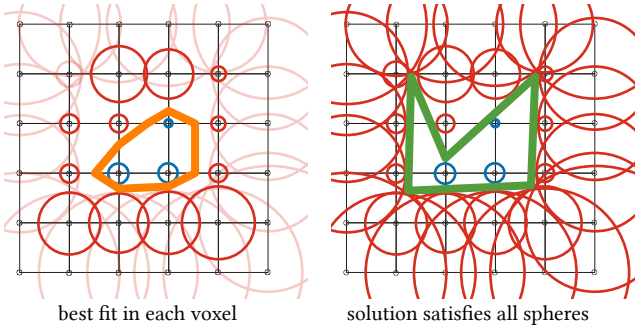


Figure 2: Reconstructing an SDF per-voxel, by finding a best fit line segment in each voxel containing both positive (red) and negative (blue) values, discards much of the available global information. Our main insight is that a solution adhering to the constraints of *all* spheres yields better results.

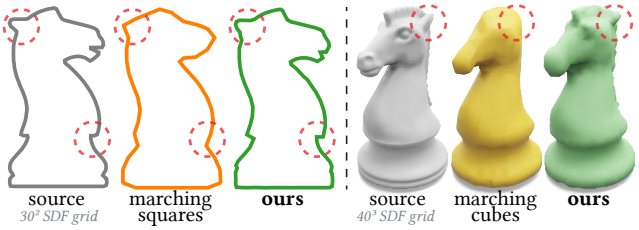


Figure 3: Using global information (not just per-voxel data), we reconstruct sharp features even at low resolutions.

patches of the surface triangulation. While effective and appropriate for *general* implicit surface data, these schemes ignore the unique and fundamental properties of SDFs to their detriment. Indeed, mesh reconstructions of SDF data invariably exhibit severe oversmoothing and feature loss (see Fig. 1). Approaches like dual contouring [Ju et al. 2002; Kobbelt et al. 2001] can better recover sharp features by additionally relying on surface normals. However, discrete SDFs lack the necessary gradient information and finite difference estimates give disappointing results. Is there any hope of achieving better reconstructions from the SDF data alone?

Neural Marching Cubes [Chen and Zhang 2021] and Neural Dual Contouring [Chen et al. 2022a] have recently answered this question in the affirmative: they demonstrate better per-cell reconstructions by training on a large dataset of SDFs and using wider (7^3) stencils of SDF grid points. The quality of these results suggests that there exists some additional information implicit in the SDF data. Our objective is therefore to explicitly identify and directly exploit this overlooked geometric information, without recourse to learning approaches, and thereby achieve superior reconstructions.

The key insight underpinning our method is that (see Fig. 2) *each SDF sample $\phi(\mathbf{x})$ corresponds to a spherical region*, centered at $\mathbf{x}$ and with radius equal to $|\phi(\mathbf{x})|$. By definition, the true surface represented by the discrete SDF must be tangent to every sphere at least once while strictly containing every sphere with negative value and excluding every positive value one. Through these constraints,

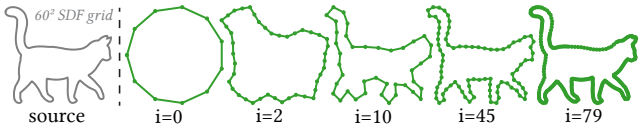


Figure 4: Our 2D flow at different iteration counts on a cat, sampled from the source mesh on the *left*.

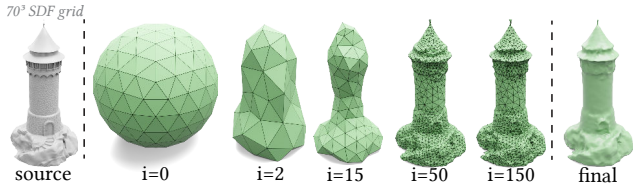


Figure 5: Our 3D flow at different iteration counts on a tower, sampled from the source mesh (*left*). The final version (*right*) has been Loop subdivided.

the SDF samples contain significantly more information about the surface than samples from a generic implicit representation would.

To fully exploit this information, we first formulate an energy that measures the degree to which a proposed surface violates the tangency constraints of the input SDF samples. We propose an algorithm that starts from a triangle mesh enclosing the surface, then "shrinkwraps" the underlying surface via a gradient flow that minimizes our energy, interleaved with remeshing to ensure mesh quality. The fidelity of the resulting reconstructions surpasses that of prior methods, especially for sparsely sampled SDFs. Additionally, since our method has no intrinsic dependence on a grid, it is amenable to unstructured point cloud SDFs, and even incorporating new samples, where available, to improve the reconstruction.

2 RELATED WORK

2.1 Signed Distance Fields

SDFs have been used in countless applications spanning the computational sciences, so we highlight only a representative sample. In computer graphics they have been applied to liquid surface tracking [Foster and Fedkiw 2001], geometric modeling [Museth et al. 2002], collision detection [Fuhrmann et al. 2003], and ray (sphere) tracing [Hart 1996]. In traditional computer vision, uses have included image / volume segmentation [Chan and Vese 1999] and surface reconstruction from multiview data [Faugeras and Keriven 1998] or point clouds [Zhao et al. 2001]. In computational physics, SDFs have been applied (via the level set method) to model combustion, crystal growth, and fluid dynamics [Osher and Fedkiw 2003; Sethian 1999]. SDFs have also been applied to manufacturing [Brunton and Rmaileh 2021] and robot path planning [Liu et al. 2022].

SDFs have recently seen renewed interest in the context of geometric deep learning. In particular, the DeepSDF approach [Park et al. 2019] replaces the discrete SDF with a learned continuous SDF of a shape or a space of shapes. This concept represents a subset of general neural implicit surfaces and of neural fields even more broadly [Xie et al. 2022]. Differentiable rendering with SDFs has been investigated to solve inverse problems [Bangaru et al. 2022;

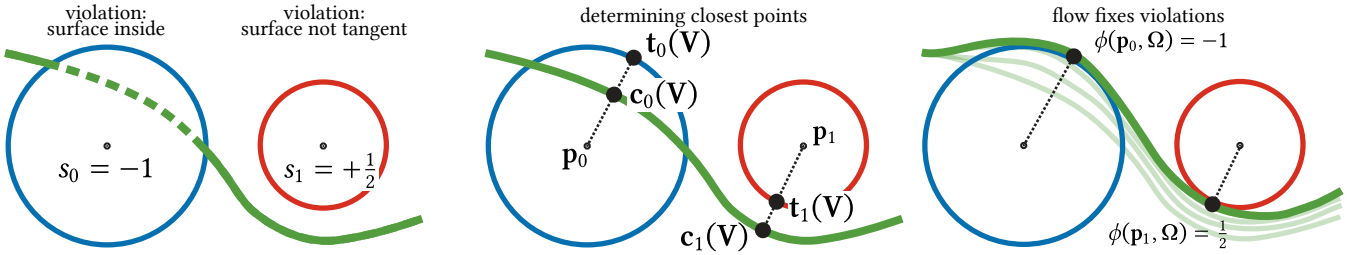


Figure 6: A green surface the sphere constraints from two SDF samples *(left)*. The method identifies the closest point on the surface and the closest point on the sphere for each sample point $\mathbf{p}_i$ that makes the sphere correctly lie inside or outside the surface, as prescribed by s_i *(middle)*. Our flow fixes all violations until the constraints $\phi(\mathbf{p}_i, \mathcal{M}) = s_i$ are fulfilled *(right)*.

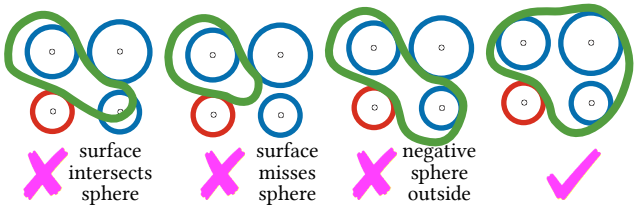


Figure 7: Three surfaces violating the SDF constraints in different ways, and a surface that satisfies them *(left to right)*.

Vicini et al. 2022]. Since neural implicits often fail to exactly retain the signed distance property, Sharp and Jacobson [2022] propose techniques for robust geometric queries in this setting.

Despite widespread adoption of SDFs, prior techniques often view SDFs as a convenient and canonical but *general* implicit surface function. They may exploit the distance property in some respects, but none that we are aware of consider the additional subgrid information implied by our tangent-spheres interpretation (mentioned by Batty [2011]; Kobbelt et al. [2001]). Instead, the location of the zero isosurface is assumed to be that of the linear (or occasionally polynomial) interpolant of the SDF samples. However, away from the data points, such interpolated fields are not true SDFs.

2.2 Isosurfacing Approaches

The task of generating an explicit mesh corresponding to a given implicit surface is variously referred to as isosurfacing, polygonization, surface reconstruction, or simply meshing. There is an extensive body of literature on the topic, so we refer the reader to the survey by De Araújo et al. [2015]. There exist three broad categories of isosurfacing schemes: first, spatial subdivision schemes like Marching Cubes; second, advancing front methods, which start at a point and incrementally attach new triangles as they propagate across the surface until it is covered [Hilton et al. 1996; Sharf et al. 2006]; and third, shrinkwrap or inflation methods, which start with an initial closed surface and gradually grow it inwards or outwards to conform to the desired isosurface [Hanocka et al. 2020; Stander and Hart 1997; Van Overveld and Wyvill 2004]. Our method falls into the last category, which is relatively under-explored compared to the others. The work of Bukenberger and Lensch [2021] employs evolving meshes with periodic remeshing that conform to target SDFs, like our approach. They require the SDF to be resampled at every iteration of their method, while our method only requires the SDF to be evaluated on a finite set of evaluation points at the beginning (although our method can support resampling as well, see Fig. 14), and they do not use all SDF spheres' global information.

Isosurfacing methods are often applied to SDFs, but seldom exploit the signed distance property. Indirectly, Neural Marching Cubes [Chen and Zhang 2021] and Neural Dual Contouring [Chen et al. 2022a] represent two key exceptions. Their SDF dependence is not explicit in their algorithms, but implicitly encoded into their neural networks when trained on exact SDFs to achieve improved results compared to prior non-neural schemes. Our approach avoids any reliance on deep learning, instead making explicit use of fundamental geometric properties of SDFs.

Recently, Deep Marching Cubes (DMC) [Liao et al. 2018], MeshSDF [Remelli et al. 2020], and FlexiCubes [Shen et al. 2023] were developed to offer *differentiable* isosurfacing procedures. Their goal is often to incorporate isosurfacing into end-to-end deep learning pipelines for applications like shape completion, shape optimization, or single-view reconstruction. By contrast, we focus on achieving the highest quality of reconstruction of discrete SDFs. Our SDF energy has connections to the losses used in such work.

2.3 Mesh Optimization

Our algorithm uses gradient flow to optimize an energy with respect to the vertex positions of a triangle mesh, with the aim of finding a valid, tangency-aware surface reconstruction. Variational techniques in this style are ubiquitous in geometry processing applications, such as mean curvature flow and surface fairing [Desbrun et al. 1999; Kazhdan et al. 2012], mesh quality improvement [Alliez et al. 2005], Willmore flow [Crane et al. 2013], constructing coarse cages [Sacht et al. 2015], developability [Stein et al. 2018], and morphological operations [Sellán et al. 2022]. To maintain and improve mesh quality during our flow, we employ the local remeshing scheme of Botsch and Kobbelt [2004].

Our flow displaces the vertices of a mesh such that the mesh is tangent to a set of spheres centered at the SDF sample points. In a way, this can be seen as solving a reverse formulation of the medial axis computation problem. In it, one searches for maximally contained spheres tangent to a given surface, often through a combination of greedy decompositions and progressive simplifications [Li et al. 2016; Ma et al. 2012; Rebain et al. 2019].

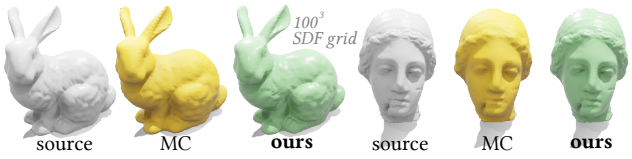


Figure 8: While our algorithm shines at low and medium resolutions, it also recovers high-frequency detail Marching Cubes misses at higher resolutions.

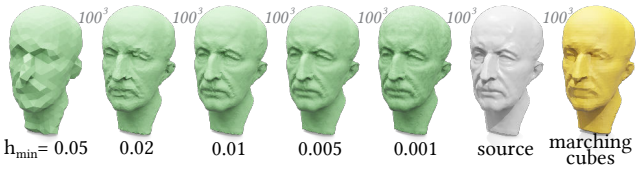


Figure 9: A critical parameter in our method is the maximum resolution, encoded in the edge-length $h_{\min}$, which balances the under- or over-constrained nature of our optimization.

3 METHOD

Let us assume we are given access to the values $s_1, \dots, s_n \in \mathbb{R}$ of the Signed Distance Function ϕ for an unknown surface Ω sampled at n points in space $\mathbf{p}_1, \dots, \mathbf{p}_n \in \mathbb{R}^3$, $s_i = \phi(\mathbf{p}_i, \Omega)$. Our task will be to reconstruct a *valid* surface Ω; i.e., one that is consistent with the SDF observations. This can be expressed as the constraints

$$\phi(\mathbf{p}_i, \Omega) = s_i\,, \quad \forall i \in \{1, \dots, n\}\,. \tag{1}$$

Intuitively, one may visualize this condition by drawing a sphere S_i of radius $|s_i|$ around each $\mathbf{p}_i$ and requiring that the surface Ω be tangent to all of them with the correct orientation (see Fig. 7).

We begin with a simple idea: turn (1) into an energy minimization problem over the space of surfaces. To this end, we define the *SDF energy* of a surface to be the squared difference between s_i and the SDF value of the surface at $\mathbf{p}_i$:

$$\mathcal{E}_\phi(\Omega) = \frac{1}{2}\sum_{i=1}^{n} \left(\phi(\mathbf{p}_i, \Omega) - s_i\right)^2\,. \tag{2}$$

Exploring the entire space of surfaces Ω to find one that minimizes the above energy is intractable. Instead, we propose to start from an initial surface Ω^0 and follow the gradient flow of the SDF energy

$$\frac{\partial \Omega}{\partial t} = -\nabla\, \mathcal{E}_\phi(\Omega)\,. \tag{3}$$

We refer to this as our *sphere reaching* flow, since it will encourage Ω^t to touch every S_i at least once, while strictly containing all negative S_i and strictly excluding all positive S_i (see Fig. 6).

4 DISCRETIZATION

We discretize the time dimension of our flow using an implicit scheme to obtain the sequence $\Omega^0, \Omega^1, \dots, \Omega^T$ of surfaces such that

$$\Omega^t = \Omega^{t-1} - \tau \nabla\, \mathcal{E}_\phi\left(\Omega^t\right) \tag{4}$$

for some small time step τ. Equivalently, given a surface Ω^{t-1}, we will aim to find a new surface Ω^t that minimizes the energy

$$\Omega^t = \operatorname*{argmin}_{\Omega} \frac{1}{2\tau}\left\|\Omega - \Omega^{t-1}\right\|_2^2 + \mathcal{E}_\phi(\Omega)\,. \tag{5}$$

In order to sequentially solve this minimization problem for $t = 0, \dots, T$, we will need to discretize the space of surfaces Ω. We will do so by representing each surface Ω^t as a triangle mesh $\mathbf{\Omega^t}$ with vertices $\mathbf{V}^t$ and faces $\mathbf{F}^t$. (5) can then be written as

$$\Omega^t = \operatorname*{argmin}_{\Omega} \left(\frac{1}{2\tau}\left\|\mathbf{V} - \mathbf{V}^{t-1}\right\|_{\mathbf{M}}^2 + \mathcal{E}_\phi(\Omega)\right), \tag{6}$$

where $\|\cdot\|_{\mathbf{M}}$ is the mass-matrix-integrated norm $\|\cdot\|_{\mathbf{M}}^2 = \cdot^\top \mathbf{M} \cdot$.

Unfortunately, $\mathcal{E}_\phi(\Omega)$ is not convex or even continuously differentiable, which makes it difficult to minimize. To circumvent this, we first define $\mathbf{c}_i(\Omega)$ as the specific closest point on the surface Ω to $\mathbf{p}_i$,[1] which we can write as $\mathbf{a}_i(\Omega)\mathbf{V}$ for some sparse vector of barycentric coordinates $\mathbf{a}_i(\Omega)$. Then $\mathcal{E}_\phi(\Omega)$ becomes

$$\mathcal{E}_\phi(\Omega) = \frac{1}{2}\sum_{i=1}^{n} \left(\phi(\mathbf{p}_i, \mathbf{c}_i(\Omega)) - s_i\right)^2 \tag{7}$$

where we have slightly abused notation to let $\phi(\mathbf{p}_i, \mathbf{c}_i(\Omega))$ return the distance $\|\mathbf{p}_i - \mathbf{c}_i(\Omega)\|$ with the sign of $\phi(\mathbf{p}_i, \Omega)$. This modified energy penalizes distances between each closest point and the corresponding sphere's surface. Refer to Fig. 6 for the geometric picture.

We next define $\mathbf{t}_i(\Omega)$ as the projection of $\mathbf{p}_i$, along the line through $\mathbf{p}_i(\Omega)$ and $\mathbf{c}_i(\Omega)$, onto the signed distance sphere S_i,

$$\mathbf{t}_i(\Omega) = \mathbf{p}_i + \sigma_i |s_i| \frac{\mathbf{c}_i(\Omega) - \mathbf{p}_i}{\|\mathbf{c}_i(\Omega) - \mathbf{p}_i\|}\,, \tag{8}$$

where σ_i depends on the orientation of the surface Ω at $\mathbf{c}_i(\Omega)$:

- If $\mathbf{p}_i$ is inside/outside Ω and the sign of s_i is negative/positive, then $\sigma_i = 1$.
- If $\mathbf{p}_i$ is inside/outside Ω and the sign of s_i is positive/negative, then $\sigma_i = -1$.

That way, if the surface were translated such that $\mathbf{c}_i(\Omega)$ coincided with $\mathbf{t}_i$, the SDF would be satisfied at $\mathbf{p}_i$. We use the mesh element's normal vector at $\mathbf{c}_i(\Omega)$ to distinguish between inside and outside.[2]

As $\mathbf{t}_i(\Omega)$ and $\mathbf{c}_i(\Omega)$ will be equal for a valid solution, we approximate $(\phi(\mathbf{p}_i, \mathbf{c}_i(\Omega)) - s_i)$ by $\|\mathbf{c}_i(\Omega) - \mathbf{t}_i(\Omega))\|$. Thus, (7) becomes

$$\frac{1}{2}\sum_{i=1}^{n} \|\mathbf{c}_i(\Omega) - \mathbf{t}_i(\Omega))\|^2 = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{a}_i(\Omega)\mathbf{V} - \mathbf{t}_i(\Omega)\|^2\,, \tag{9}$$

which we further simplify by fixing $\mathbf{t}_i(\Omega)$ to $\mathbf{t}_i\left(\mathbf{\Omega^{t-1}}\right)$.

Concatenating $\mathbf{a}_i$ and $\mathbf{t}_i$ into matrices $\mathbf{A}$ and $\mathbf{S}$, this becomes

$$\frac{1}{2}\|\mathbf{AV} - \mathbf{S}\|_F^2\,, \tag{10}$$

which we can now incorporate into (6):

$$\mathbf{V}^t = \operatorname*{argmin}_{\mathbf{V}} \frac{1}{2\tau}\left\|\mathbf{V} - \mathbf{V}^{t-1}\right\|_{\mathbf{M}}^2 + \frac{1}{2}\|\mathbf{AV} - \mathbf{S}\|_F^2\,. \tag{11}$$

[1] We use an AABB tree to speed up computation of the closest point, and we compute the query for every p_i at once.

[2] In practice, we do not distinguish between inside and outside spheres if $\mathbf{c}_i(\Omega)$ is on the boundary of a mesh element, since normal information is not reliable there — $\mathbf{t}_i(\Omega)$ is simply the closest point on the sphere.

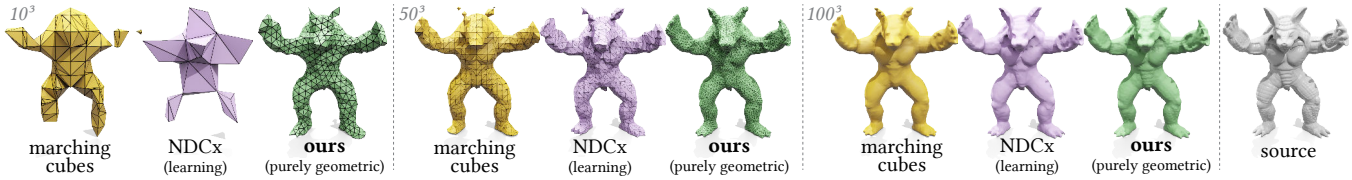


Figure 10: With no training and no network weight storage, our method consistently outperforms MC and outperforms or matches Neural DC across resolutions.

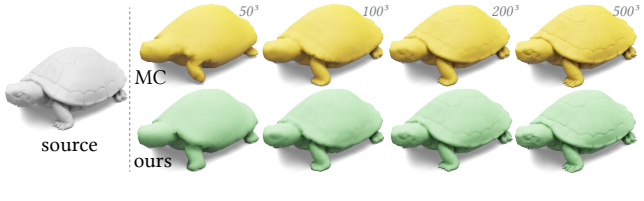


Figure 11: Our flow can be used on sparse (recovering more detail than MC) and very dense (matching MC) SDF grids.

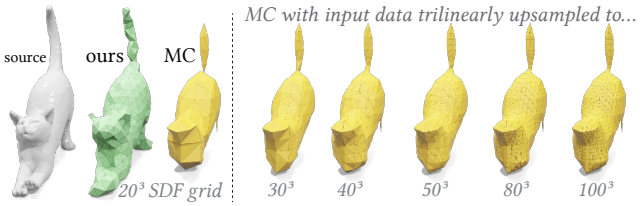


Figure 12: Our algorithm's full exploitation of all the SDF input data means its improved performance against Marching Cubes is preserved even if one artificially upsamples the SDF input before using MC.

This quadratic optimization problem on the vertex positions $\mathbf{V}$ can be solved via the linear system

$$\mathbf{Q}\mathbf{V}^t = \mathbf{B} \tag{12}$$

where

$$\mathbf{Q} = \mathbf{M} + \tau\mathbf{A}^\top\mathbf{A}, \qquad \mathbf{B} = \mathbf{M}\mathbf{V}^{t-1} + \tau\mathbf{A}^\top\mathbf{S}. \tag{13}$$

The matrices $\mathbf{A}$, $\mathbf{M}$, and $\mathbf{Q}$ are sparse, thus the linear system can be efficiently solved using, e.g., Cholesky decomposition. A simplified step of this flow can be seen on the right of Fig. 6.

Choosing step size τ. The formulation in (12) produces a flow that, for a small enough τ, will reduce the SDF energy each iteration. However, choosing τ too small can be inefficient, while τ too large will violate the linearization assumptions made in our discretization and cause flow instabilities. We use a heuristic inspired by Armijo's condition [Nocedal and Wright 2006] to choose the optimal step size, $\tau = \rho\,\mathrm{clamp}(\tau_*, \tau_{\min}, \tau_{\max})$, where $\rho = \frac{1}{n}$,

$$\tau_* = -\frac{\rho(\mathbf{A}-\mathbf{S})\cdot(\mathbf{A}\mathbf{P}) + 0.01\|\mathbf{P}\|^2}{\rho\|\mathbf{A}\mathbf{P}\|^2}, \quad \mathbf{P} = -\rho\mathbf{A}^\top(\mathbf{A}-\mathbf{S}), \tag{14}$$

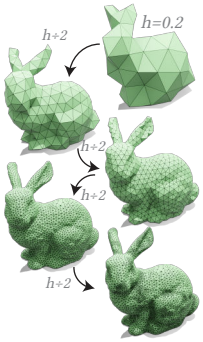
and, by default, $\tau_{\min} = 10^{-6}$, $\tau_{\max} = 50$.

4.1 Mesh resolution and quality

As mesh vertices $\mathbf{V}$ move during our flow, mesh quality rapidly degrades, producing degeneracies, flipped and thin triangles, and self-intersections. We solve this common problem of geometric flows by remeshing with the algorithm of Botsch and Kobbelt [2004], which uses a sequence of local improvement operations. After each flow iteration, we apply a single remeshing iteration using a given target edge-length h (more iterations may be used if needed). We implement this operation in an output-sensitive manner, analogous to the approach of Sellán et al. [2022], by remeshing exclusively the regions of the surface that are the closest point on Ω to any of the $\mathbf{p}_i$ and violate the SDF value s_i by more than a tolerance ε (by default, $5 \cdot 10^{-3}$ for 2D and 10^{-2} for 3D).

Both our flow and our remeshing operations preserve the intrinsic shape's topology. This restriction helps us avoid some of the most catastrophic failures of existing methods like Marching Cubes, which can produce wrongly disconnected mesh components at low resolutions (see Figs. 1, 10). At the same time, it means that our starting surface mesh Ω^0 needs to agree with the topology of the surface to be reconstructed.

Careful consideration must also be given to the mesh resolution during our flow, as encoded in the remesher's target edge-length, h. As shown in Fig. 3, our flow is capable of recovering much more faithful surface detail than existing grid-based methods. Thus, we naturally wish to provide it with enough degrees of freedom (sufficiently low target edge-length h) to accurately represent the surface. At the same time, too many mesh vertices will make our flow iterations costly and the sphere reaching problem underconstrained.



Ideally, then, we wish our flow to produce the lowest possible resolution mesh that can explain all SDF samples. We will achieve this by starting from a very high value of h and running our flow until convergence, as identified by the energy's failure to decrease further than by a tolerance ($10^{-3}\varepsilon$) in the past 10 iterations, to obtain a coarse approximation of the surface (see inset). We will then halve h and run our flow again, noting that our output-sensitive remesher will only refine the regions of the shape that are contributing to the energy (i.e., those that need the additional resolution). We repeat this process until a minimum h_{min} is reached, by default set to be the average distance between SDF samples $\mathbf{p}_i$. Once $h_{\min}$ is reached, we run our flow until the energy has not decreased by more than $10^{-3}\varepsilon$ in the past 100 iterations.

Figure 13: Our flow is not limited to genus zero shapes, but the topology of the initial surface must match the reconstruction's. Marching Cubes may provide a good starting surface in these non-genus-zero cases.

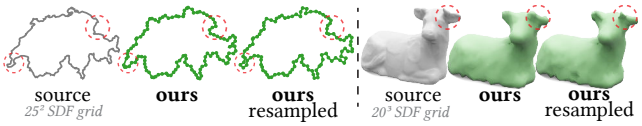


Figure 14: We can further improve the reconstruction quality on coarse grids by adding just a few SDF samples after the flow has converged, due to the gridless nature of our method.

Batching. In practice, our algorithm's computational cost is dominated by the assembly of $\mathbf{A}$, which requires n signed distance and closest point queries between each sphere origin and the current reconstruction mesh. Even though we manage to resolve each query in logarithmic time by assembling and using a bounding volume hierarchy, computing all n queries can be costly. Thus, for large values of n (e.g., $n > 50^3$), we propose using randomly chosen batches of spheres at each iteration. Empirically, we note that interior spheres are more critical to our flow's stability; therefore, we only batch exterior spheres. By default, we make the batch size $\min(n, 20000)$.

5 RESULTS & EXPERIMENTS

Implementation details. We implemented our algorithm in Python using GPytoolbox [Sellán and Stein 2023] for common geometry processing subroutines including our flow's remesher as well as the Marching Cubes reconstruction [Lorensen and Cline 1987] used in our comparisons. Our comparisons to Neural Dual Contouring [Chen et al. 2022a] use the authors' publicly available implementation, including following their preprocessing instructions. We report timings on a 20-Core M1 Ultra Mac Studio with 128GB RAM. We rendered our figures in Blender, using Blender-Toolbox [Liu 2023].

Our method's complexity is determined by three algorithmic steps that must be executed at each flow iteration. First, the signed distances from each $\mathbf{p}_i$ to the current mesh are computed with a complexity of $O((b+m)\log(m))$ (where m is the number of current mesh vertices and b is the batch size). Secondly, the linear system in Eq. (12) adds an $O(m^{1+f})$ term, where f accounts for the sparse system solve (we cannot take advantage of precomputation as the mesh changes in every iteration). Finally, our remeshing step adds an $O(\tilde{m})$ term, where $\tilde{m} < m$ is the size of the active region of the current mesh. In the limit, this means each flow iteration will be asymptotically dominated by $\max(b\log(m), m^{1+f})$; in practice, we find this to be the case except for very low values of m and b, where the constant factors in the remesher complexity dominate.

Our input shapes are scaled to fit the box $[-\frac{1}{2}, \frac{1}{2}]^n$, and our SDF grids are constructed in $[-1, 1]^n$. We use default parameters, unless otherwise specified in the supplemental material. Unless otherwise specified, we initialize our examples with a unit icosahedral sphere, but our flow can handle other initializations (Fig. 13).

Figure 15: SDF sampled from the same source on a grid and with the same number of samples on a noisy point cloud of the source. Alternate sampling strategies unavailable to grid-based methods allow us to recover more information with the same number of SDF samples.

5.1 Comparisons

The sole input to our algorithm is a set of query points $\mathbf{p}_i$ and corresponding SDF values s_i. In the specific case where these samples are placed on a structured grid, this input matches that of the timeless reconstruction algorithm Marching Cubes [Lorensen and Cline 1987]. By allowing for the training on a vast dataset and the storing of a large number of network weights, recent advances like Neural Dual Contouring [Chen et al. 2022a] have been shown to outperform most other reconstruction algorithms.

Qualitative comparisons. Throughout the paper, we qualitatively show our algorithm's improved performance against Marching Cubes (MC). At low resolutions, MC often produces little more than disconnected "blobs", often missing entire regions of the shape (see Figs. 1, 10). By contrast, our method shines at these resolutions, where exploiting all the global information provided by the SDF samples can recover features completely absent from the MC reconstruction (see Figs. 3 and 15), an advantage that is preserved even if the data is upsampled artificially to denser grids (see Fig. 12). Even at higher resolutions, our global SDF-aware reconstruction captures significantly more detail (see Figs. 8, 9 and 11).

In Figs. 1, 10, and 21, we additionally compare our algorithm's effectiveness with Neural Dual Contouring [Chen et al. 2022a]. To make the comparison as generous as possible, we used the highest performing version of the authors' publicly available trained models [Chen et al. 2022b], NDCx, which combines their network with elements of their previous work's learned model [Chen and Zhang 2021]. Even though their data-driven approach manages to outperform Marching Cubes in almost all our tests, we qualitatively find that our purely geometric algorithm consistently outperforms both at low and medium resolutions despite requiring no training.

Quantitative comparisons. Inspired by the evaluations in the work by Chen et al. [2022a], we also compare our algorithm's performance quantitatively. In Fig. 21, we run our algorithm using its default parameters as well as Marching Cubes and NDCx on shapes from a diverse set of origins whose SDFs have been sampled at different resolutions. In our supplemental material, we attach a table comparing the Hausdorff distance to the ground truth mesh as well as Chamfer distance and our own SDF energy $\mathcal{E}_\phi$, while Table 1 shows the average values for each resolution. While placing fewer requirements on the input (any set of points $\mathbf{p}_i$ versus a

Grid size	Hdf MC	Hdf NDCx	Hdf Ours	Chr MC	Chr NDCx	Chr Ours	$\mathcal{E}_\phi$ MC	$\mathcal{E}_\phi$ NDCx	$\mathcal{E}_\phi$ Ours	Time Ours
6^3	0.3351	0.2597	**0.1236**	0.1918	0.1135	**0.0569**	31.4645	10.2969	**0.1091**	1.1214
10^3	0.2518	0.1954	**0.0846**	0.1053	0.0662	**0.0343**	13.8933	6.2637	**0.1152**	1.2686
20^3	0.1486	0.1163	**0.0631**	0.0465	0.0311	**0.0210**	4.7667	2.8065	**0.1377**	4.7881
30^3	0.0756	0.0494	**0.0396**	0.0206	0.0127	**0.0118**	0.6125	0.1451	**0.0280**	6.0939
40^3	0.0581	**0.0366**	0.0417	0.0143	**0.0089**	0.0100	0.3933	0.0910	**0.0135**	6.6929
50^3	0.0501	0.0360	**0.0253**	0.0107	0.0077	**0.0074**	0.2451	0.1042	**0.0050**	9.4438

Table 1: Across a diverse set of examples and resolutions, our flow exhibits lower Hausdorff ("Hdf"), Chamfer ("Chr") and SDF ($\mathcal{E}_\phi$) errors than Marching Cubes, while surpassing or matching data-driven approaches like Neural Dual Contouring. Time given in seconds. Data averaged over Table 2 (supplemental).

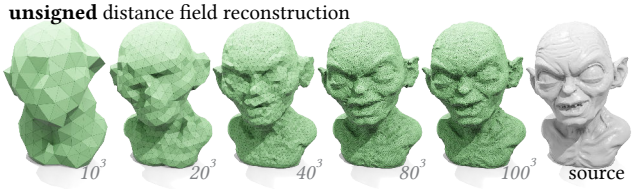


Figure 16: By relaxing the constraints in our method, our flow can be seamlessly applied to *unsigned* distance fields at diverse resolutions.

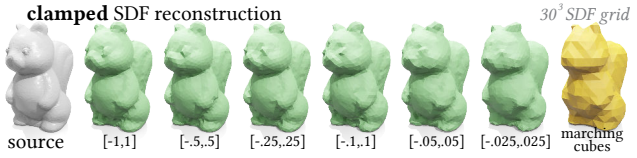


Figure 17: *Clamped* or *truncated* SDFs discard information our method needs to capture the shape's detail, but our method degrades gracefully and still outperforms Marching Cubes even for aggressive clamping parameters.

structured grid), our algorithm consistently outperforms Marching Cubes across the board, often by several integer factors. While requiring no training, our algorithm surpasses NDCx at low resolutions and remains competitive at medium and higher resolutions. We note that MC requires between 20^3 and 30^3 SDF grid samples to match the accuracy of our algorithm at the lowest of resolutions (6^3 grid). Thus, our algorithm reduces memory storage requirements for equal surface accuracy by a factor of between 37 and 125.

5.2 Parameters

A number of parameters affect our method's ability to extract all information from its SDF input. Among these, the most crucial is the minimum mesh edge-length h_{min}. Choosing h_{min} too high can cause the method to miss information available in the SDF samples. A very small h_{min} can negatively affect performance, and also underconstrain the problem, leading to (completely valid) solutions that have high-frequency noise. Our remeshing procedure contains a regularization step that combats this noise, but does not completely obviate it. Empirically, we find that setting h_{min} to be the average closest-distance between samples $\mathbf{p}_i$ (i.e., the gridless analogue of the grid edge-length) is a useful heuristic, whose reliability we show across resolutions in Fig. 21, Table 1 and the supplemental.

5.3 SDF sampling

While many algorithms rely on SDF samples to be located on a structured (regular or not) grid, our method is completely agnostic to the position of the samples $\mathbf{p}_i$. We can take advantage of this in multiple ways. For example, we can run our algorithm, unchanged, on SDF data sampled on fully unstructured point clouds, exploiting prior information (Fig. 15). In settings where the source SDF *function* is available to be queried, we can add more samples $(\mathbf{p}_i, s_i)$ after our method has converged, and run it from the previous result (Fig. 14) to incrementally improve the reconstruction. Our heuristic for adding samples is to generate m_{trial} samples on Ω, randomly displace them in the normal direction by a normal distribution scaled by 0.05, and select the m_{new} samples farthest away from the surface of any SDF sphere (but at most one per mesh element). By default, $m_{\text{new}} = 2\sqrt{n}$ in 2D, $m_{\text{new}} = 2\sqrt[3]{n}$ in 3D, and $m_{\text{trial}} = 50m_{\text{new}}$.

5.4 Beyond SDFs

Signed Distance Fields are a powerful representation that we have shown can be exploited to obtain a surprisingly large amount of information about a given object. Often, however, computing exact SDFs can be costly or impracticable, forcing one to relax some of the assumptions in the traditional SDF definition.

Consider the case of *unsigned* distance fields, which lack the inside-outside information contained in the sign of traditional SDFs. As shown in Fig. 16, our flow can very easily be employed to reconstruct meshes from these functions, merely by always making $\sigma_i = 1$ in (8). Intuitively, this means we move the surface towards the closer of the sphere's two possible tangent points, $\mathbf{t}_i$.

Another relaxation of SDFs are *clamped*, *truncated*, or *narrow band* SDFs, that take a constant value at spatial positions "sufficiently far" from the surface. This is a common representation in highly performant modelling applications and, more recently, in machine learning models when one wants to focus learning near the object's surface (see, e.g., [Park et al. 2019]). Generalizing our algorithm to these representations is conceptually simple: for a given clamp value σ_c, we allow the tangency requirement (but not the intersection-free one) to be violated for those spheres with radii larger than σ_c. In practice, this amounts to zeroing out the i-th row

Figure 18: A *swept volume* SDF is accurate only outside the object. Inside, it is only a bound on distance. By relaxing its assumptions, we can use our method for swept volume reconstruction.

of $\mathbf{A}$ if $|\phi(\mathbf{p}_i, \Omega^t)| > |s_i| > \sigma_c$. Our algorithm relies on faraway spheres to provide additional information about the reconstruction; therefore, using a clamped SDF necessarily results in a progressive loss of detail (see Fig. 17).

Yet another common SDF-based representation is formed by instead providing bounds on the true shape's signed distance (these are referred to as *conservative* SDFs by Takikawa et al. [2022]). Such SDFs appear naturally as the output of Boolean operations on signed distance functions. A recently studied example of this are *swept volumes*, which can be represented by taking the minimum of the SDF of an object along a trajectory; however, this representation is only an exact SDF *outside* the volume, while only a bound inside. As we show in Fig. 18, all that is needed to apply our flow to swept volume reconstruction is to relax the tangency constraint of the negative-sign spheres only. In practice, this amounts to zeroing the i-th row of A if $|\phi(\mathbf{p}_i, \Omega^t)| > |s_i|$ and $s_i < 0$. We believe this to be a promising application of our work, as swept volume approximate SDFs are often extremely costly to query [Sellán et al. 2021].

6 DISCUSSION AND CONCLUSIONS

We have leveraged our new tangent-spheres interpretation to develop an effective isosurfacing method, called *Reach for the Spheres*, that exploits the full representational power of SDFs. Using only geometric information present in a standard discrete SDF, we are able to recover noticeably more detail than previous general-purpose isosurfacing schemes. Our method especially shines on low-resolution SDF grids, where it is able to exploit every last bit of information that other methods might miss, and it can match traditional methods for high resolutions. By releasing our method to the Graphics community, we hope to renew interest in lightweight, low-resolution SDF representations and enable novel, scalable applications.

Our method is not yet robust to self-intersections, nor does it support topology changes, as needed to straightforwardly handle difficult multi-component or nonzero genus shapes. Thus, as a limitation, our method can exhibit self-intersection and pinching effects due to singularities in the discrete flow (Fig. 19). Our algorithm's current inability to handle these singularities also limits its efficacy on noisy SDF data (see Fig. 20). We are optimistic that existing mesh-based fluid simulation surface tracking techniques [Wojtan et al. 2011] can help overcome these restrictions.

Furthermore, a surface that perfectly satisfies a given discrete SDF can often still have significant flexibility at the finer scales; an exciting direction is to incorporate specific priors for particular applications, via additional regularization or data-driven approaches.

Figure 19: Like many geometric flows, our algorithm can occasionally produce singularities, corresponding to attempts to dynamically change topology.

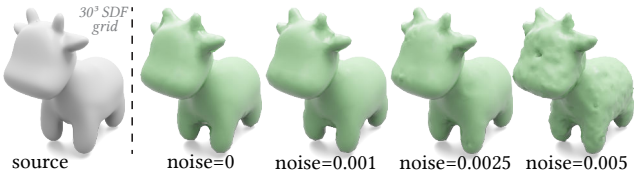


Figure 20: Adding Gaussian noise to the SDF input values, with increasing standard deviation. For small values, our flow degenerates gracefully. At a standard deviation of 0.005 (i.e., 0.5% of the shape's bounding box length), our flow hits a singularity before the stopping criterion is reached.

There is also ample room to improve the performance of our method, by using more elaborate methods for closest point computations, solving linear equations, and remeshing.

Beyond surface reconstruction, a vast array of other graphics techniques rely on discrete SDFs across simulation, geometry processing, and rendering. We look forward to exploring whether incorporating the tangent-spheres perspective can yield comparable improvements for these applications as well.

ACKNOWLEDGEMENTS

This work is funded in part by the Natural Sciences and Engineering Research Council of Canada (Grant RGPIN-2021-02524), an NSERC Vanier Scholarship and an Adobe Research Fellowship.

We thank Abhishek Madan for technical help and for proofreading. We thank Aravind Ramakrishnan, and Hsueh-Ti Derek Liu for proofreading. The first author would also like to thank the second and third authors for inviting her to visit their respective institutions, which served as the foundation for this collaboration.

We acknowledge the authors of the 3D models used throughout this paper and thank them for making them available for academic use. Figures in this work contain the koala [Thunk3D scanner 2019], springer [TimEdwards 2014], tower [jansentee3d 2018], bunny [The Stanford 3D Scanning Repository 1994], turtle [YahooJAPAN 2013b], armadillo [The Stanford 3D Scanning Repository 1996], cat [billyd 2016], duck [Crane 2013a], cow [pmoews 2016], strawberry [GSC_Nakamura 2013], plush toy [cosmicblobs 2021], spot [Crane 2013b], scorpion [YahooJAPAN 2013a], mushroom [Holinaty 2020], Nefertiti [Al-Badri and Nelles 2016], Max Planck [Max Planck Institute 2023], Igea [Cyberware Inc. 2023b], horse [Cyberware Inc. 2023a], Argonath [Patrick Bentley 2017], squirrel [Aim Shape repository 2011], teddy [Cornelis Brouwers 2014] and Lucy [The Stanford 3D Scanning Repository 2023] meshes.

REFERENCES

Aim Shape repository. 2011. Squirrel. https://www.thingiverse.com/thing:11705.

Nora Al-Badri and Jan Nikolai Nelles. 2016. Nefertiti. https://github.com/odedstein/meshes/tree/master/objects/nefertiti.

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational Tetrahedral Meshing. In *ACM SIGGRAPH 2005 Papers*. 617–625. https://doi.org/10.1145/1186822.1073238

Sai Praveen Bangaru, Michaël Gharbi, Fujun Luan, Tzu-Mao Li, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable rendering of neural SDFs through reparameterization. In *SIGGRAPH Asia 2022 Conference Papers*. Article 5, 9 pages. https://doi.org/10.1145/3550469.3555397

Christopher Batty. 2011. Problem: Reconstructing meshes with sharp features from signed distance data. Personal website, https://web.archive.org/web/20111112202136/http://www.cs.columbia.edu/~batty/misc/levelset_meshing/level_set_reconstruction.html.

billyd. 2016. Cat Stretch. https://www.thingiverse.com/thing:1565405.

Mario Botsch and Leif Kobbelt. 2004. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 185–192. https://doi.org/10.1145/1057432.1057457

Alan Brunton and Lubna Abu Rmaileh. 2021. Displaced Signed Distance Fields for Additive Manufacturing. *ACM Trans. Graph.* 40, 4, Article 179 (2021), 13 pages. https://doi.org/10.1145/3450626.3459827

Dennis R. Bukenberger and Hendrik P. A. Lensch. 2021. Be Water My Friend: Mesh Assimilation. *Vis. Comput.* 37, 9–11 (2021), 2725–2739. https://doi.org/10.1007/s00371-021-02183-6

Tony Chan and Luminita Vese. 1999. An active contour model without edges. In *Scale-Space Theories in Computer Vision*. Springer, 141–151. https://doi.org/10.1007/3-540-48236-9_13

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022a. Neural dual contouring. *ACM Trans. Graph.* 41, 4, Article 104 (2022), 13 pages. https://doi.org/10.1145/3528223.3530108

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022b. Neural dual contouring – GitHub repository. https://github.com/czq142857/NDC.

Zhiqin Chen and Hao Zhang. 2021. Neural marching cubes. *ACM Trans. Graph.* 40, 6, Article 251 (2021), 15 pages. https://doi.org/10.1145/3478513.3480518

Cornelis Brouwers. 2014. Teddy. https://www.thingiverse.com/techkey/designs.

cosmicblobs. 2021. cheburashka. accessed at libigl https://github.com/libigl/libigl-tutorial-data.

Keenan Crane. 2013a. bob. https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/.

Keenan Crane. 2013b. Spot. https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph.* 32, 4, Article 61 (2013), 10 pages. https://doi.org/10.1145/2461912.2461986

Cyberware Inc. 2023a. Horse. https://github.com/alecjacobson/common-3d-test-models/blob/master/data/horse.obj (year denotes online access).

Cyberware Inc. 2023b. Igea. https://github.com/alecjacobson/common-3d-test-models/blob/master/data/igea.obj (year denotes online access).

Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. 2015. A survey on implicit surface polygonization. *ACM Computing Surveys (CSUR)* 47, 4, Article 60 (2015), 39 pages. https://doi.org/10.1145/2732197

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 317–324. https://doi.org/10.1145/311535.311576

O. Faugeras and R. Keriven. 1998. Variational principles, surface evolution, PDEs, level set methods, and the stereo problem. *IEEE Transactions on Image Processing* 7, 3 (1998), 336–344. https://doi.org/10.1109/83.661183

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 23–30. https://doi.org/10.1145/383259.383261

Sarah F Frisken, Ronald N Perry, Alyn P Rockwood, and Thouis R Jones. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 249–254. https://doi.org/10.1145/344779.344899

Arnulph Fuhrmann, Gerrit Sobotka, and Clemens Groß. 2003. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon*, Vol. 2003. 58–65.

GSC_Nakamura. 2013. Strawberry. https://www.thingiverse.com/thing:153548.

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Trans. Graph.* 39, 4, Article 126 (2020). https://doi.org/10.1145/3386569.3392415

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545. https://doi.org/10.1007/s003710050084

Adrian Hilton, Andrew J Stoddart, John Illingworth, and Terry Windeatt. 1996. Marching triangles: range image fusion for complex object modelling. In *Proceedings of 3rd IEEE international conference on image processing*, Vol. 2. IEEE, 381–384. https://doi.org/10.1109/ICIP.1996.560840

Josh Holinaty. 2020. Mushroom. https://github.com/odedstein/meshes/tree/master/objects/mushroom.

jansentee3d. 2018. Dragon Tower. https://www.thingiverse.com/thing:3155868.

Mark W Jones, J Andreas Baerentzen, and Milos Sramek. 2006. 3D distance fields: A survey of techniques and applications. *IEEE Transactions on visualization and Computer Graphics* 12, 4 (2006), 581–599. https://doi.org/10.1109/TVCG.2006.56

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. https://doi.org/10.1145/566570.566586

Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can Mean-Curvature Flow be Modified to be Non-singular? *Computer Graphics Forum* 31, 5 (2012), 1745–1754. https://doi.org/10.1111/j.1467-8659.2012.03179.x

Leif P Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. 2001. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 57–66. https://doi.org/10.1145/383259.383265

Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2016. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Trans. Graph.* 35, 1, Article 8 (2016), 16 pages. https://doi.org/10.1145/2753755

Yiyi Liao, Simon Donne, and Andreas Geiger. 2018. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2916–2925. https://doi.org/10.1109/CVPR.2018.00308

Hsueh-Ti Derek Liu. 2023. BlenderToolbox. https://github.com/HTDerekLiu/BlenderToolbox.

Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. 2022. Regularized Deep Signed Distance Fields for Reactive Motion Generation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 6673–6680. https://doi.org/10.1109/IROS47612.2022.9981456

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* 21, 4 (1987), 163–169. https://doi.org/10.1145/37401.37422

Jaehwan Ma, Sang Won Bae, and Sunghee Choi. 2012. 3D medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer* 28 (2012), 7–19. https://doi.org/10.1145/37401.37422

Max Planck Institute. 2023. Max Planck bust. https://github.com/alecjacobson/common-3d-test-models/blob/master/data/max-planck.obj (year denotes online access).

Ken Museth, David E Breen, Ross T Whitaker, and Alan H Barr. 2002. Level set surface editing operators. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 330–338. https://doi.org/10.1145/566570.566585

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (2 ed.). Springer New York. https://doi.org/10.1007/978-0-387-40065-5

Stanley Osher and Ronald Fedkiw. 2003. *Level set methods and dynamic implicit surfaces*. Springer New York. https://doi.org/10.1007/b98879

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 165–174. https://doi.org/10.1109/CVPR.2019.00025

Patrick Bentley. 2017. Argonath. https://www.thingiverse.com/thing:2243300.

pmoews. 2016. Traditional Cow. https://www.thingiverse.com/thing:1602712/files.

Daniel Rebain, Baptiste Angles, Julien Valentin, Nicholas Vining, Jiju Peethambaran, Shahram Izadi, and Andrea Tagliasacchi. 2019. LSMAT least squares medial axis transform. In *Computer Graphics Forum*, Vol. 38. 5–18. https://doi.org/10.1111/cgf.13599

Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. 2020. Meshsdf: Differentiable iso-surface extraction. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Article 1884, 11 pages.

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested Cages. *ACM Trans. Graph.* 34, 6, Article 170 (2015). https://doi.org/10.1145/2816795.2818093

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2021. Swept volumes via spacetime numerical continuation. *ACM Trans. Graph.* 40, 4, Article 55 (2021), 11 pages. https://doi.org/10.1145/3450626.3459780

Silvia Sellán, Yang Shen Ang, Jacob Kesten, and Alec Jacobson. 2022. Opening and Closing Surfaces. *ACM Trans. Graph.* 41, 6, Article 198 (2022), 12 pages. https://doi.org/10.1145/3414685.3417778

Silvia Sellán and Oded Stein. 2023. gpytoolbox: A Python Geometry Processing Toolbox. https://gpytoolbox.org/.

James Albert Sethian. 1999. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge university press.

Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen–Or. 2006. Competing Fronts for Coarse–to–Fine Surface Reconstruction. *Computer Graphics*

Forum 25, 3 (2006), 389–398. https://doi.org/10.1111/j.1467-8659.2006.00958.x

Nicholas Sharp and Alec Jacobson. 2022. Spelunking the deep: guaranteed queries on general neural implicit surfaces via range analysis. *ACM Trans. Graph.* 41, 4, Article 107 (2022), 16 pages. https://doi.org/10.1145/3528223.3530155

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. *ACM Trans. Graph.* 42, 4, Article 37 (jul 2023), 16 pages. https://doi.org/10.1145/3592430

Barton T. Stander and John C. Hart. 1997. Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. 279–286. https://doi.org/10.1145/258734.258868

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4, Article 77 (2018), 14 pages. https://doi.org/10.1145/3197517.3201303

Towaki Takikawa, Andrew Glassner, and Morgan McGuire. 2022. A dataset and explorer for 3D signed distance functions. *Journal of Computer Graphics Techniques Vol* 11, 2 (2022).

The Stanford 3D Scanning Repository. 1994. Stanford Bunny. http://graphics.stanford.edu/data/3Dscanrep/.

The Stanford 3D Scanning Repository. 1996. Armadillo. http://graphics.stanford.edu/data/3Dscanrep/.

The Stanford 3D Scanning Repository. 2023. Lucy. http://graphics.stanford.edu/data/3Dscanrep/ (year denotes online access).

Thunk3D scanner. 2019. koala bear. https://sketchfab.com/3d-models/koala-bear-221d8d6519944a65b473ea56fc032570.

TimEdwards. 2014. glChess chess set 2. https://www.thingiverse.com/thing:335658.

Kees Van Overveld and Brian Wyvill. 2004. Shrinkwrap: An efficient adaptive algorithm for triangulating an iso-surface. *The visual computer* 20 (2004), 362–379. https://doi.org/10.1007/s00371-002-0197-4

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable signed distance function rendering. *ACM Trans. Graph.* 41, 4, Article 125 (2022), 18 pages. https://doi.org/10.1145/3528223.3530139

Chris Wojtan, Matthias Müller-Fischer, and Tyson Brochu. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*. 1–84. https://doi.org/10.1145/2037636.2037644

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum* 41, 2 (2022), 641–676. https://doi.org/10.1111/cgf.14505

YahooJAPAN. 2013a. Scorpion. https://www.thingiverse.com/thing:182363.

YahooJAPAN. 2013b. Turtle. https://www.thingiverse.com/thing:182332.

Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. 2001. Fast surface reconstruction using the level set method. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*. 194–201. https://doi.org/10.1109/VLSM.2001.938900

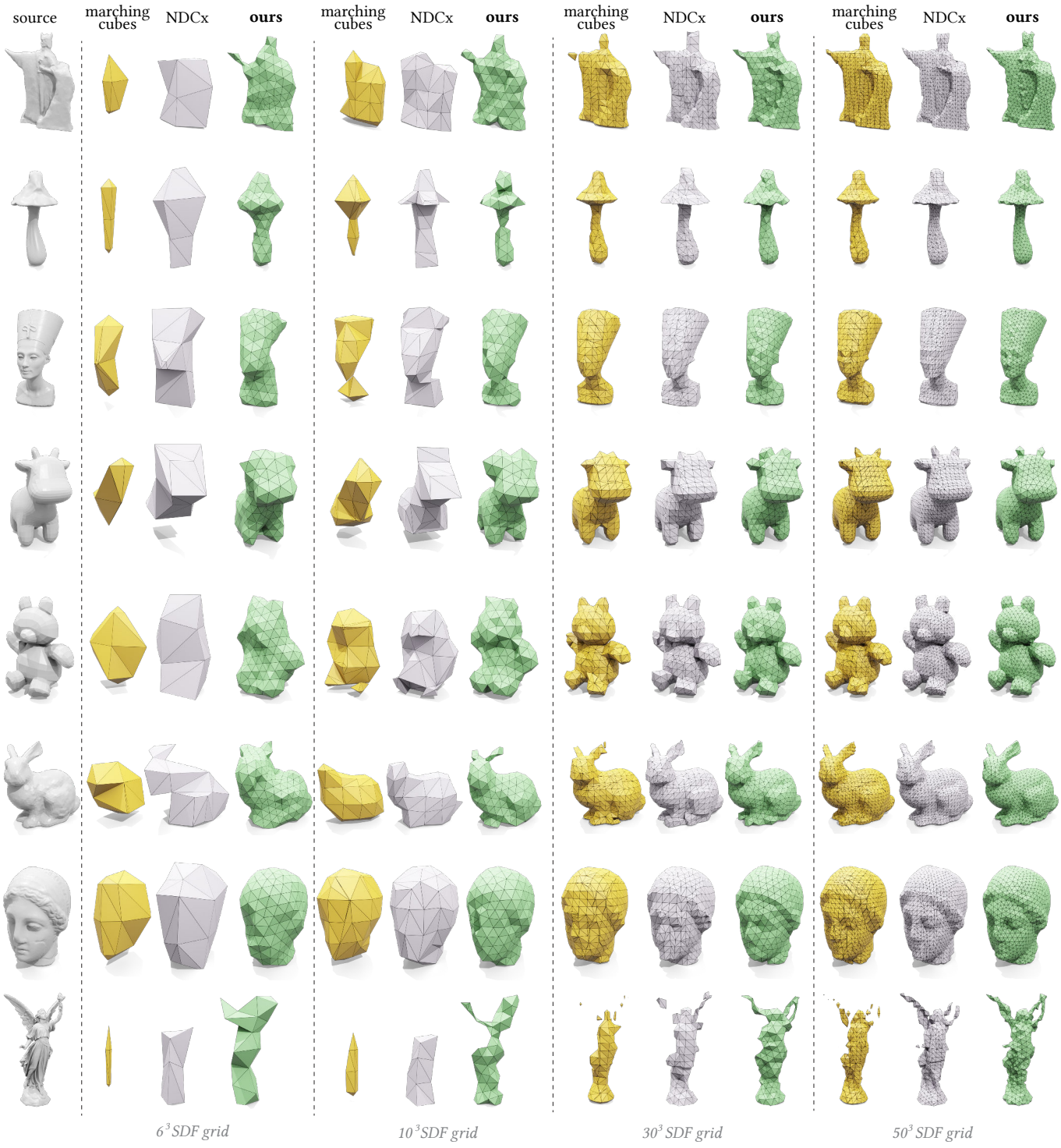


Figure 21: Our algorithm strikingly outperforms Marching Cubes and Neural Dual Contouring (NDCx) at low and medium resolutions.

SUPPLEMENTAL MATERIAL

Parameters

In this section we list $h_{\min}$ and non-default parameters used in this article.

Fig. 1. $h_{\min} = 0.02$ for grid size 10. $h_{\min} = 0.01$ for grid size 50. $\varepsilon = 10^{-4}$. 3 remesh iterations.

Fig. 3. $h_{\min} = 0.03$. $t_{\max} = 10$ in 3D.

Fig. 4. $h_{\min} = 0.01$.

Fig. 5. $h_{\min} = 0.02$. Batching turned off.

Fig. 8. $h_{\min} = 0.008$.

Fig. 10. Left to right: $h_{\min} = 0.05$, 0.02 and 0.008.

Fig. 11. For grid size n^3, $h_{\min} = \frac{2}{n}$, and $\varepsilon = \frac{10^{-2}}{n}$.

Fig. 14. $h_{\min} = 0.035$ in 2D; $h_{\min} = 0.04$ in 3D.

Fig. 15. $h_{\min} = 0.06$. $\varepsilon = 10^{-3}$

Fig. 20. $h_{\min} = 0.015$.

Detailed Quantitive Evaluation Data

Table 2 contains the detailed results of our quantitative evaluations for the SDF reconstruction problem on a variety of shapes, comparing our method with Marching Cubes and NDCx.

Shape	n	Hdf MC	Hdf NDCx	Hdf Ours	Chr MC	Chr NDCx	Chr Ours	$\mathcal{E}_\phi$ MC	$\mathcal{E}_\phi$ NDCx	$\mathcal{E}_\phi$ Ours	Time Ours
mushroom	6	0.1952	0.1489	**0.1417**	0.1274	0.1103	**0.0753**	11.7905	1.1676	**0.0519**	0.5147
mushroom	10	0.1221	**0.0819**	0.1023	0.0715	0.0456	**0.0420**	3.9187	0.4795	**0.0455**	1.1983
mushroom	20	0.0611	**0.0485**	0.0606	0.0345	**0.0197**	0.0200	0.5752	0.1567	**0.0519**	6.0584
mushroom	30	0.0546	**0.0233**	0.0407	0.0215	**0.0114**	0.0116	0.3133	0.0758	**0.0150**	3.9053
mushroom	40	0.0532	**0.0217**	0.0238	0.0165	0.0077	**0.0077**	0.2285	0.0286	**0.0092**	4.1968
mushroom	50	0.0382	**0.0186**	0.0188	0.0120	0.0065	**0.0062**	0.1012	0.0128	**0.0038**	7.1066
nefertiti	6	0.2007	0.1296	**0.0988**	0.1384	0.0671	**0.0462**	11.9997	0.6109	**0.0520**	1.8593
nefertiti	10	0.1826	0.0760	**0.0645**	0.0976	0.0379	**0.0188**	5.1541	0.3250	**0.0427**	1.6463
nefertiti	20	0.0744	0.0419	**0.0381**	0.0304	0.0146	**0.0142**	0.9196	0.0742	**0.0416**	6.6072
nefertiti	30	0.0406	0.0382	**0.0286**	0.0144	**0.0080**	0.0086	0.2235	0.0414	**0.0145**	10.2500
nefertiti	40	0.0382	0.0287	**0.0246**	0.0119	**0.0063**	0.0069	0.1655	0.0190	**0.0100**	6.4140
nefertiti	50	0.0341	0.0316	**0.0264**	0.0076	**0.0056**	0.0059	0.0459	0.0128	**0.0032**	11.1034
bunny	6	0.2947	**0.1669**	0.1769	0.1997	0.0938	**0.0626**	15.2121	1.9051	**0.0438**	0.9502
bunny	10	0.4295	0.4051	**0.0860**	0.1199	0.0982	**0.0332**	16.6344	11.3773	**0.0685**	1.3017
bunny	20	0.1975	0.1348	**0.0612**	0.0453	0.0271	**0.0182**	3.6855	1.0939	**0.0527**	4.2061
bunny	30	0.0580	**0.0383**	0.0397	0.0158	0.0110	**0.0097**	0.2503	0.0349	**0.0226**	8.9931
bunny	40	0.0638	**0.0273**	0.0351	0.0131	0.0086	**0.0080**	0.1889	0.0196	**0.0097**	10.1580
bunny	50	0.0487	0.0309	**0.0219**	0.0090	0.0069	**0.0063**	0.0789	0.0096	**0.0035**	14.7060
argonath	6	0.3763	0.2707	**0.1146**	0.2341	0.0827	**0.0393**	42.1481	6.2328	**0.1402**	0.7281
argonath	10	0.3271	0.2104	**0.0751**	0.0919	0.0548	**0.0314**	12.6527	5.5259	**0.2200**	0.9053
argonath	20	0.1707	0.1538	**0.0486**	0.0482	0.0334	**0.0214**	4.4039	1.7451	**0.0794**	5.2204
argonath	30	0.0953	0.0401	**0.0320**	0.0197	0.0120	**0.0119**	0.8446	0.1481	**0.0318**	4.0973
argonath	40	0.0513	**0.0282**	0.0303	0.0149	**0.0084**	0.0089	0.3563	0.0245	**0.0139**	6.2986
argonath	50	0.0443	**0.0237**	0.0274	0.0097	**0.0065**	0.0067	0.2083	0.0196	**0.0067**	8.0372
lucy	6	0.4025	0.3718	**0.0974**	0.1601	0.1128	**0.0549**	45.0055	19.7607	**0.5921**	0.3424
lucy	10	0.4490	0.4405	**0.1046**	0.1599	0.1356	**0.0499**	44.8871	29.1435	**0.4019**	0.7341
lucy	20	0.4054	0.3611	**0.0937**	0.1161	0.0943	**0.0349**	27.0360	20.5396	**0.8438**	4.6527
lucy	30	0.1243	0.1190	**0.0757**	0.0358	0.0251	**0.0232**	1.5015	0.4685	**0.0952**	4.6289
lucy	40	0.1162	0.1166	**0.0534**	0.0258	0.0183	**0.0164**	1.3426	0.4506	**0.0531**	6.1530
lucy	50	0.1409	0.1176	**0.0380**	0.0227	0.0178	**0.0140**	1.4041	0.8705	**0.0117**	5.4008
igea	6	0.1716	0.1026	**0.0602**	0.1192	0.0561	**0.0218**	4.3137	1.1548	**0.0158**	1.2822
igea	10	0.1078	0.0727	**0.0470**	0.0554	0.0294	**0.0164**	1.3410	0.3004	**0.0119**	2.3998
igea	20	0.0653	**0.0421**	0.0569	0.0203	0.0149	**0.0143**	0.2639	0.0748	**0.0151**	4.3184
igea	30	0.0386	**0.0276**	0.0292	0.0116	0.0093	**0.0087**	0.0687	0.0237	**0.0047**	6.2384
igea	40	0.0296	**0.0184**	0.0237	0.0089	**0.0071**	0.0074	0.0368	0.0061	**0.0023**	7.4121
igea	50	0.0259	**0.0136**	0.0207	0.0075	**0.0065**	0.0067	0.0222	0.0049	**0.0017**	9.0712
armadillo	6	0.5501	0.4145	**0.1573**	0.2729	0.1810	**0.0911**	78.0223	33.4996	**0.0651**	1.0500
armadillo	10	0.3669	0.2590	**0.1167**	0.1816	0.1028	**0.0490**	35.4634	11.6023	**0.1679**	1.0748
armadillo	20	0.1925	0.1766	**0.0495**	0.0658	0.0488	**0.0209**	6.9130	3.7000	**0.0893**	4.1866
armadillo	30	0.1382	0.0931	**0.0396**	0.0283	0.0189	**0.0129**	1.4777	0.4933	**0.0418**	5.0136
armadillo	40	0.1067	**0.0624**	0.1335	0.0191	**0.0116**	0.0221	1.2729	0.2819	**0.0149**	2.7483
armadillo	50	0.0770	0.0573	**0.0302**	0.0125	0.0091	**0.0085**	0.3559	0.0883	**0.0083**	8.8009
teddy	6	0.2284	0.1726	**0.1407**	0.1369	0.0836	**0.0587**	11.9344	2.0409	**0.0347**	1.7151
teddy	10	0.1692	0.1615	**0.1039**	0.0922	0.0655	**0.0418**	4.4468	1.6534	**0.0409**	1.2807
teddy	20	0.1280	0.1122	**0.0821**	0.0363	0.0262	**0.0255**	1.0904	0.4025	**0.0473**	4.4172
teddy	30	0.0480	**0.0279**	0.0353	0.0177	0.0115	**0.0113**	0.1661	0.0561	**0.0144**	7.2587
teddy	40	0.0410	**0.0232**	0.0473	0.0117	**0.0083**	0.0090	0.0641	0.0361	**0.0067**	5.6651
teddy	50	0.0343	**0.0266**	0.0275	0.0092	**0.0068**	0.0070	0.0284	0.0093	**0.0031**	8.0650
spot	6	0.2901	0.3463	**0.1325**	0.2069	0.1743	**0.0619**	24.3347	11.6865	**0.0313**	1.2620
spot	10	0.2188	0.1574	**0.0654**	0.1177	0.0621	**0.0291**	10.7422	2.1484	**0.0498**	1.0666
spot	20	0.0932	**0.0451**	0.0508	0.0313	**0.0176**	0.0189	1.0994	0.2004	**0.0602**	4.6396
spot	30	0.0654	0.0474	**0.0465**	0.0158	0.0101	**0.0092**	0.2841	0.0871	**0.0135**	4.3807
spot	40	0.0320	**0.0209**	0.0231	0.0095	0.0069	**0.0063**	0.0520	0.0293	**0.0043**	7.5936
spot	50	0.0251	0.0264	**0.0180**	0.0077	0.0059	**0.0056**	0.0302	0.0112	**0.0021**	11.0150

Table 2: Quantitative Evaluation Data

Neural Stochastic Screened Poisson Reconstruction

Silvia Sellán
University of Toronto
Canada
sgsellan@cs.toronto.edu

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

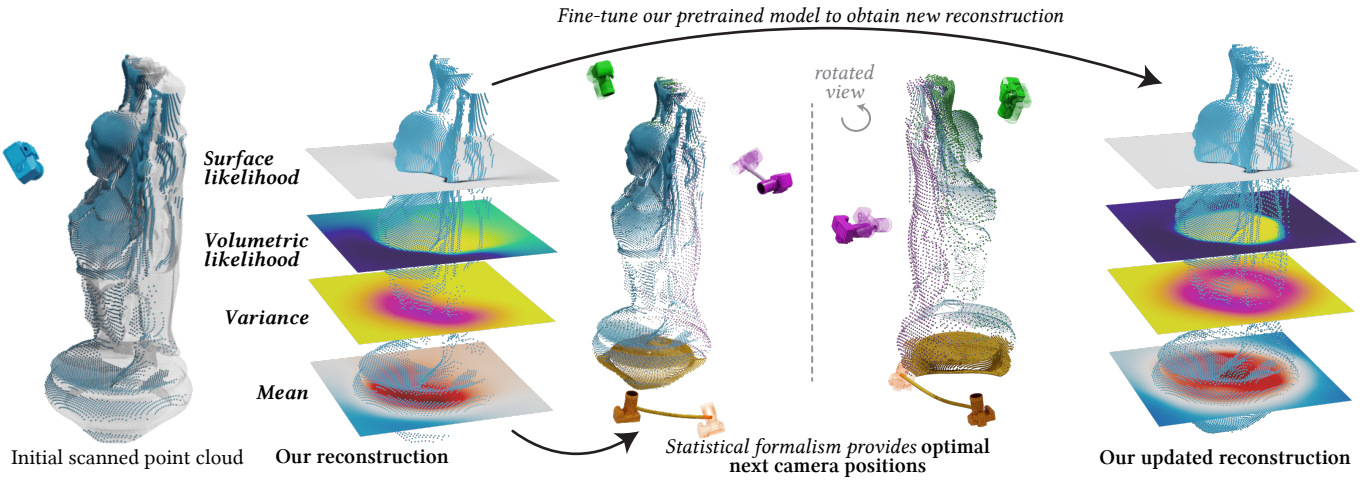


Figure 1: We use a neural network to quantify the reconstruction uncertainty in Poisson Surface Reconstruction (center left), allowing us to efficiently select next sensor positions (center right) and update the reconstruction upon capturing data (right).

ABSTRACT

Reconstructing a surface from a point cloud is an underdetermined problem. We use a neural network to study and quantify this reconstruction uncertainty under a Poisson smoothness prior. Our algorithm addresses the main limitations of existing work and can be fully integrated into the 3D scanning pipeline, from obtaining an initial reconstruction to deciding on the next best sensor position and updating the reconstruction upon capturing more data.

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**; *3D imaging*; *Active vision*; **Uncertainty quantification**.

KEYWORDS

neural surface reconstruction, uncertainty quantification

ACM Reference Format:
Silvia Sellán and Alec Jacobson. 2023. Neural Stochastic Screened Poisson Reconstruction. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3610548.3618162

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12…$15.00
https://doi.org/10.1145/3610548.3618162

1 INTRODUCTION

Surface reconstruction is the process of transforming a discrete set of points in space (a common format for captured 3D geometry) into a complete two-dimensional manifold for use in downstream scientific applications. Given the fundamentally underdetermined nature of the problem, algorithms must rely on priors to decide on an output surface.

Absent task-specific knowledge, the predominant geometry processing algorithm for surface reconstruction is *Poisson Surface Reconstruction* (PSR) [Kazhdan et al. 2006]. PSR encourages smoothness in the reconstruction through a Partial Differential Equation (PDE) whose solution can be computed efficiently and robustly. Drawing inspiration from it, Dai and Nießner [2022] recently introduced a neural approximation of PSR, which sidesteps the PDE perspective on the problem, achieving some performance gains at the cost of losing theoretical guarantees (see Figure 4), overfitting (see Figure 3) and additional requirements (e.g., sensor positioning).

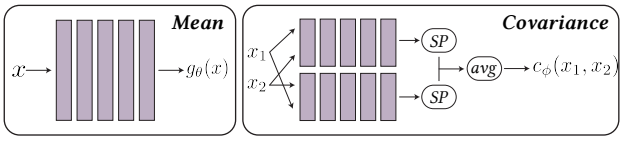


Figure 2: We use neural networks to parametrize the stochastic implicit function describing the reconstructed surface. The mean is a simple five-layered MLP while the covariance includes a SoftPlus (SP) pass and an averaging step to enforce positiveness and symmetry, respectively.

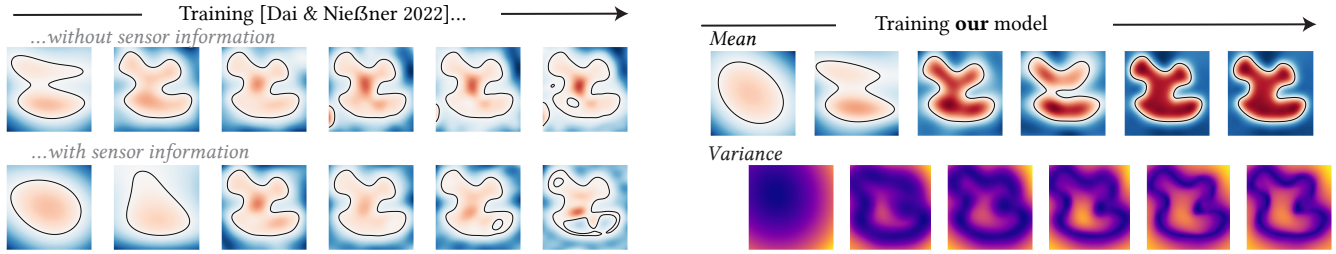


Figure 3: Unlike the Poisson-inspired model by Dai and Nießner [2022], we propose using a neural network to solve the Poisson equation in Poisson Surface Reconstruction, avoiding overfitting in sparsely sampled point clouds. Additionally, we provide a full statistical formalism, including variances (bottom right).

Statistically, PSR generates the most probable reconstruction based on the selected prior. This choice inherently defines an entire *posterior* distribution in the space of possible reconstructions. While *Stochastic* PSR [Sellán and Jacobson 2022] computes this distribution for the first time in the context of PSR, it demands a complex discretization scheme and relies on multiple approximations to achieve computational tractability.

We build on the work by Sellán and Jacobson [2022] and introduce a neural formulation of Stochastic PSR that provides a full statistical formalism of the reconstruction process while avoiding overfitting and requiring no additional sensor information. Unlike Sellán and Jacobson [2022], we parametrize the mean and covariance of the implicit field describing the reconstructed surface using a neural network (see Figure 2), which we optimize using gradient-based optimization on losses derived from the variational version of the Poisson equation. Our neural formulation also allows us to extend this stochastic perspective beyond the original PSR and into *Screened* PSR [Kazhdan and Hoppe 2013].

We showcase the power of our algorithm by showing its performance in a breadth of applications made possible by our novel neural perspective. In particular, we show how one can fully integrate our algorithm in the 3D scanning pipeline, from obtaining an initial reconstruction to defining a differential camera score that can guide the choice of the next best scanning position and efficiently updating the previous reconstruction (see Figure 1) by fine-tuning our network with additional data. We also explore promising avenues for future work, like latent space generalization over scanning positions for a given object or over a space of objects.

2 RELATED WORK

2.1 Surface reconstruction

Three-dimensional geometry is often captured by recording the distance from a sensor or *depth camera* to a real-world object [Özyeşil et al. 2017; Raj et al. 2020]. Combining the information from many sensors allows us to represent the raw captured geometry as a discrete set of points in space or *point cloud*. It is often possible to use properties about the sensor positioning or heuristics based on global or local cloud attributes [Hoppe et al. 1992; König and Gumhold 2009; Metzer et al. 2021; Schertler et al. 2017] to equip every point with a normal direction, allow for the slightly more complete representation of an *oriented point cloud*.

Despite their ubiquitousness, (oriented) point clouds are a fundamentally underdetermined surface representation: by specifying only a discrete set of points in space through which a surface passes, it describes a theoretically infinite number of possible surfaces. *Surface Reconstruction* algorithms (see [Berger et al. 2017] for a survey) use a *prior* to decide between them and output a fully determined surface, usually in a format appropriate for specific downstream tasks like a mesh or an implicit function. These priors range from simple geometric primitives [Schnabel et al. 2009] to global properties like symmetry [Pauly et al. 2008] or self-similarity [Williams et al. 2019], user-specified ones [Sharf et al. 2007] and, especially in recent years, data-driven [Groueix et al. 2018; Remil et al. 2017].

Absent task-specific knowledge, a commonly used prior is smoothness. This can be enforced explicitly by considering only surfaces parametrized by a smooth family of functions; for example, spatially-varying polynomials [Alexa et al. 2003; Levin 2004; Ohtake et al. 2005] and linear combinations of radial basis functions [Carr et al. 2001]. Smoothness can also be enforced variationally: *Poisson Surface Reconstruction* (PSR) [Kazhdan et al. 2006; Kazhdan and Hoppe 2013] encodes volumetric smoothness away from the input point cloud by minimizing the integrated gradient of the surface's implicit representation and remains one of the best performing general surface reconstruction algorithms in terms of robustness and efficiency (see Table 1 in [Berger et al. 2017]). While the authors solve this optimization problem using the Finite Element Method on a hierarchical grid, Dai and Nießner [2022] have recently proposed using a neural network for a similar task, albeit they suggest forgoeing the volumetric integration and instead minimizing the gradient only at the point cloud points (see Figs. 3 and 4). We cover PSR and its variants in more detail in Section 3.1.

2.2 Stochastic Surface Reconstruction

From a statistical perspective, the vast majority of surface reconstruction works limit themselves to outputting the likeliest surface given the point cloud observations and their assumed prior. Relatively fewer works take this stochastic perspective one step further and compute a *posterior* distribution of all possible surfaces conditioned on the observations. For example, Pauly et al. [2004] quantify the likelihood of any spatial point belonging to the reconstructed surface by measuring its alignment with the point cloud.

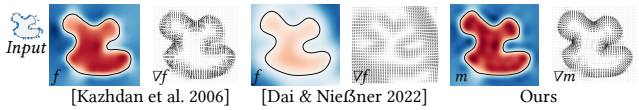


Figure 4: Even before overfitting, the result by Dai and Nießner [2022] does not replicate the PSR output, with non-zero gradients away from the data.

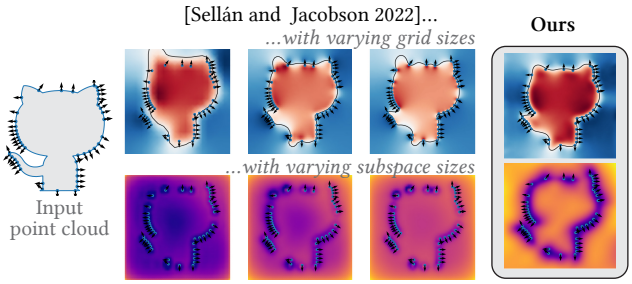


Figure 5: Sellán and Jacobson [2022] couple the reconstruction lengthscale with their discretization grid spacing, and require a subspace approximation. Our neural network discretizations avoids both issues.

More recently, *Stochastic Poisson Surface Reconstruction* [Sellán and Jacobson 2022] reinterprets the classic algorithm as a Gaussian Process, enabling the computation of statistical queries crucial to the reconstruction process and applications such as ray casting, point cloud repair, and collision detection. The authors utilize a Finite Element discretization to compute the mean and covariance functions of the posterior multivariate Gaussian distribution, which represents the likelihood of all possible reconstructions (see Section 3.2), resorting to a several approximations and parameter choices for computational tractability (see Figure 5). In contrast, our work proposes the parametrization of these functions using neural networks, optimizing them through gradient-based methods for a more efficient and flexible approach while still computing the same statistical quantities (see Figure 7).

2.3 Neural PDE solvers

We propose solving the Poisson equation in Stochastic PSR using a neural network. As such, our algorithm is one more application in the growing field of neural partial differential equation solvers. A broad class of these are *Physics-Informed Neural Networks* (PINNs) (see [Cuomo et al. 2022] for a literature review), which effectively soften a PDE and its boundary conditions into integral loss terms that are minimized with, e.g., stochastic gradient descent.

If a given PDE accepts a variational formulation, the above process can be done in a more principled way, as shown by Yu et al. [2018]. This is the case for the Poisson equation, which can be equivalently described as a variational Dirichlet energy minimization. This is noted by Sitzmann et al. [2020], who show the impressive performance of sinusoidal activation functions when applied to Dirichlet-type problems. We borrow from their observations and propose a network architecture with sine activations.

2.4 Next-Best-View planning

A key benefit of proposed approach is its integration in the 3D scanning process. Specifically, it allows us to compute a *score* function that quantifies how useful a proposed next sensor position would be for the reconstruction task. This is a common first step in the *active vision* or *next-best-view planning* pipeline, which has been a subject of study for decades (see, e.g., [Chen et al. 2011; Scott et al. 2003] for surveys). In it, prospective sensor placements may be scored by accounting for one or several factors like coverage [Bircher et al. 2016; Connolly 1985; Yamauchi 1997], navigation distance, expected reconstruction error [Vasquez-Gomez et al. 2014], scene segmentation entropy [Xu et al. 2015] and redundancy of multiple views [Lauri et al. 2020]. Orthogonally, works may need to rely on coarse shape priors for the reconstruction [Zhang et al. 2021; Zhou et al. 2020] or balance improving reconstruction in sampled areas with exploring new unsampled ones.

More recently, volumetric methods like those by Isler et al. [2016], and Daudelin and Campbell [2017] use simple heuristics (e.g., distance to the point cloud combined with visibility) to quantify the marginal likelihood of a given point in space being contained in the reconstructed object. This quantity is discretized onto a voxel grid and used to quantify the expected information gain from a given sensor position. Building on these works, our proposed utility function requires no heuristics, coming instead directly from the statistically formalized reconstruction process and, unlike [Daudelin and Campbell 2017], accounts for the possible spatial interdependencies along a single ray (see Figure 13). Further, since our reconstruction is parametrized by a neural network, this score is differentiable with respect to the sensor parameters, allowing for the efficient discovery of locally optimal camera placements (see Figure 6). While we introduce said novel, differentiable utility function, the development of a comprehensive next-best-view planning pipeline, which would encompass global searches, travel times, collision avoidance, and robot constraints, falls outside the scope of this paper.

Finally, outside of the point cloud reconstruction realm, the recent popularity of Neural Radiance Fields [Mildenhall et al. 2021] has also given rise to uncertainty-driven approaches for next-best-view planning in RGB multi-view representations (see, e.g., [Jin et al. 2023; Kong et al. 2023; Smith et al. 2022; Sucar et al. 2021]).

3 BACKGROUND

Given an oriented point cloud $\mathcal{P}$ with points $p_1, \ldots, p_n$ and corresponding (outward-facing) normal observations $\vec{n}_1, \ldots, \vec{n}_n$, we consider the implicit reconstruction task of finding a function $f : \mathbb{R}^d \to \mathbb{R}$ such that

$$f(p_i) = 0\,, \qquad \nabla f(p_i) = \vec{n}_i\,, \qquad \forall i \in \{1, \ldots, n\}\,. \tag{1}$$

The zero levelset $\mathcal{S} = f^{-1}(\{0\})$ is the reconstructed surface, whose interior is $\Omega = \{x \in \mathbb{R}^d \,:\, f(x) \leq 0\}$. We will be consistent with this convention that places *negative* implicit function values *inside* the reconstruction, and *positive* ones *outside*.

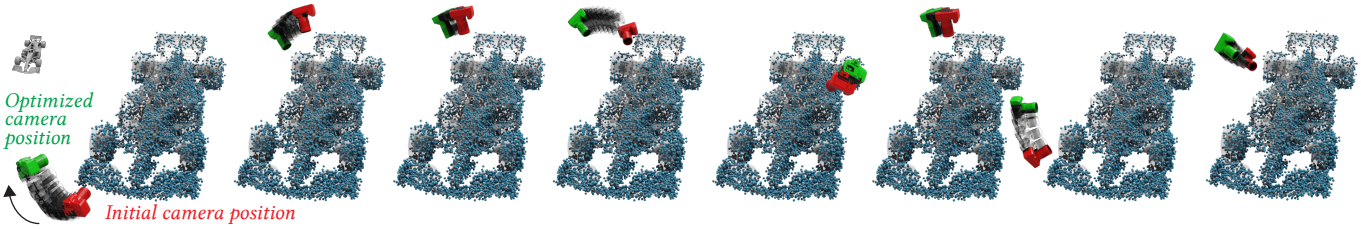


Figure 6: We provide a differentiable utility function that we can optimize to explore local next-best-views.

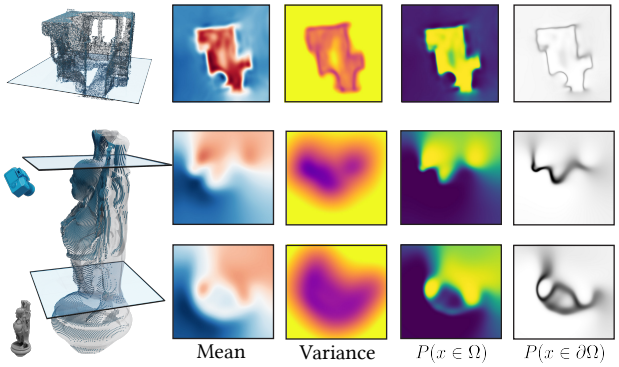


Figure 7: Like Sellán and Jacobson [2022], our algorithm can respond to statistical queries related to the reconstruction.

3.1 Poisson Surface Reconstruction

Poisson Surface Reconstruction (PSR) [Kazhdan et al. 2006] builds f in two steps. First, a smear kernel F is used to interpolate $\vec{n}_i$ into a vector field $\vec{v} : \mathbb{R}^d \to \mathbb{R}^d$ defined in a box B containing $\mathcal{P}$:

$$\vec{v}(x) = \sum_{i=1}^{n} F(x, x_i)\vec{n}_i \,. \tag{2}$$

Then, f is defined as the function whose gradient best matches $\vec{v}$:

$$f = \underset{g}{\operatorname{argmin}} \int_B \|\vec{v}(x) - \nabla g(x)\|^2 \, \mathrm{dx} \,. \tag{3}$$

This variational problem is equivalent to the Poisson equation

$$\Delta f = \nabla \cdot \vec{v}(x) \,, \tag{4}$$

which the authors discretize using the Finite Element Method on an octree and solve using a purpose-built multigrid algorithm. Since Eq. (4) alone does not uniquely determine f, a valid f is computed and its values shifted to best satisfy $f(p_i) = 0$.

In *Screened Poisson Surface Reconstruction*, Kazhdan and Hoppe [2013] circunvent this by adding a *screening* term to Eq. (3)

$$f = \underset{g}{\operatorname{argmin}} \int_B \|\vec{v}(x) - \nabla g(x)\|^2 \, \mathrm{dx} + \lambda \sum_{i=1}^{n} g(p_i)^2 \tag{5}$$

which translates into a Screened Poisson equation

$$(\Delta - \lambda I) f = \nabla \cdot \vec{v}(x) \,, \tag{6}$$

for a specific masking operator I.

3.2 Stochastic Poisson Surface Reconstruction

Screened or not, the output of Poisson reconstruction is a single function f. However, the reconstruction task is fundamentally uncertain: Eq. (1) alone is underdetermined and satisfied by an infinite number of possible functions f. When subject to appropriate boundary conditions, Poisson reconstruction selects one particular solution, which can be understood as the most likely solution under a given prior. Sellán and Jacobson [2022] formalize this statistical intuition by interpreting $(p_i, \vec{n}_i)$ as observations of a Gaussian Process and computing the posterior distribution

$$\vec{v} \,|\, (p_1, \vec{n}_1), \ldots, (p_1, \vec{n}_n) \sim \mathcal{N}(\vec{\mu}(x), \Sigma(x, x')) \,. \tag{7}$$

Eq. (3) is then enforced in the space of distributions, obtaining a posterior for f,

$$f \,|\, (p_1, \vec{n}_1), \ldots, (p_1, \vec{n}_n) \sim \mathcal{N}(m(x), k(x, x')) \,, \tag{8}$$

whose mean and covariance functions m, k are solutions to the variational problem

$$m = \underset{g}{\operatorname{argmin}} \int_B \|\vec{\mu}(x) - \nabla g(x)\|^2 \, \mathrm{dx} \,, \tag{9}$$

$$k = \underset{c}{\operatorname{argmin}} \iint_B \|\Sigma(x_1, x_2) - \mathbf{D}c(x_1, x_2)\|_F^2 \, \mathrm{dx}_1 \, \mathrm{dx}_2 \,, \tag{10}$$

where $\mathbf{D}c(x_1, x_2)$ is the $d \times d$ matrix whose i, j entries are

$$\left. \frac{\partial^2}{\partial a_i \partial b_j} c(a, b) \right|_{a=x_1, b=x_2} \tag{11}$$

In the same way of Eq. (3), Eqs. (9) and (10) can be written as Poisson-style PDEs that are solved using the Finite Element Method on a uniform or hierarchical grid. Like the original work by Kazhdan et al. [2006], Sellán and Jacobson [2022] shift the values of m and k after the fact to satisfy $m(p_i) = k(p_i, p_i) = 0$ on average.

4 METHOD

We propose discretizing g and c in Eqs. (9) and (10) using neural networks parametrized by weights θ and ϕ and solving them directly using gradient-based optimization.

4.1 Loss

Given s samples $x_1, \ldots, x_s \in \mathbb{R}^d$ drawn from a uniform distribution of B (see Figure 8), let us define the *Dirichlet mean loss* as

$$\mathcal{L}_D^m(\theta) = \frac{|B|}{s} \sum_{i=1}^{s} \|\vec{\mu}(x_i) - \nabla g_\theta(x_i)\|^2 \tag{12}$$

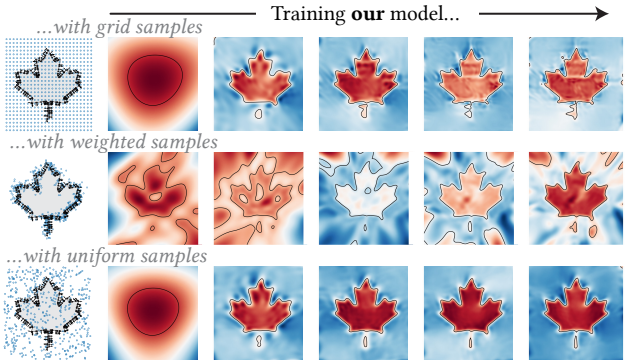


Figure 8: We choose to draw samples uniformly from a bounding box around the point cloud after observing overfitting when using different strategies.

and its covariance counterpart

$$\mathcal{L}_D^k(\phi) = \frac{|B|}{s} \sum_{i=1}^{s} \sum_{j=1}^{s} \|\Sigma(x_i, x_j) - \mathbf{D}c_\phi(x_i, x_j)\|_F^2 . \tag{13}$$

By Monte Carlo integration, we have

$$\mathcal{L}_D^m(\theta) \approx \int_B \|\vec{\mu}(x) - \nabla g_\theta(x)\|^2 \, \mathrm{d}x \tag{14}$$

and

$$\mathcal{L}_D^k(\phi) \approx \iint_B \|\Sigma(x_1, x_2) - \mathbf{D}c_\phi(x_1, x_2)\|_F^2 \, \mathrm{d}x_1 \, \mathrm{d}x_2 . \tag{15}$$

Thus, the functions $g_{\theta^\star}$ and $c_{\phi^\star}$ parametrized by the minimizers

$$\{\theta^\star, \phi^\star\} = \underset{\theta,\phi}{\operatorname{argmin}} \, \mathcal{L}_D^m(\theta) + \mathcal{L}_D^k(\phi) \tag{16}$$

are solutions to the variational problem in Eqs. (9) and (10) when restricted to the space of neural-network-parametrized functions. Thus, they are also Poisson solutions.

It should be noted that, if one substitutes the samples x_i with the points in the input point cloud p_i in Eq. (12), $\mathcal{L}_D^m(\theta)$ is identical to the loss proposed by Dai and Nießner [2022]. However, our decoupling of the sampling from the point cloud is critical. Importantly, it is only by sampling from the volumetric bounding box in Eq. (12) that we can claim to be approximating the volumetric integral in Eq. (14) and thus solving a Poisson equation. Theoretically, this choice has the effect of making our algorithm into a strict generalization of PSR (see Figure 4); in practice, it imposes a volumetric smoothness prior that avoids overfitting (see Figure 3).

An immediate benefit of this neural perspective is the possibility to extend the statistical formalism of Sellán and Jacobson [2022] from the original Poisson Surface Reconstruction [Kazhdan et al. 2006] to its improved, screened version [Kazhdan and Hoppe 2013]. We can do so merely by adding mean and covariance *screen losses*

$$\mathcal{L}_S^m(\theta) = \frac{1}{n} \sum_{i=1}^{n} \|g_\theta(p_i)\|^2 , \quad \mathcal{L}_S^k(\phi) = \frac{1}{n} \sum_{i=1}^{n} \|c_\phi(p_i, p_i)\|^2 , \tag{17}$$

which we combine with the Dirichlet losses to reach our total loss

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_D^m(\theta) + \mathcal{L}_D^k(\phi) + \lambda_S \mathcal{L}_S^m(\theta) + \lambda_S \mathcal{L}_S^k(\phi) \tag{18}$$

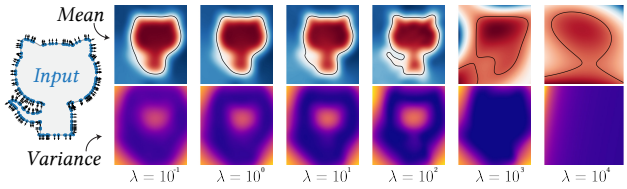


Figure 9: Our screen weight balances smoothness with input fidelity, but can complicate convergence at high values.

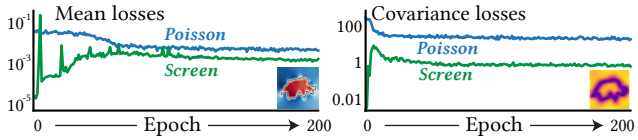


Figure 10: For our choice of hyperparameters, the Poisson losses regularly dominate over the screening terms.

Inspired by the choice made by Dai and Nießner [2022], which we validate experimentally (see Figure 9), we fix $\lambda_S = 100$.

4.2 Data generation

To evaluate $\mathcal{L}(\theta, \phi)$, we first choose B to be a loose box around the input point cloud and uniformly sample $x_1, \ldots, x_s \in B$. Then, as described by Sellán and Jacobson [2022], we compute the matrices

$$\mathbf{K}_1 = (F(x_i, x_j))_{i,j} \in \mathbb{R}^{s \times s} , \quad \mathbf{K}_2 = (F(x_i, p_j))_{i,j} \in \mathbb{R}^{s \times n} , \tag{19}$$

as well as the lumped sample covariance matrix

$$\mathbf{D} \approx \mathbf{K}_3 = (F(p_i, p_j))_{i,j} \in \mathbb{R}^{n \times n} . \tag{20}$$

We employ the same approximated Gaussian kernel suggested by the authors and make use of its compact support to efficiently evaluate the above matrices with a KD tree. Using these matrices, we compute the Gaussian Process posterior mean

$$\boldsymbol{\mu} = \mathbf{K}_2 \mathbf{D}^{-1} \mathbf{N} , \tag{21}$$

where $\mathbf{N} \in \mathbb{R}^{n \times d}$ concatenates $\vec{n}_1, \ldots, \vec{n}_n$, and the covariance

$$\boldsymbol{\Sigma} = \mathbf{K}_1 - \mathbf{K}_2 \mathbf{D}^{-1} \mathbf{K}_2^\top . \tag{22}$$

The row entries in $\boldsymbol{\mu}$ then correspond to $\vec{\mu}(x_i)$, while each scalar entry in $\boldsymbol{\Sigma}$ determines the $d \times d$ matrix Σ through $\Sigma(x_i, x_j) = \boldsymbol{\Sigma}_{i,j} \mathbf{I}$. As we validate experimentally in Figure 8, sampling B uniformly during training is necessary to maintain the theoretical guarantees in Eqs. 14 and 15. More elaborate strategies beyond this work's scope (e.g., Metropolis-Hastings integration) that would result in weights being added in Eqs. 14 and 15 may yield performance improvements.

4.3 Architecture & Training

We model g_θ and c_ϕ using two five-layered MLPs (see Figure 11) with 512 internal hidden units and sine activation functions [Sitzmann et al. 2020]. Our covariance network c_ϕ also includes a SoftPlus layer to enforce positivity, followed by an averaging $(c_\phi(x_1, x_2) + c_\phi(x_2, x_1))/2$ (see Figure 2). Combined with Schwarz's theorem, this forces $\mathbf{D}c(x_1, x_2)$ in Eq. (15) to be symmetric by construction. We

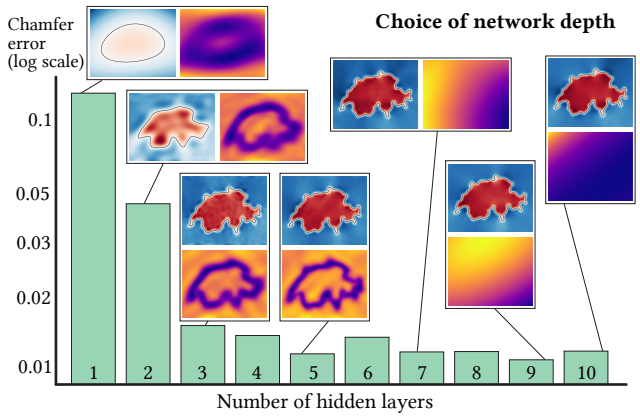


Figure 11: Too few hidden layers can limit the geometric detailed captured un our reconstructions. At the same time, we observe diminishing returns and difficuly with covariance convergence for higher layer numbers.

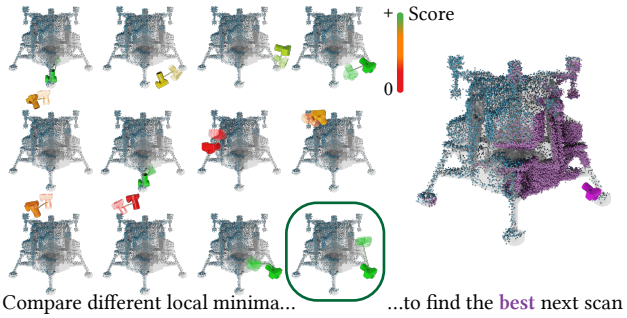


Figure 12: Our local next-best-view search is best combined with a global search, where the scores of different local optima are compared.

experimented with residual connection layers as suggested by Yu et al. [2018], but found no significant performance improvement.

At each epoch, we generate 100,000 covariance and 100,000 mean Poisson samples x_i together with an equal number of screening samples selected from the point cloud p_i (with repetition if necessary) as detailed in Section 4.2. This sampling results in four datasets (covariance, mean, covariance screening and mean screening). We cycle through all four with repetition until they are all exhausted with a 512 batch size, evaluating our losses and backpropagating through them to compute the gradient of $\mathcal{L}(\theta, \phi)$ with respect to (θ, ϕ). We then use the Adam [Kingma and Ba 2014] optimizer with learning rate 10^{-4} and weight decay 10^{-5}. We repeat this process for a number of epochs between 50 and 200 (see Figure 10).

Implementation details. We implement our algorithm in Python, using PyTorch to build and train our model and Gptoolbox [Sellán et al. 2023] for common geometry processing subroutines. In our 3.0Ghz 18-core Linux machine with a 48 GB NVIDIA RTX A6000 graphics card and 528 GB RAM, our unoptimized implementation lasts around 30 seconds to train each epoch, the main bottleneck being the backpropagation through the $\mathbf{D}$ operator in Eq. (11). For Figures 3 and 4, we implemented the algorithm by Dai and Nießner [2022] following their instructions in the absence of author-provided code. We rendered our 3D results using Blender.

5 RESULTS & APPLICATIONS

5.1 3D Scanning integration

Once a point cloud has been captured, our method can be used to compute all kinds of statistical queries useful to the reconstruction (Figure 7) in the same way as described by Sellán and Jacobson [2022]. However, our novel neural perspective goes one qualitative step further and allows for a full integration into the scanning process, providing feedback over where to scan next and efficiently updating a given reconstruction upon capturing more data.

5.1.1 Ray casting. Given a captured scan and a proposed sensor position $\mathbf{r}$ and direction $\mathbf{d}$, a crucial question is where a ray travelling from the sensor would intersect the surface. In traditional volumetric rendering terms, this amounts to computing the *opacity* along the ray, or the likelihood that a ray emanating from the sensor reaches a given distance without terminating.

Sellán and Jacobson [2022] suggest computing the marginal probabilities along the ray

$$p(t) = P(f(\mathbf{r} + t\mathbf{d}) \leq 0) \tag{23}$$

and interpreting these as densities

$$\rho(t) = \frac{p(t)}{1 - p(t)} \tag{24}$$

that they propose integrating to compute the opacity

$$o(t) = 1 - e^{-\int_0^t \rho(\tau)\, \mathrm{d}\tau}. \tag{25}$$

However, we note that this expression for the opacity is usually employed in the context of gases, for which the effects of inter-particle interactions are negligible and one can assume that the likelihood of encountering a gas particle at time τ is independent of encountering one at time $\tau + \mathrm{d}\tau$, giving validity to the integral in Eq. (25). This independence assumption does not hold for the case of uncertain solids, as evidenced by Figure 13: while the marginal likelihood is $p(t) = 0.5$ for all t between t_1 and t_2, there is no configuration of the shape for which a ray terminates at t. Statistically, this is because the point at time t is fully correlated with the point at time t_1. While Figure 13 is an extreme example, this difference appears in general reconstruction examples (see Figure 14)

Accounting for these correlations is simple. Instead of Eq. (25), one can compute the opacity as the joint probability that f was positive at every point in the ray prior to $\mathbf{r} + t\mathbf{d}t$:

$$o(t) = P(f(\mathbf{r} + \tau\mathbf{d}) > 0\,, \forall \tau < t)\,. \tag{26}$$

We uniformly discretize the interval $[0, t]$ such that it amounts to querying a cumulative multivariate Gaussian. Fortunately, as shown by Marmin et al. [2015, Sec. 6], this expression can be differentiated with respect to the entries in the covariance matrix with the aid of Plackett's formula [Berman 1987]. We use the PyTorch implementation of this formula by Marmin [2023] for this task.

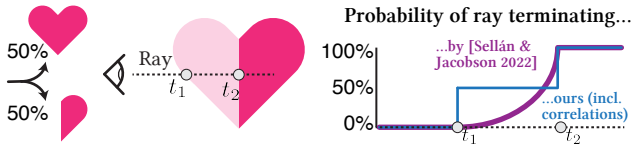


Figure 13: Sellán and Jacobson [2022] consider only marginal likelihoods to compute the termination probability along a ray. This leads to inaccuracies in cases with high correlations among spatial points (see text).

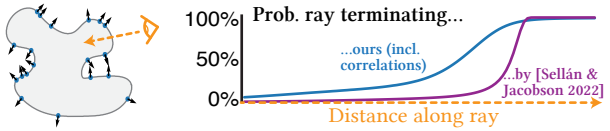


Figure 14: Accounting for correlations leads to significant differences in the ray termination distribution for general point cloud reconstruction examples.

5.1.2 Next view planning. As seen above, the time travelled by a ray from a given camera position before colliding with the surface can be interpreted as a random variable, whose cumulative distribution function is the opacity in Eq. (26). Crucially, by Foubini's theorem, this means one can compute the expected collision time as

$$\langle t(\mathbf{r},\mathbf{d})\rangle = \int_0^\infty (1 - o(\tau))\,\mathrm{d}\tau\,, \tag{27}$$

leading to the expected collision point

$$\mathbf{p}^\star(\mathbf{r},\mathbf{d}) = \mathbf{r} + \langle t(\mathbf{r},\mathbf{d})\rangle\,\mathbf{d}\,. \tag{28}$$

The optimal sensor position will be one that generates a new point cloud point in an area of high variance. Therefore, it makes sense to define the *score* of a camera as

$$u(\mathbf{r},\mathbf{d}) = \sigma(\mathbf{p}^\star(\mathbf{r},\mathbf{d})) = c_{\phi^\star}(\mathbf{p}^\star(\mathbf{r},\mathbf{d}),\mathbf{p}^\star(\mathbf{r},\mathbf{d}))\,. \tag{29}$$

While Sellán and Jacobson [2022] propose a camera scoring criteria, our novel neural perspective allows us to backpropagate through c_ϕ, meaning that we can compute the gradient of the score with respect to camera parameters $(\mathbf{r},\mathbf{d})$, and find an optimal camera position with gradient descent. We show the potential of this contribution in Figure 6, inspired by Fig. 26 by Sellán and Jacobson [2022].

This gradient-based next view angle optimization will often converge to suboptimal local minima. Indeed, as we show in Figure 12, it is better combined with a global search by sampling several initial sensor positions, backpropagating to find an optimum near them, and then choosing the converged camera with the best global score. In Figure 15, we quantify the quality of our subsequent chosen views of a mechanical object by showing they improve on randomly sampled ones. Only in this simplified setup in which views are sampled from a sphere around the object and the directions are constrained to aim to the same spatial point, we are able to compare also to other heuristics like furthest-point sampling, which we show our more generally applicable method matches or outperforms.

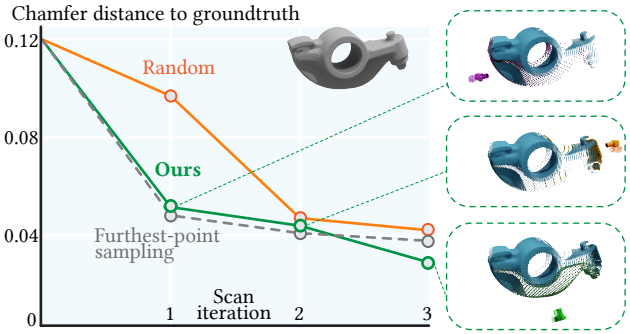


Figure 15: In next-view selection, our algorithm outperforms random sampling and even matches or improves on commonly used heuristics like furthest point in simple setups when the latter are available.

5.1.3 Fine-tuning. Once a new sensor position is chosen and a new scan is taken, points are added to the cloud. Traditional algorithms like PSR would then require investing in an updated discretization and entirely new Poisson solve to obtain an updated reconstruction.

Fortunately, our neural perspective allows us to take advantage of an earlier reconstruction to update it more efficiently. Indeed, as shown in Figure 16, we may consider our model's training on the initial point cloud as a *pretraining* of our model, which is *fine-tuned* for only a few epochs every time new points are captured.

Our model can thus be integrated in an end-to-end scanning pipeline, as once an updated mean and variance is obtained, the best next view angle optimization can start again (see Figure 1). Our algorithm can even provide a stopping criterion, in the form of the integrated uncertainty proposed by Sellán and Jacobson [2022].

5.2 Generalization

Another major advantage of our neural formalism over a traditional one is the possibility to train our network on many given reconstructions and trust it to generalize to similar-yet-unseen data. This can circumvent expensive optimizations in cases where one has access to a large training set of point clouds and must quickly make inference on a newly observed set of points.

We show a prototypical example of what such a process could look like in Figure 17, where a training set of point clouds is captured by scanning a shape from several different angles. Our model is then expanded to accept a latent encoding z, the values of which are trained simultaneously with the model parameters in the "autodecoder" style proposed by Park et al. [2019]. When a new scan $\mathcal{S}$ of the object is captured, test-time optimization (with the model parameters frozen) produces an optimal latent encoding for the new point cloud. This reconstruction can be used as-is or fine-tuned for a very limited number of epochs for a final reconstruction.

We believe this generalization capability can prove useful in industrial applications, where one may be able to produce a number of partial training scans of an object. Then, objects on an assembly line can be quickly scanned and projected into the learned latent space of partial scans. As we show in Figure 18, our model's statistical formalism can then be used (in the form of the point cloud's average log likelihood) to identify foreign objects or defective pieces.

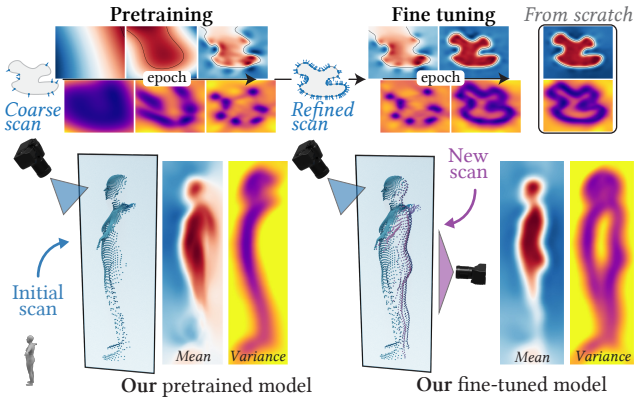


Figure 16: Sequential scanning setups benefit from our model, whose reconstruction can be updated when the object is observed from new angles.

One can also use our model to generalize over a space of different-yet-similar shapes, as we show in Figure 19, where a latent space of scans is learned over 20 diverse human scans generated using STAR [Osman et al. 2020]. Upon capturing a new scan, test-time latent code optimization can efficiently provide a novel reconstruction.

6 LIMITATIONS & CONCLUSION

As we have shown, a key advantage of our neural formulation is the possibility to iteratively fine-tune reconstructions upon capturing more data. To fully take advantage of our method's efficiency, one may need to optimize its runtime, which we did not do beyond asymptotics. We believe the clearest avenues for speedups are exploring non-uniform distributions for data generation and task-specific weight initializations.

We introduce a method for formalizing reconstruction uncertainty using a neural network. However, it should be noted that this uncertainty is encoded by the Gaussian Process used to generate data, while the network is merely solving a PDE. A promising avenue for future work is circumventing the GP altogether, using Machine Learning uncertainty quantification techniques to obtain a posterior distribution directly from the input point cloud. While this may mean deviating from Poisson Surface Reconstruction, it could present a major improvement in accuracy (removing the need for covariance matrix lumping) and applicability (allowing for sensor-specific non-Gaussian noise patterns).

All our generalization results (Figures 17, 18 and 19) use identical (virtual) scanning devices, and every input point cloud is re-scaled to the unit cube; as such, we do not expect our results to generalize beyond these choices. Future work could mitigate this; for example, by learning a latent space of device parameters and positions as suggested by Martin-Brualla et al. [2021] in the context of NeRF.

While uncertainty quantification has become a common consideration in neighboring fields like Computer Vision and Robotics [Kendall and Gal 2017], it remains rare for Computer Graphics works to expose their algorithmic uncertainties. It is our hope that as our tool set grows and our field's application realm diversifies, our work can serve as a first step in the right direction.

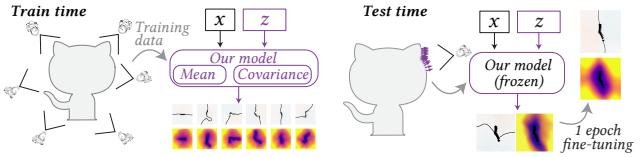


Figure 17: In cases where several scans of an object are available, our model can be combined with autodecoder to efficiently reconstruct new views via test-time optimization.

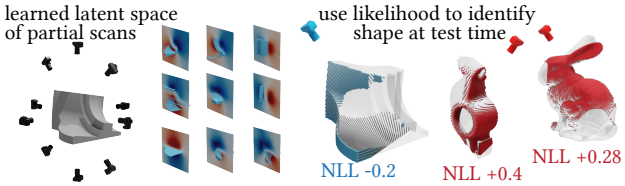


Figure 18: In an industrial setting, one can use our algorithm to learn a latent space of partial scans of an object in order to detect anomalies through any point cloud's negative log likelihood (NLL).

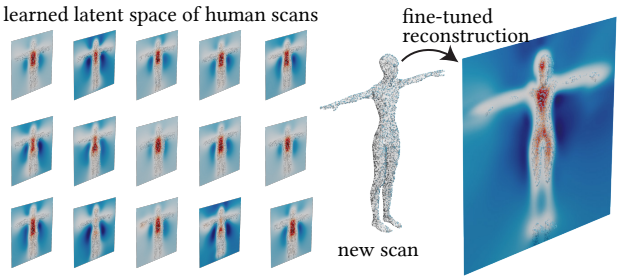


Figure 19: A dataset of similar shapes can be used to learn a latent space of possible scans onto which new scans can be efficiently projected.

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 and the DSI Catalyst Grant program. The first author is funded in part by an NSERC Vanier Scholarship.

We thank Kirill Serkh, Kiriakos Kutulakos, David Lindell, Eitan Grinspun, David I.W. Levin, Oded Stein, Andrea Tagliasacchi, Otman Benchekroun, Lily Goli and Claas A. Voelcker for insightful conversations that inspired us in this work; Hsueh-Ti Derek Liu for his help rendering our results; as well as Rafael Rodrigues (Fig. 6, CC BY-NC-SA 4.0) and ShaggyDude (Fig. 13, CC BY 4.0) for releasing their 3D models for academic use. We would also like to thank Xuan Dam, John Hancock and all the University of Toronto Department of Computer Science research, administrative and maintenance staff.

REFERENCES

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. 2003. Computing and rendering point set surfaces. *IEEE TVCG* 9, 1 (2003), 3–15.

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 *Comput. Graph. Forum*, Vol. 36. Wiley Online Library, 301–329.

Simeon M Berman. 1987. An extension of Plackett's differential equation for the multivariate normal density. *SIAM Journal on Algebraic Discrete Methods* 8, 2 (1987), 196–197.

Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. 2016. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 1462–1468.

Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 67–76.

Shengyong Chen, Youfu Li, and Ngai Ming Kwok. 2011. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research* 30, 11 (2011), 1343–1377.

Cl Connolly. 1985. The determination of next best views. In *Proceedings. 1985 IEEE international conference on robotics and automation*, Vol. 2. IEEE, 432–435.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. 2022. Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing* 92, 3 (2022), 88.

Angela Dai and Matthias Nießner. 2022. Neural Poisson: Indicator Functions for Neural Fields. *arXiv preprint arXiv:2211.14249* (2022).

Jonathan Daudelin and Mark Campbell. 2017. An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects. *IEEE Robotics and Automation Letters* 2, 3 (2017), 1540–1547.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 2018. A papier-mâché approach to learning 3d surface generation. In *Proc. CVPR*. 216–224.

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.

Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. 2016. An information gain formulation for active volumetric 3D reconstruction. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3477–3484.

Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popović. 2023. NeU-NBV: Next Best View Planning Using Uncertainty Estimation in Image-Based Neural Rendering. *arXiv preprint arXiv:2303.01284* (2023).

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proc. SGP*, Vol. 7. 0.

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Trans. Graph.* 32, 3 (2013), 1–13.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Proc. NeurIPS* 30 (2017).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Xin Kong, Shikun Liu, Marwan Taher, and Andrew J Davison. 2023. vMAP: Vectorised Object Mapping for Neural Field SLAM. *arXiv preprint arXiv:2302.01838* (2023).

Sören König and Stefan Gumhold. 2009. Consistent Propagation of Normal Orientations in Point Clouds.. In *VMV*. 83–92.

Mikko Lauri, Joni Pajarinen, Jan Peters, and Simone Frintrop. 2020. Multi-Sensor Next-Best-View Planning as Matroid-Constrained Submodular Maximization. *IEEE Robotics and Automation Letters* 5, 4 (Oct. 2020), 5323–5330. https://doi.org/10.1109/LRA.2020.3007445 arXiv:2007.02084

David Levin. 2004. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*. Springer, 37–49.

Sébastien Marmin. 2023. Torch-MvNorm. https://github.com/SebastienMarmin/torch-mvnorm.

Sébastien Marmin, Clément Chevalier, and David Ginsbourger. 2015. Differentiating the multipoint expected improvement for optimal batch design. In *International Workshop on Machine Learning, Optimization and Big Data*. Springer, 37–48.

Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7210–7219.

Gal Metzer, Rana Hanocka, Denis Zorin, Raja Giryes, Daniele Panozzo, and Daniel Cohen-Or. 2021. Orienting point clouds with dipole propagation. *ACM Trans. Graph.* 40, 4 (2021), 1–14.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. 2005. Multi-level partition of unity implicits. In *Acm Siggraph 2005 Courses*. 173–es.

Ahmed AA Osman, Timo Bolkart, and Michael J Black. 2020. Star: Sparse trained articulated human body regressor. In *Proc. ECCV*. Springer, 598–613.

Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. 2017. A survey of structure from motion*. *Acta Numerica* 26 (2017), 305–364.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*. 165–174.

Mark Pauly, Niloy J Mitra, and Leonidas J Guibas. 2004. Uncertainty and variability in point cloud surface data.. In *PBG*. 77–84.

Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. 2008. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.* (2008), 1–11.

Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. 2020. A survey on LiDAR scanning mechanisms. *Electronics* 9, 5 (2020), 741.

Oussama Remil, Qian Xie, Xingyu Xie, Kai Xu, and Jun Wang. 2017. Surface reconstruction with data-driven exemplar priors. 88 (2017), 31–41.

Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold. 2017. Towards globally optimal normal orientations for large point clouds. In *Comput. Graph. Forum*, Vol. 36. Wiley Online Library, 197–208.

Ruwen Schnabel, Patrick Degener, and Reinhard Klein. 2009. Completion and reconstruction with primitive shapes. In *Comput. Graph. Forum*, Vol. 28. Wiley Online Library, 503–512.

William R Scott, Gerhard Roth, and Jean-François Rivest. 2003. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)* 35, 1 (2003), 64–96.

Silvia Sellán and Alec Jacobson. 2022. Stochastic Poisson Surface Reconstruction. *ACM Trans. Graph.* 41, 6 (2022), 1–12.

Silvia Sellán, Oded Stein, et al. 2023. gptyoolbox: A Python Geometry Processing Toolbox. https://gpytoolbox.org/.

Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. 2007. Interactive topology-aware surface reconstruction. *ACM Trans. Graph.* 26, 3 (2007), 43–es.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*.

Edward J Smith, Michal Drozdzal, Derek Nowrouzezahrai, David Meger, and Adriana Romero-Soriano. 2022. Uncertainty-Driven Active Vision for Implicit Scene Reconstruction. *arXiv preprint arXiv:2210.00978* (2022).

Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. 2021. iMAP: Implicit mapping and positioning in real-time. In *Proc. ICCV*. 6229–6238.

J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. 2014. Volumetric next-best-view planning for 3D object reconstruction with positioning error. *International Journal of Advanced Robotic Systems* 11, 10 (2014), 159.

Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. 2019. Deep geometric prior for surface reconstruction. In *Proc. CVPR*. 10130–10139.

Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. 2015. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Trans. Graph.* 34, 6 (2015), 1–14.

Brian Yamauchi. 1997. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 146–151.

Bing Yu et al. 2018. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics* 6, 1 (2018), 1–12.

Han Zhang, Yucong Yao, Ke Xie, Chi-Wing Fu, Hao Zhang, and Hui Huang. 2021. Continuous aerial path planning for 3D urban scene reconstruction. *ACM Trans. Graph.* 40, 6 (2021), 225–1.

Xiaohui Zhou, Ke Xie, Kai Huang, Yilin Liu, Yang Zhou, Minglun Gong, and Hui Huang. 2020. Offsite aerial path planning for efficient urban scene reconstruction. *ACM Trans. Graph.* 39, 6 (2020), 1–16.