Variational Neural Surfacing of 3D Sketches

YUTAO ZHANG, Université de Montréal, Canada
STEPHANIE WANG, Adobe Research, USA
MIKHAIL BESSMELTSEV, Université de Montréal, Canada

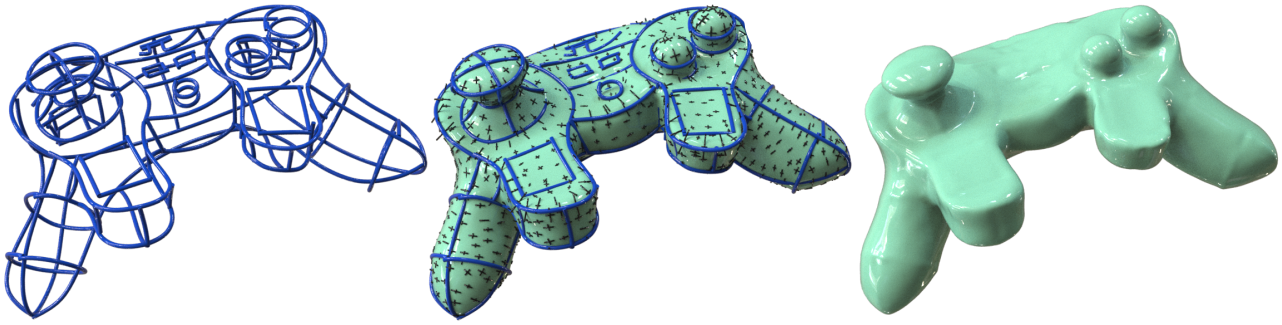


Fig. 1. Given a rough 3D sketch (left, note small gaps that are intended to be connected), our variational optimization reconstructs a 3D surface that aims to capture designer intent (right). One of the core goals of this optimization is to smoothly interpolate principal curvatures guided by the input strokes (center).

3D sketches are an effective representation of a 3D shape, convenient to create via modern Virtual or Augmented Reality (VR/AR) interfaces or from 2D sketches. For 3D sketches drawn by designers, human observers can consistently imagine the surface they imply, yet reconstructing such a surface with modern methods remains an open problem. Existing methods either assume a clean, well-structured 3D curve network (while in reality most 3D sketches are rough and unstructured), or make no effort to produce a surface consistent with perceptual observations. We propose a novel method that addresses this challenge by designing a system that reconstructs a surface that better aligns with human perception from a clean or rough set of 3D sketches. As the topology of the desired surface is unknown, we use an implicit neural surface representation, parameterized via its gradient field.

As suggested by previous perception and modelling literature, human observers tend to imagine the surface by interpreting some of the input strokes as *representative flow-lines*, related to the lines of curvature, and imagining the surface whose curvature agrees with those. Inspired by these observations, we design a novel loss that finds the surface with the smoothest principal curvature field aligned with the input strokes. Together with approximation and piecewise smoothness requirements, we formulate a variational optimization that performs robustly on a wide variety of 3D sketches. We validate our algorithmic choices via a series of qualitative and quantiative evaluations, and comparisons to ground truth surfaces and previous methods.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: 3D sketches, neural implicits, surfacing, flow lines

Authors' Contact Information: Yutao Zhang, Université de Montréal, Montréal, QC, Canada, yutao.zhang@umontreal.ca; Stephanie Wang, Adobe Research, Seattle, WA, USA, evast@g.ucla.edu; Mikhail Bessmeltsev, Université de Montréal, Montréal, QC, Canada, bmpix@iro.umontreal.ca.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/2025/12
https://doi.org/10.1145/3757377.3763900

ACM Reference Format:
Yutao Zhang, Stephanie Wang, and Mikhail Bessmeltsev. 2025. Variational Neural Surfacing of 3D Sketches. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3757377.3763900

1 Introduction

Powered by modern virtual reality (VR) interfaces, 3D sketches are fast to create, intuitive, and expressive. They can capture the essence of a 3D shape, while allowing for more flexibility and requiring less specialized training than standard surface modelling tools. By their design, many 3D sketches are drawn to convey a unique surface to human observers, yet reconstructing that surface is still an unsolved challenge.

One kind of 3D sketch, a *structured curve network*, is a set of clean 3D curves neatly connected at junctions and intersections (*e.g.* , Fig. 2). This type has been the focus of much of the previous research. A common approach to surfacing structured curve networks is to rely on their precise connectivity and to decompose them into loops [Abbasinejad et al. 2011; Zhuang et al. 2013]. Each loop can be then fitted separately by a surface patch [Malraison 2012], possibly enforcing smoothness with adjacent surface patches if necessary. For such curve networks, previous work observed that the input curves are dominated by *representative flow lines*, roughly related to lines of curvature but allowing for artistic license [Bordegoni and Rizzi 2011; Gahan 2010]. Viewers then mentally complete the network of given flow lines, smoothly interpolating the given strokes and following the given principal curvatures [Bessmeltsev et al. 2012; Iarussi et al. 2015; Pan et al. 2015]. Mimicking this perceptual process, the previous work proposed surfacing each loop by minimizing a specialized *flow-alignment energy* that aligns principal curvature field of the surface with a smooth *flow field* — a pair of directions and magnitudes per point — aligned with the input strokes [Pan et al. 2015].

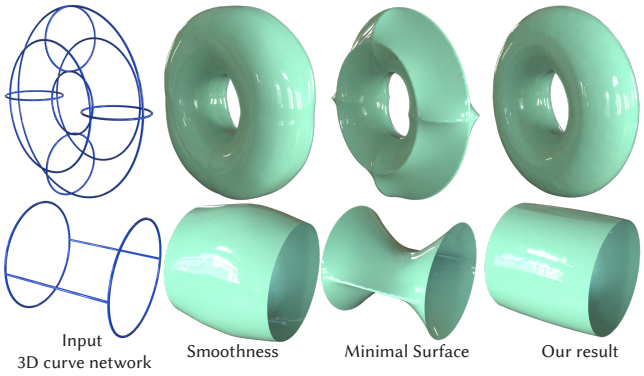


Fig. 2. Compared to more traditional regularizations like surface smoothness or surface area (producing minimal surfaces), our flow alignment energy (right) follows the perceptual observations, making the surface agree better with observer's expectations.

The majority of 3D sketches, however, are not clean and not structured; they contain overdrawing, imprecise junctions, and cannot be easily decomposed into loops (Fig. 1, Fig. 11), making them inaccessible to those methods. While there are systems surfacing those curve networks, for instance by simply converting them into point clouds, they make no attempt to find perceptually valid surfaces. Instead, they focus on particular, purely geometric desiderata like (piecewise) smoothness, minimizing surface area, or developability (Fig. 2). Thus, the problem of surfacing general 3D curve networks, rough or clean, in a manner consistent with designer intent remains an open challenge.

We address this challenge by introducing a novel system that targets both clean and rough 3D sketches in a variational optimization framework. The variational optimization allows our system to be robust to the inaccuracies and inconsistencies in the input sketch, while avoiding converting it into a point cloud that would lose the tangent information. We represent the surface implicitly via a gradient-domain neural network coupled with a high-resolution Poisson equation solver via a Fast Fourier Transform. Using this representation, we reformulate and reparameterize the flow alignment energy, which forms the core component of our system, regularizing surfaces in a manner consistent with designer perception.

Our main technical contributions are:

- a reformulation and reparameterization of the flow-alignment energy to the implicit surface representation and
- a novel system for flow-aligned surfacing of 3D sketches or clean curve networks.

We validate our system on a series of 3D sketches and clean curve networks. We demonstrate that compared to previous works, our surfaces better match designer intent both qualitatively (via visual inspection) and quantitatively (via common metrics).

2 Related Work

2.1 Surfacing 3D Sketches

Progress in surfacing 3D sketches parallels advances in sketching interfaces. Early systems emphasized clean *3D curve networks with explicit connectivity*, created via sketch-based interfaces [Bae et al. 2008; Schmidt et al. 2009] or lifted from 2D strokes [Cordier et al. 2013; Xu et al. 2014]. VR/AR tools such as TiltBrush, GravitySketch, and Quill [Google 2016; GravitySketch 2017; Smoothstep 2021] popularized intuitive 3D sketching but typically yield rough, unstructured curve clouds [Arisoy et al. 2012], due to the lack of hand support and fine motor control [Arora et al. 2017; Machuca et al. 2018]. Systems like CASSIE [Yu et al. 2021] address this by consolidating sketches in real time, at the cost of altering the natural sketching process.

Consequently, the methods of surfacing 3D sketches can be divided into two groups: clean curve networks and rough sketches. For the former, early works surface cycles independently [Abbasinejad et al. 2011; Orbay and Kara 2012; Sadri and Singh 2014; Zhuang et al. 2013], extended by Rasoulzadeh et al. [2023] to imprecise architectural sketches. More recent efforts reconstruct perceptually valid, flow-aligned surfaces [Bessmeltsev et al. 2012; Pan et al. 2015], which we explore in our work (Sec. 3). Most relevant, Pan et al. [2015] initialize with minimal surfaces per cycle and deforms them to align curvature with the induced flow field. Inspired by this, we adapt their energy to neural implicit surfaces, enabling reconstruction from rough sketches without stroke connectivity.

Our focus is surfacing both clean and rough 3D sketches. A pioneering work [Arisoy et al. 2012] reconstructs a surface via a deformation of an initial sphere mesh to match a gradient vector field estimated from the input sketch. Guided by similar intuition, Sureshkumar et al. [2025] deform a surface via solving a series of biharmonic surface interpolations. Exploring similar ideas, a recent work [Yu et al. 2022] proposes a complementary approach to ours that deforms a given mesh template, either created manually or via point cloud reconstruction methods, to fit the 3D sketch, focusing on creating sharp ridges. Our method can be used to produce this initial template of a better quality than previous methods. We show an example of applying their deformation method to our results in Fig. 8.

The work of Stanko et al. [2016] reconstructs surfaces when the normals are also known. Normals are usually non-trivial to obtain. A related input modality, *VR ribbons*, uses the VR controller's orientation to obtain normals. In general, drawing ribbons can be physically challenging for the wrists, but this can be somewhat alleviated given a convenient, albeit non-standard, VR interface [Rosales et al. 2021]. We focus on a simpler and more popular input modality, 3D sketches, which are composed of 3D strokes with no known normals. Our system supports both structured 'clean' 3D curve networks and unstructured rough 3D sketches equally well (Sec. 6).

2.2 Surfacing Point Clouds

One way to surface 3D sketches is to ignore the stroke information and simply convert them into a point cloud by sampling the strokes. Our method is inspired by the progress in this area despite leveraging the stroke information. A full survey of numerous works in point cloud reconstruction is outside our scope, please see surveys [Berger et al. 2017; Huang et al. 2024].

Learning–free Methods. Compared to typical point clouds, 3D sketches are extremely sparse (Fig. 10). As discussed in Yu et al.

[2022], many point cloud reconstruction methods assume dense sampling, so most of these methods fail on our input data [Carr et al. 2001; Fleishman et al. 2005; Hoppe et al. 1992; Kazhdan et al. 2020, 2006; Kazhdan and Hoppe 2013]. Many of those methods additionally require normals, which are hard to estimate for 3D sketches. Among earlier surfacing methods, some only support filling in small holes [Sorkine-Hornung and Kobbelt 2006] or use primitive fitting to complete missing parts of the point clouds [Schnabel et al. 2009; Tagliasacchi et al. 2009].

Our method is partially inspired by VIPSS [Huang et al. 2019], an important exception to this rule. Their method finds an implicit surface that minimizes surface smoothness expressed as Duchon's energy, while simultaneously enforcing unit gradient on the input points. Because of this crucial constraint related to the *Eikonal equation*, their method works even on such sparse inputs as 3D sketches. Their choice of basis and energy, however, often leads to overly smooth results (Sec. 6).

A few other works can support sparse point clouds, including sampled 3D sketches. Hou et al. [2022] iteratively performs Poisson surface reconstruction and updates the point cloud normals. In a related work, Lin et al. [2022] express an indicator function via the Gauss formula, and solve for linearized surface elements associated with the input points. Xu et al. [2023] orients the normals of a point cloud such that the generalized winding number field becomes as close as possible to binary-valued 0-1. In contrast to those methods, our surface reconstruction uses the flow alignment energy to mimic the perceptual process of surfacing 3D sketches. We compare with those methods in Sec. 6.

Learning-Based Methods. A large volume of previous literature learns geometric priors for point cloud reconstruction in a supervised fashion [Chabra et al. 2020; Erler et al. 2020; Huang et al. 2022; Jiang et al. 2020; Lin et al. 2023; Ma et al. 2022; Mescheder et al. 2018; Park et al. 2019; Tang et al. 2021; Wang et al. 2022]. Unfortunately, 3D sketches do not have a sizable ground truth dataset with corresponding surfaces, so supervised learning is not possible. Generating such a dataset from existing shape datasets is challenging, as the modern techniques to extract a curve network from a 3D surface do not capture the stylistic range of real designer 3D sketches [Gori et al. 2017].

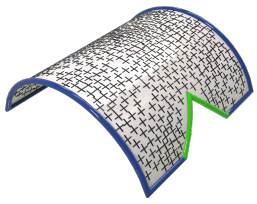
Self-supervised learning for point cloud reconstruction [Atzmon and Lipman 2019, 2020; Li et al. 2023] is more relevant to our work. Atzmon and Lipman [2019; 2020] use neural network to learn a signed distance function to the surface from an unsigned distance to the point cloud via a sign-agnostic loss. Li et al. [2023] incorporates neural gradients along with the neural function itself into the loss. Their method promotes local planarity, which is not the right prior for generic 3D sketches, so we use flow field alignment instead. Gropp et al. [2020] learns an implicit geometric prior by minimizing an Eikonal term on the input point cloud. We use a similar regularizer. A known issue with Eikonal equation is that its solutions are not unique, leading to problematic convergence. An interesting approach to address this issue is a p-Poisson equation-based regularization and splitting the implicit function and its gradient into separate variables [Park et al. 2023]. Alternatively, one can further regularize the learned signed distance field via a 'pull' reprojection loss [Ma et al. 2020]. Another approach is to observe that gradients of signed distance fields tend to have low divergence almost everywhere, which motivates using a divergence-minimizing term as a regularizer [Ben-Shabat et al. 2021]. Some other works use more targeted priors for developable surfaces by minimizing either the absolute value of Gaussian curvature [Dong et al. 2024b] or the rank of the Hessian of the implicit surface [Wang et al. 2023]. Even though piecewise developable surfaces are a mainstay in specific domains, such as machine-made shapes, not all sketched surfaces are intended to be developable (Fig. 2, Fig. 11). Instead, we use a prior of flow-aligned surfaces, specifically designed for curve networks (Sec. 3).

Some works extend these approaches to shapes with boundaries by learning an unsigned distance field [Tian et al. 2024; Zhou et al. 2022], after which the surface extraction process becomes more challenging. Unlike point clouds with unknown surface boundaries *a priori*, in 3D sketches, boundaries are delineated by one or a few strokes that are trivial to annotate [Yu et al. 2022]. Therefore, following previous work [Huang et al. 2019], we prefer working with a signed implicit function, and extracting the surface via marching cubes followed by cutting along those strokes (Suppl. Sec.0.1.2).

3 Background and Observations

Mathematically, the problem of surfacing a given 3D sketch, i.e., set of 3D strokes, is ill-posed, as there are infinitely many surfaces interpolating or approximating the input. In contrast, a 3D sketch drawn by a designer often conveys a unique surface to a human observer. To explain this phenomenon, previous work has provided insight into how designers create 3D sketches, which strokes comprise a sketch, and how human observers interpret sketches; these insights inform our algorithmic decisions.

In the context of clean structured 3D curve networks, design literature, as well as previous perceptual and geometric modeling research [Bessmeltsev et al. 2012; Bordegoni and Rizzi 2011] suggests that many designer-drawn strokes can be classified into two types: *flow-lines* (inset, blue), which represent lines of curvature of the intended surface, and *trim lines*, which outline surface boundaries or sharp features (inset, green). The viewers then leverage



this descriptive representation of the surface [Xu et al. 2014] and mentally interpolate the given strokes according to the stroke types, imagining the final surface [Pan et al. 2015].

Pan et al. [2015] formalise this surfacing insight in the following way. They classify input strokes into flow lines and trim curves; the former control the surface curvature, the latter are only expected to lie on the surface. Starting from an initial mesh that bounds the input curves, they iterate between computing a cross field (on that mesh) that aligns with flow lines and deforming the mesh to minimize the

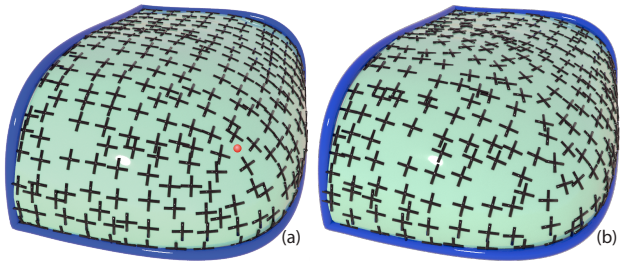


Fig. 3. (a) The smoothest flow field aligned with the input boundary strokes. (b) A surface with its principal curvature directions (note they are different from the flow field in (a)). Previous studies have suggested that the surface (b) with princial curvature directions as close as possible to the smoothest flow field (a) is consistent with artist intent on clean curve networks. We follow this observation, applying and adapting it to the context of rough 3D sketches with no adjacency information, for a neural implicit representation. Note the necessary singularity for a smooth flow field (red vertex in (a)).

difference between its principal curvatures and the cross field. The final result is a surface whose principal curvature directions align with the identified flow lines.

Concretely, for an arbitrary surface containing these strokes, the flow lines Γ° give rise to a *flow field* — a smooth cross field with associated (signed) magnitudes. More specifically, the flow field is

$$\begin{aligned} \mathbf{U} &= (\mathbf{u}_1, \mathbf{u}_2) : \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}^3, \\ \boldsymbol{\lambda} &= (\lambda_1, \lambda_2) : \mathbb{R}^3 \to \mathbb{R} \times \mathbb{R}, \end{aligned} \tag{1}$$

where $\mathbf{u}_1(\mathbf{x})$ and $\mathbf{u}_2(\mathbf{x})$ are orthogonal unit vectors and $\lambda_1(\mathbf{x}), \lambda_2(\mathbf{x})$ are curvature descriptors. On and near the curves Γ°, one of the pair $(\mathbf{u}_1, \lambda_1)$, $(\mathbf{u}_2, \lambda_2)$ is provided by the curve tangent $\mathbf{T}(\mathbf{x})$ and curvature $\kappa(\mathbf{x})$, thus *aligning* the flow field to the input flow lines Γ°. Away from the input strokes, they expect the flow field to be as smooth as possible (Fig. 3).

The *second fundamental form* of a surface S is the bilinear form II_S associated with the differential of the normal:

$$\mathrm{II}_S(\mathbf{v}, \mathbf{w}) = \langle d_{\mathbf{v}}\mathbf{n}, \mathbf{w} \rangle. \tag{2}$$

The eigenvalues of the second fundamental form are the principal curvatures κ_1, κ_2, and the corresponding eigenvectors $\mathbf{p}_1, \mathbf{p}_2$ are the principal curvature directions; this decomposition encodes an orthonormal basis of the tangent plane along which the normal curvature is maximized / minimized. Pan et al. [2015] align the flow field with the principal curvatures by minimizing

$$\min_{S,\mathbf{U},\lambda} \int_S ||II(\boldsymbol{\lambda}, \mathbf{U}) - \mathrm{II}_S||^2 dx + \mathcal{L}_{\text{smooth}}(\mathbf{U}, \boldsymbol{\lambda}) + \mathcal{L}_{\text{align}}(\mathbf{U}, \boldsymbol{\lambda}), \tag{3}$$

where the two last terms are flow field smoothness and alignment to the input strokes, and

$$II(\boldsymbol{\lambda}, \mathbf{U}) = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix}.$$

Pan et al. [2015] iteratively solve for the smoothest flow field for a given surface (Fig. 3a) and then update the surface to match the flow field (Fig. 3b). Their strategy can be seen as coordinate descent, alternating between minimizing two last terms in Eq. 3 for a fixed S and minimizing the first term holding the $\mathbf{U}, \lambda$ fixed.

Observations. In contrast to the clean curve networks addressed by Pan et al. [2015], rough 3D sketches, the focus of this work are more complex in that they contain gaps, broken or overdrawn curves, and imprecisely drawn junctions and intersections (Fig. 11). We observe, however, that design 3D sketches are also dominated by flow lines. Thus, we expect the viewers to complete the surface in a similar manner with clean curve networks. In general, 3D sketches may also contain *construction lines* that do not control the surface shape and are needed for alignment [Gryaditskaya et al. 2019]; we assume those are explicitly labelled and ignored by our pipeline.

Our input is a set of 3D sketch strokes $\Gamma = \{\gamma_0, \cdots, \gamma_N\}$, where each γ_i is a curve in $\mathbb{R}^3$ with or without end points; some strokes Γ^∂ might be annotated as the desired boundary of the surface. We assume all geometry is scaled to fit the unit cube $\Omega = [0, 1]^3$.

We formulate the following requirements for the target surface:

- **Stroke approximation.** We expect the final surface to be as close as possible to the input strokes where there is no ambiguity, approximating rough and overdrawn strokes on average, not necessarily interpolating them. This applies to both flow lines and trim curves equally.
- **Flow-alignment**. We expect the surface's principal curvatures to be aligned with the smooth flow-field induced by the flow line strokes. We expect the magnitudes of principal curvatures to also change smoothly over the surface, including across the input strokes.
- **Piecewise smoothness.** Natural shapes are usually smooth, while machine-made shapes, are usually piecewise smooth. We target both of these common shape types in our system. We furthermore follow the observation that designer-created sketches are usually *descriptive* [Xu et al. 2014], meaning that they contain all the necessary features for a human observer to imagine the shape. As a result, we do not expect any discontinuities to appear away from the drawn strokes. We therefore require, subject to the first two requirements, the final surface so be at least piecewise smooth, with sharper creases possible only at some input strokes.

These requirements guide our algorithmic choices, including the representation, losses, and the optimization strategy, as described in the following section.

4 Method

We first describe our surface representation in Sec. 4.1, then formulate a flow alignment energy for our representation in Sec. 4.2. We first describe our variational optimization for the scalar function Sec. 4.3, then reformulate it in the gradient domain Sec. 4.4. The discretized optimization and details of the gradient based optimization are described in Sec. 5.

4.1 Implicit Surface Representation

Since the topology of our target surface is unknown *a priori*, we opt for an implicit representation of our surface $S = \{x \in \Omega : f(x) = 0\}$. Here $f : \Omega \to \mathbb{R}$ is a scalar field, and $\Omega = [0, 1]^3$ denotes our chosen computational domain, a unit cube. While an isosurface

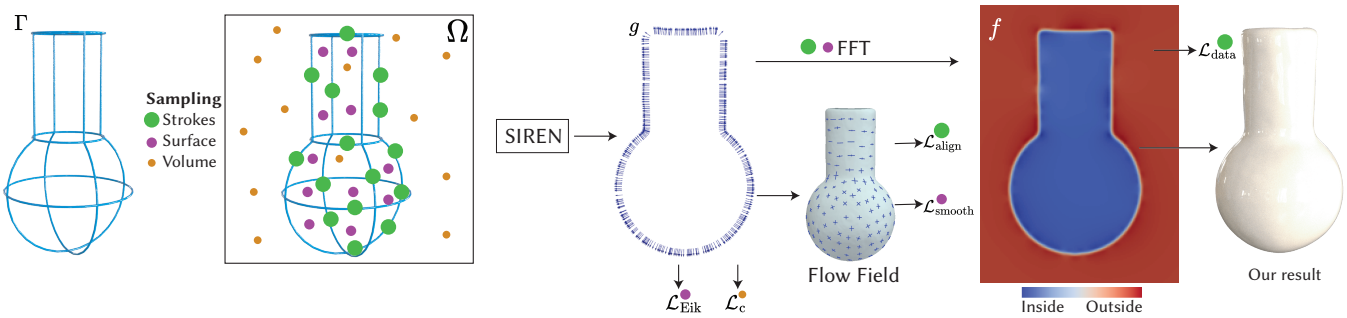


Fig. 4. Given a 3D sketch Γ, our neural network outputs a gradient vector field g of an implicit function f representing the surface, which we reconstruct by integrating the gradient field via Fast Fourier Transform (FFT). We train this network in unsupervised manner via variational optimization, which performs Monte-Carlo integration sampling points on the input strokes (large green disks), the reconstructed surface (medium purple disks), and in the volume of the computational domain Ω (small orange disks). The samples involved in each loss and FFT reconstruction are denoted with those disks.

of a smooth function f is always closed, if some input strokes are annotated as a boundary, we cut the surface along that boundary as a post-processing step (Supplementary Sec.0.1.2).

We parameterize f via a neural network, which is later trained via Adam [Kingma and Ba 2014]. For simplicity of exposition, we first describe the variational optimization for the scalar field f directly (Sec. 4.2, 4.3). Later we parameterize f via a gradient field and formulate the final optimization in the gradient domain (Sec. 4.4). The most important requirement stemming from the neural network parameterization is the differentiability of our losses, which we now introduce, starting with the core component — the flow field losses.

4.2 Flow Field

Our goal is to find a surface minimizing the flow-field energy (Eq. 3) for an implicitly represented surface S. Naïvely following the method for meshes [Pan et al. 2015] in our setup would mean extracting a mesh at each iteration and solving for a smoothest flow field in a differentiable manner. It is unclear how to differentiate that optimization, as it involves numerous discrete decisions.

Instead, our main insight is to reparameterize Eq. 3, defining $II(\boldsymbol{\lambda}, \mathbf{U}) \equiv \text{II}_S$. This way, the first term always evaluates to zero, the flow field directions $\mathbf{U}$ and magnitudes λ are no longer free variables; they are, respectively, eigenvectors and eigenvalues of II_S. Note that theoretically this makes our optimization different from the optimization in Pan et al. [2015]: The flow fields they find are not necessarily realizable as a surface, so they find a surface with the closest prinicipal curvature field, effectively 'projecting' the flow field onto some constraint set. In contrast, our optimization variable is the surface itself, albeit represented implicitly, therefore we are always constrained to this set, making our minima theoretically different.

The flow field smoothness and alignment terms can be then formulated directly in terms of the second fundamental form II_S, which in turn can be expressed via the implicit function f. To formalize this insight, denote the normalized gradient of f as $\mathbf{n} = \frac{\nabla f}{|\nabla f|}$, equal to the surface normal for points on the surface. The second fundamental form can be now expressed by differentiating the normal via Eq. 2.

Flow Field Alignment. The alignment of a 2D vector with the flow field can be measured as an alignment with some principal direction. More specifically, given a stroke $\gamma_i : [0, 1] \to \mathbb{R}^3$, we orthogonally project its tangent vector onto the tangent plane $T_{\gamma_i(t)}S$, forming the angle α with the largest principal direction $\mathbf{p_1}$, which is an eigenvector of II_S (Sec. 3). We can now define the alignment energy as

$$\mathcal{L}_{\text{align}} = \frac{1}{|\Gamma|} \int_{\Gamma} (\cos(4\alpha) - 1)^2 dl, \tag{4}$$

where $|\Gamma|$ is the total length of sketch strokes. The 4-symmetry integrand allows to measure alignment with any of the 2 orthogonal principal directions.

Among all input strokes, [Pan et al. 2015] only consider flow lines for principal curvature direction alignment. However, unlike clean curve drawings where classification of input strokes into flow lines vs. trim curves is relatively straightforward, such discrete decisions are challenging for rough sketches. Instead of relying on heuristics, our method treats all strokes the same. Since the majority of the 3D strokes are flow lines (Sec. 3), this decision does not seem to lead to any significant artifacts (e.g. rectangles in Fig.1 are not flow lines).

Flow Field Smoothness. Measuring smoothness of 2D cross fields has been traditionally done via an angle θ and integer period jumps (e.g., $\theta_i = \theta_j + \frac{\pi}{2}k, k \in \mathbb{Z}$), leading to an NP-hard mixed-integer optimization [Bommes et al. 2009]. One of the more recent approaches to expressing 2D cross field smoothness, with its 4-RoSy symmetry, is via identifying each frame with a complex exponential $e^{i4\theta}$ and measuring its smoothness [Ray and Sokolov 2015].

Such representation, however, does not naturally support cross field singularities even when they are necessary for a smooth field (Fig. 3a) [Huang et al. 2011; Ray and Sokolov 2015]. Instead, we propose to allow for singularities and avoid the issue with the umbilical points by considering both the direction and magnitude of the cross frame, following Zhang et al. [2020] and Palmer et al. [2020]. More precisely, we minimize the smoothness of $\text{II}_S(\mathbf{x})$ directly, thus optimizing smoothness of both principal curvatures and principal directions simultaneously. This formulation allows singularities to be naturally represented via a singular II_S. For the loss term, instead of the more standard Dirichlet energy, we found Total Variation

easier to optimize in our setup:

$$\mathcal{L}_{\text{smooth}} = \frac{1}{|S|}\int_S ||\nabla_S \text{II}(\mathbf{x})||_F d\mathbf{x}, \tag{5}$$

where $|S|$ is the surface area, $||\cdot||_F$ is the Frobenius norm and ∇_S is the covariant derivative. Note that we use the covariant derivative (see Eq. 18) instead of the ambient space derivative to constrain our differentiation to the surface tangent space and thus avoid unnecessarily smoothing the surface.

4.3 Variational formulation for scalar field f

We formulate a minimization problem for the implicit function $f : \Omega \to \mathbb{R}^3$ such that its zero isolevelset $S = f^{-1}(0)$ gives us the desirable geometry:

$$\min_{f:\Omega\to\mathbb{R}} \mathcal{L}_{\text{align}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{Eik}}\mathcal{L}_{\text{Eik}} \tag{6}$$

The first two terms are defined in Sec. 4.2. Details about the other individual terms in Eq. 6 are below:

- Data term. As our surface should follow the sketch (Sec. 3), we encourage the zero isolevelset S to pass through or near all strokes Γ via the following loss:

$$\mathcal{L}_{\text{data}}(f) = \frac{1}{|\Gamma|}\int_\Gamma |f(\mathbf{x})| dl. \tag{7}$$

  Although f is not a true distance function, the magnitude of $f(\mathbf{x})$ at $\mathbf{x} \in S$ still serves as a proxy for distance to the surface.
- Eikonal term. The flow field losses and the data term are not sufficient to produce a well-behaved implicit surface, trivially minimizable by $f \equiv 0$. A function with a dense set of critical points (where its gradient vanishes) near its zero isolevelset would also introduce numerical instability when extracting the surface. Many prior works enforce an Eikonal constraint $|\nabla f| \equiv 1$ to obtain a *signed distance function* (SDF). Rather, following Huang et al. [2019], close to the zero isolevel S, we would like our implicit function to locally behave like an SDF, i.e., to have constant non-zero gradient norm, e.g., $|\nabla f| \approx 1$. Instead of making the gradient norm exactly one, in our experiments we found that using a smaller positive constant improves the approximation of small details:

$$\mathcal{L}_{\text{Eik}} = \frac{1}{2|S|}\int_S (||\nabla f(\mathbf{x})|| - \sigma)^2 \, d\mathbf{x}.$$

  Note that while the constant σ affects the local scaling of the gradients, our other terms scale differently with the gradient magnitude, so changing it will not cause simple scaling of the whole gradient field. For instance, the alignment term is magnitude-invariant, while the data term scales linearly. Empirically we set $\sigma = 0.5$.

Instead of explicitly enforcing piecewise smoothness (Sec. 3) by adding regularization terms, we reparameterize our problem (Sec 4.4).

4.4 Neural Gradient Field

Typical neural surface reconstruction methods use a neural network to predict a scalar implicit function f, for example, SDF [Park et al. 2019; Sitzmann et al. 2020]. However, the flow field loss involves computing a third-order derivative of f (Sec. 4.2). This objective is expensive and challenging to optimize, as neural networks tend to struggle with learning smooth high-order derivatives in a stable manner [Krishnapriyan et al. 2021; Wang et al. 2021]. An alternative is to optimize for a cross field separately [Dong et al. 2024a]; then aligning the implicit function with the field becomes more challenging.

To address this problem, instead of predicting the function f itself, we predict a vector field approximating the function's gradient $g(\mathbf{x} : \Theta) \approx \nabla f$, which we refer to as 'gradient field' despite its potential non-integrability. We parameterize g as a neural network with the same architecture as SIREN [Sitzmann et al. 2020]; Θ are the network parameters. Then the flow-aligned energy only involves second-order derivatives of g, making our optimization more stable.

Most of the losses, including flow field losses, can be formulated directly in terms of the vector field ∇f and therefore g. However, the data term requires the implicit function itself, so we need to integrate the vector field to evaluate this term (Sec. 4.4.1). Furthermore, this new gradient field representation needs an extra regularization term to make it approximately integrable (Sec. 4.4.2).

4.4.1 Vector Field Integration. To evaluate the data term and later extract the zero level set, our goal is to find f that minimizes $\int_\Omega ||\nabla f(x) - g(x)||^2 dx$. Under the Euler-Lagrange formulation, the minimum is the solution to the Poisson equation:

$$\Delta f = \nabla \cdot g. \tag{8}$$

The traditional finite element method suffers from scalability issues, as the grid cell count grows cubically with resolution. To address this, we solve the Poisson equation using the Fast Fourier Transform (FFT) [Iserles 2008], enabling high-resolution solutions with lower memory cost. FFT excels when working with periodic functions, so we define the computational domain $\Omega \subset \mathbb{R}^3$ as the bounding box of the 3D sketch padded by 20% in all directions, with opposite faces identified, effectively forming a 3D torus $\mathbb{T}^3$. Let g be the gradient field sampled at grid cell centers. For frequencies $\mathbf{u}$ (we use PyTorch `fftfreq`), we apply multimensional FFT to both sides of the Poisson equation (Eq. 8). Denoting $\hat{g}(\mathbf{u}) = \text{FFT}(g(\mathbf{x}))$ and $\hat{f}(\mathbf{u}) = \text{FFT}(f(\mathbf{x}))$, we get:

$$\text{FFT}(\Delta f) = -4\pi^2||\mathbf{u}||^2\hat{f} \qquad \text{FFT}(\nabla \cdot g) = 2\pi i(\mathbf{u} \cdot \hat{g}). \tag{9}$$

The *unnormalized* solution f on grid is then computed as:

$$f = \text{IFFT}(\hat{f}) = \text{IFFT}\left(\frac{i\mathbf{u} \odot \hat{g}}{-2\pi||\mathbf{u}||^2}\right), \tag{10}$$

where $\odot$ denotes the element-wise product, and trilinearly interpolated to the whole volume. To align the zero iso-level with the input stroke points $\mathbf{x} \in \Gamma$, we apply a scale-and-shift normalization [Peng et al. 2021]:

$$f = \frac{1}{|f(\mathbf{0})|}\left(f - \frac{1}{|\Gamma|}\sum_{\mathbf{x}\in\Gamma} f(\mathbf{x})\right), \tag{11}$$

where $f(\mathbf{0})$ is the value at the corner of the bounding box; we use the same symbol f on both sides for brevity. This normalization ensures that the stroke points take values sufficiently distinct from those at the domain boundaries, helping to better enforce the intended

constraints and avoid trivial harmonic solutions. See Suppl. Sect. 0.2 for more explanation.

4.4.2 Integrability and Curl-free Constraint. While the Poisson equation (Eq. 8) solves the gradient approximation problem in a least-squares sense, this does not guarantee that the reconstructed surface will follow the gradient field, as the gradient field can be non-integrable, i.e., either have a harmonic component or have non-zero curl. We explicitly avoid the harmonic solution through normalization of the integral function f (Eq. 11), so integrability can be enforced by minimizing curl of the computational domain Ω:

$$\mathcal{L}_{\text{c}} = \frac{1}{|\Omega|}\int_{\Omega} ||\nabla \times g(\mathbf{x})||^2 d\mathbf{x}. \tag{12}$$

4.4.3 Gradient-Based Variational Formulation. With these details, we are now ready to formulate our optimization in the gradient domain:

$$\begin{aligned} &\min_{\substack{f:\Omega\to\mathbb{R}\\ g:\Omega\to\mathbb{R}^3}} \mathcal{L}_{\text{align}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{Eik}}\mathcal{L}_{\text{Eik}} + \lambda_{\text{c}}\mathcal{L}_{\text{c}} \\ &\text{s.t.} \quad \Delta f = \nabla \cdot g \end{aligned} \tag{13}$$

We use $\lambda_{\text{smooth}} = 0.01, \lambda_{\text{data}} = 30, \lambda_{\text{Eik}} = 7, \lambda_{\text{curl}} = 7$ to produce all the results in the paper.

5 Discretization and Optimization

With the variational formulation above, we now discretize the necessary operators and integrals, and describe our optimization strategy.

5.1 Discretization

First, we need to express the shape operator $\text{II}_S(x)$ on the surface via the predicted gradient field. As it is a differential of the normal (Eq. 2), which is the normalized gradient of the implicit function, the second fundamental form is related to the Hessian of the function. Since it is an operator working with tangent vectors, we explicitly project the Hessian onto the tangent space:

$$\text{II}_S(\mathbf{x}) = \frac{1}{||\nabla f||}P(\mathbf{x})H(\mathbf{x})P(\mathbf{x}) = \frac{1}{||g(\mathbf{x})||}P(\mathbf{x})\nabla g(\mathbf{x})P(\mathbf{x}), \tag{14}$$

where

$$P(\mathbf{x}) = I - \mathbf{n}(\mathbf{x})\mathbf{n}(\mathbf{x})^T = I - \frac{g(\mathbf{x})g(\mathbf{x})^T}{||g(\mathbf{x})||^2} \tag{15}$$

is the projection operator, a symmetric 3×3 matrix. We convert the second fundamental form to the local surface coordinates, for arbitrary coordinate frame of the tangent space $u, v \in T_S$:

$$\text{II}_{\mathbf{uv}}(\mathbf{x}) = J^T \text{II}_S(\mathbf{x}) J, \tag{16}$$

as in our setup, its integral is invariant to the choice of (orthonormal) basis. Without loss of generality, we choose $J = \begin{bmatrix}\mathbf{u} & \mathbf{v}\end{bmatrix}$ where $\mathbf{u}, \mathbf{v}$ are eigenvectors of $\text{II}_S(\mathbf{x})$.

To implement the alignment term (Eq. 4), we project the curve tangent vector $\gamma'(t)$ onto the tangent plane T_S

$$\gamma'_{\mathbf{uv}}(t) = J^T \gamma'(t), \tag{17}$$

convert it into polar coordinates (R, α) and substitute α into Eq. 4. To implement the smoothness term (Eq. 5), we leverage the fact that the covariant derivative in Eq. 5 is the projection of the Euclidean directional derivative:

$$\nabla_S \text{II}_{\mathbf{uv}}(\mathbf{x}) = P(\mathbf{x})\nabla \text{II}_{\mathbf{uv}}(\mathbf{x}). \tag{18}$$

Discretizing Integrals. To compute the integrals in Eq. 13, we use naïve Monte Carlo integration. We uniformly sample $n_1 = 10k$ points on the input strokes Γ, $n_2 = 15k$ on the isosurface S and $n_3 = 15k$ in Ω. We discretize the integrands on the corresponding samples, and approximate the integrals as the averages of those values. The derivative operator ∇ in all the equations is computed via PyTorch's automatic differentiation.

As the data term $\mathcal{L}_{\text{data}}(f)$ involves solving the Poisson equation (Eq. 8), the quality of the reconstruction depends on the sampling strategy for the right hand side in Eq. 10. Directly evaluating the gradient function g at every grid point is impractical at high resolutions. Instead, since we only need the isosurface, we evaluate g at the set of samples $\Gamma \cup S$, i.e., stroke points and zero isosurface points, and then rasterize those values onto the grid points via trilinear interpolation. Notice that $\Gamma \cup S$ produces a non-uniform sampling, which causes the rasterized g to have larger magnitude near the strokes. This additional magnitude makes the isolevel sets of f cluster more densely around the strokes, which empirically facilitates convergence but may introduce artifacts in the shape near those regions. To mitigate this, we normalize the rasterized g at each grid point before solving the Poisson equation once the optimization has stabilized (see Suppl. Sec. 0.1.1 for more details).

5.2 Optimization Strategy

We use the default SIREN initialization with uniformly distributed weights [Sitzmann et al. 2020]. At each iteration (Fig. 4), we sample point sets Γ, S, and Ω, predict gradients g, solve the Poisson equation via FFT to obtain f, and extract its zero isosurface using marching cubes. We then evaluate the loss terms and update parameters with Adam [Kingma and Ba 2014]. Although sampling is technically non-differentiable, this is not an issue in practice since the surface evolves smoothly between steps. To bootstrap the optimization, we perform an initialization stage where we avoid sampling the surface that is not reliable in the beginning. To that end, we disable the $\mathcal{L}_{\text{smooth}}$ term and only use the sample set Γ for the Eikonal term $\mathcal{L}_{\text{Eik}}$ and sampling of the gradient field for the FFT Poisson solve. Our initialization stage performs 200 steps of Adam, followed by 800 steps of the full optimization. For all our results, we set the grid resolution to 256^3. Please see Suppl. Sec. 0.1.1 for more details.

6 Results and Validation

Qualitative Evaluation. We have automatically generated a number of surfaces from 3D sketches of different styles and levels of abstraction. These include clean curve networks, where each junction is precise (Fig. 10) and rough 3D sketches with gaps drawn via VR interfaces (Fig. 11). Most of our results are closed shapes; we show an example with a boundary in the Supplementary Material (Suppl. Sec. 0.1.2).

Quantitative Evaluation. We have performed quantitative evaluations of our method in two different ways. First, we used the

results of FlowRep [Gori et al. 2017] that takes a quadmesh and produces a 3D curve network representing the shape; while those curve networks are not designer-drawn 3D sketches, their method is designed to replicate artistic practices for creating clean curve networks. We then use their 3D curve networks as inputs and compare our surfaces, as well as the results of other methods, with the ground truth quadmeshes of [Gori et al. 2017]. For this experiment, we set Eikonal σ to 1.0, and cancel the normalization for vector integration. We use point set F-score [Wang et al. 2023] that measures recall and precision by testing whether a point in one sample set is within a certain range (0.01 in our experiments) from the points in the other sample set. The results are presented in Tab. 1 and in Fig. 5. Our results are closer to the ground truth in standard metrics (Chamfer distance and F-score) for all but two of the inputs we tried. Our method's performance is statistically insignificant from the best result (VIPSS) on one of the examples (doghead), and while Li et al. [2023] is closer to the ground truth for the phone input, our method generates better shapes overall (Fig. 5).

Comparison to previous work. For structured curve networks, where the connectivity is precise and all the junctions are exact, the method of Pan et al. [2015] produces nearly perfect results (Fig. 6). Those networks are hard to create in modern interfaces [Yu et al. 2022]: their method does not work for much more commonly used rough 3D sketches (Fig. 11). Our method works for both clean and rough 3D sketches and produces surfaces that are similar in shape due to a similar energy, albeit somewhat smoother due to the inherent smoothness of our gradient integration.

We run our method on a set of clean curve networks (Fig. 10) and rough sketches (Fig. 11) collected from [Huang et al. 2019; Pan et al. 2015; Yu et al. 2022], and compare our results with the previous surface reconstruction methods: VIPSS [Huang et al. 2019], PGR [Lin et al. 2022], IPSR [Hou et al. 2022], NeuralGF [Li et al. 2023], and two normal orientation methods that use [Kazhdan et al. 2006] for surfacing: GCNO [Xu et al. 2023] and WNNC [Lin et al. 2024]. These methods input point clouds, so we resample each sketch stroke with segment length of 0.02 — we found this sampling works best for their methods. Unlike our method that relies on stroke tangents as well as positions, their methods ignore stroke connectivity, only using sampled points. On clean curve networks (Fig. 10), our results are either on par (box) or significantly better (dress, iron, sewing machine, coffee machine). Those sparse clean curve networks are particularly challenging for the methods that reconstruct the surface from normals defined on strokes (GCNO, WNNC, IPSR), since without any normals between the strokes, [Kazhdan et al. 2006] often does not perform well. The VIPSS result for the coffee machine is of similar quality to ours, but the bulging top surface is inconsistent with perception, as discussed in [Pan et al. 2015].

On rough 3D sketches (Fig. 1, 11), our pipeline is often better at reconstructing smaller details and thin features (e.g. gamepad and guitar). For other, denser sketches, our results are on par with the best previous methods, as the ambiguity we address with the flow energies is less for denser sketches.

Ablations and Parameters. We study the effect of each of our loss terms in an ablation study Fig. 9. Without the alignment term, the surface merely passes through the curves but does not align its curvature lines with them. Conversely, when the flow field smoothness is disabled, the principal curvature directions align with the input curves, but the curvature field lacks smooth interpolation across the surface. Without the Eikonal or curl-free terms, the optimization does not produce useful surfaces or does not converge.

Performance. On our NVIDIA GeForce RTX 4080 (16 GiB VRAM) with AMD Ryzen 7 7700X, our Python implementation takes around 10 minutes per input. VIPSS takes between 40 seconds and hours, depending on sampling; WNNC, PGR, and IPSR take a few seconds per input; NeuralGF takes $\approx$ 70 seconds per input; GCNO ranges from 4–150 seconds. Most of our time is spent in the Poisson solver. We can improve the performance by running the solver once every few iterations or adopting a multi-resolution strategy.

Limitation: sharp features. Due to Poisson surface reconstruction intrinsic smoothness and finite grid resolution, sharp features are a challenge for our method. However, our method can be seen as complementary to the method by Yu et al. [2022] that deforms an input template to fit a 3D sketch, forming sharp features at the curves. They use either manually created templates or outputs of VIPSS [Huang et al. 2019]. Our method doesn't require any templates as input, but can be viewed as a method to generate those templates for their method. In Fig. 8 we show that this way we can further improve the quality of the results.

Other limitations and Future Work. For sparse 3D sketches, our method can occasionally misinterpret thin features (Fig. 7), such as the wall of the bathtub. We hypothesize that one of the reasons why it is not ambiguous for human observers is because we recognize the object; our method, in contrast, has no data priors. Perhaps combining our method with a data prior can further disambiguate such sketches.

7 Conclusions

We have presented a novel method for surfacing rough 3D sketches and clean 3D networks that supports arbitrary topology and introduces a flow field alignment and smoothness loss amenable with neural implicit representation. Our method makes a step forward towards fully functional 3D sketch-based modeling software, enabling intuitive modeling in 3D.

Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant No.: RGPIN-2024-04968, the NSERC - Fonds de recherche du Québec - Nature et technologies (FRQNT) NOVA Grant No. 314090, FRQNT team grant No. 361570, and a gift from Adobe.

References

Fatemeh Abbasinejad, Pushkar Joshi, and Nina Amenta. 2011. Surface Patches from Unorganized Space Curves. *Computer Graphics Forum* 30 (2011). https://api.semanticscholar.org/CorpusID:14378260

Erhan Batuhan Arisoy, Günay Orbay, and Levent Burak Kara. 2012. Free Form Surface Skinning of 3D Curve Clouds for Conceptual Shape Design. *J. Comput. Inf. Sci. Eng.* 12 (2012). https://api.semanticscholar.org/CorpusID:12887331

Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Pratap Singh, and George W. Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces

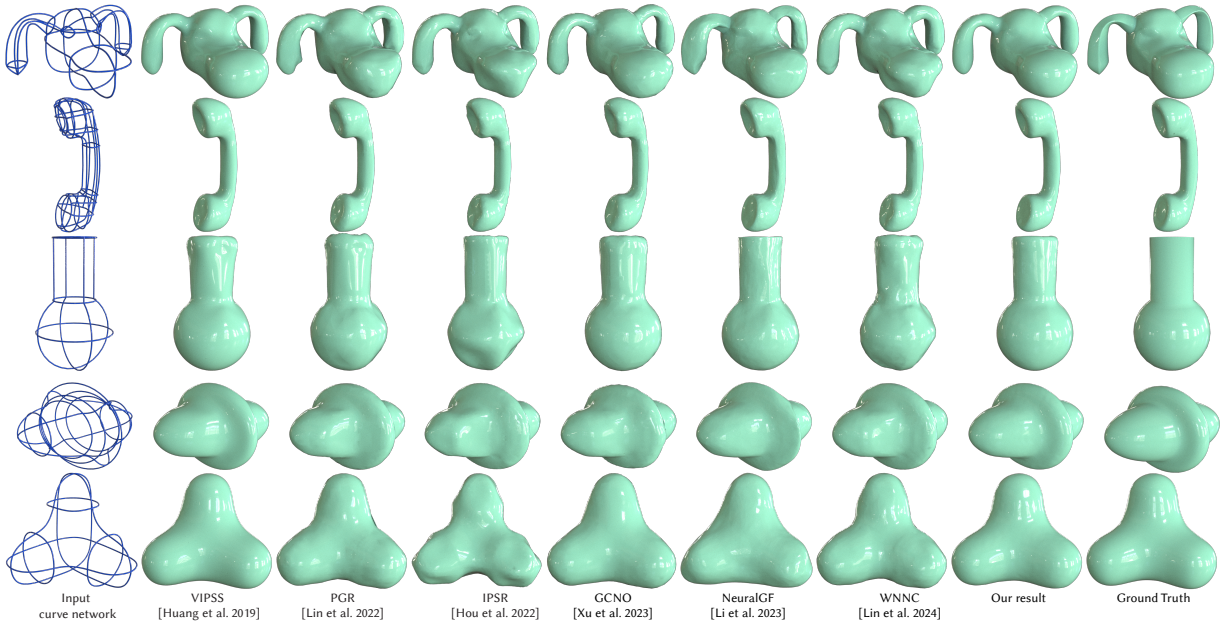


Fig. 5. Comparing our results, prior works, and the ground truth surfaces as provided in [Gori et al. 2017].

Table 1. Quantitative comparison across multiple methods. Lower Chamfer distance ($1e^{-3}$) and higher F-Score (%) are better.

	Mean		Std.		doghead		phone		spherecylinder		ellipsetorus		trebol	
	Chamfer ↓	F-Score ↑	Chamfer	F-Score	Chamfer ↓	F-Score ↑	Chamfer ↓	F-Score ↑	Chamfer ↓	F-Score ↑	Chamfer ↓	F-Score ↑	Chamfer ↓	F-Score ↑
VIPSS	6.21	78.56	2.79	12.85	**7.96**	**68.70**	3.46	92.61	9.50	68.33	6.96	70.55	3.18	92.60
GCNO	8.51	64.91	2.78	13.23	8.63	63.81	5.58	79.07	7.07	74.54	13.00	44.94	8.27	62.21
IPSR	17.18	51.85	8.26	20.30	17.35	50.40	4.55	86.52	22.93	39.91	15.16	47.36	25.91	35.07
WNNC	11.44	61.84	5.47	16.29	15.88	50.57	4.56	83.96	18.04	42.26	8.90	69.68	9.84	62.71
PGR	8.95	70.89	4.12	14.59	14.74	52.62	3.66	91.80	10.33	62.84	9.21	72.90	6.82	74.28
NeuralGF	9.80	67.09	4.08	15.74	12.76	62.17	**2.80**	**95.11**	11.93	57.71	9.71	60.28	11.81	60.19
Ours	**4.55**	**86.06**	**2.19**	**11.90**	7.97	67.91	3.33	93.26	**5.51**	**79.97**	**3.19**	**93.27**	**2.77**	**95.87**

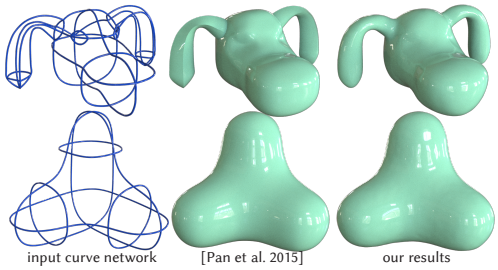


Fig. 6. On clean curve networks, our results are often similar to the specialized method designed for just that kind of input [Pan et al. 2015]. Since other methods generate a surface per loop, their results are less smooth on some strokes (*e.g.* dog's ear).

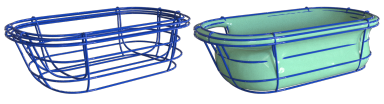


Fig. 7. Thin features, especially on sparse ambiguous input, can be a challenge to our data-free optimization.

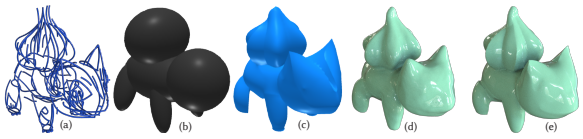


Fig. 8. The method of Yu et al. [2022] deforms a manually designed template (b) to fit the input sketch (a), introducing sharp features (c). Our method is complementary to theirs: using our automatic result (d) as an input to their method, we can introduce sharp ridges into our result (e) without any manual labour to model the template (b).

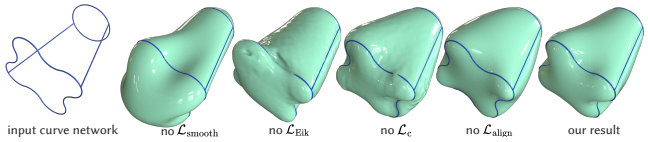


Fig. 9. Ablation study: the effects of disabling each term in our optimization (Eq. 13). By combining all the terms, our result (right) is flow-aligned and smooth.

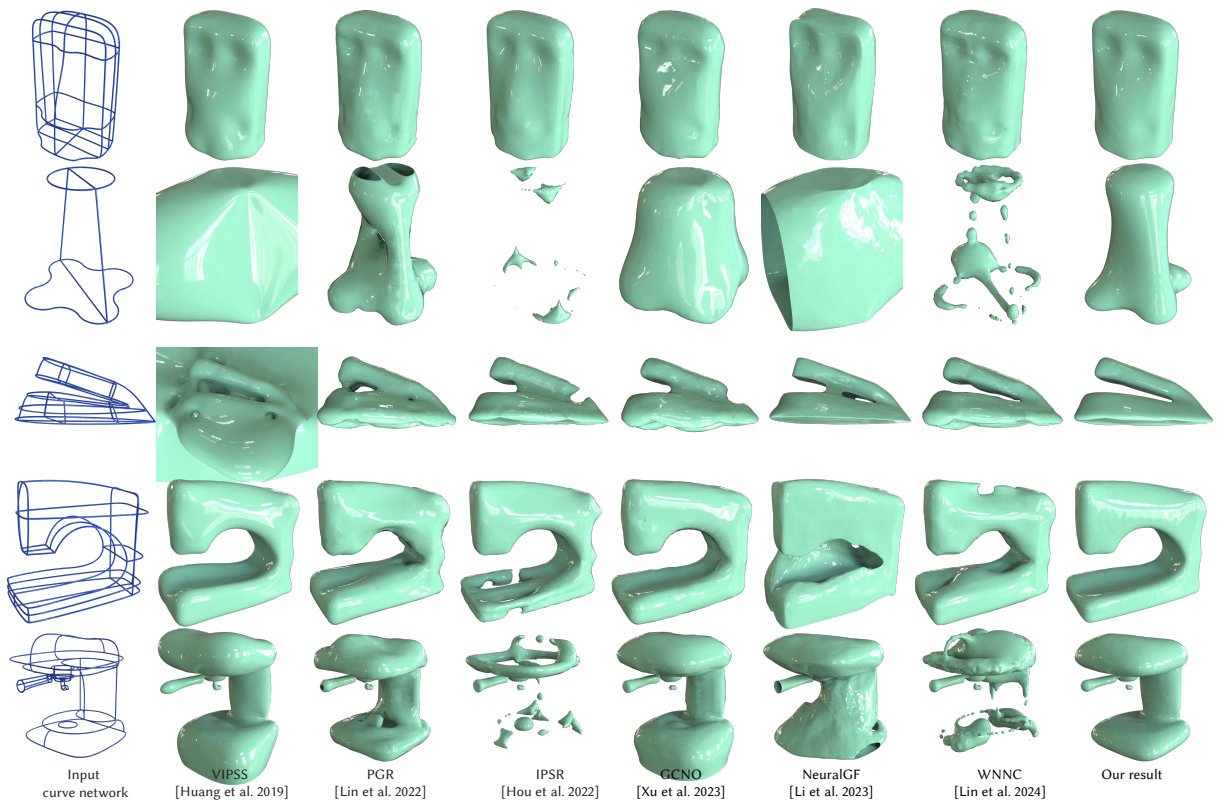


Fig. 10. A comparison of our method with previous works on clean curve networks, top to bottom: 'box', 'dress', 'iron', 'sewing machine', 'coffee machine'. For all methods, we compute the surface for the coffee machine body and handle separately.

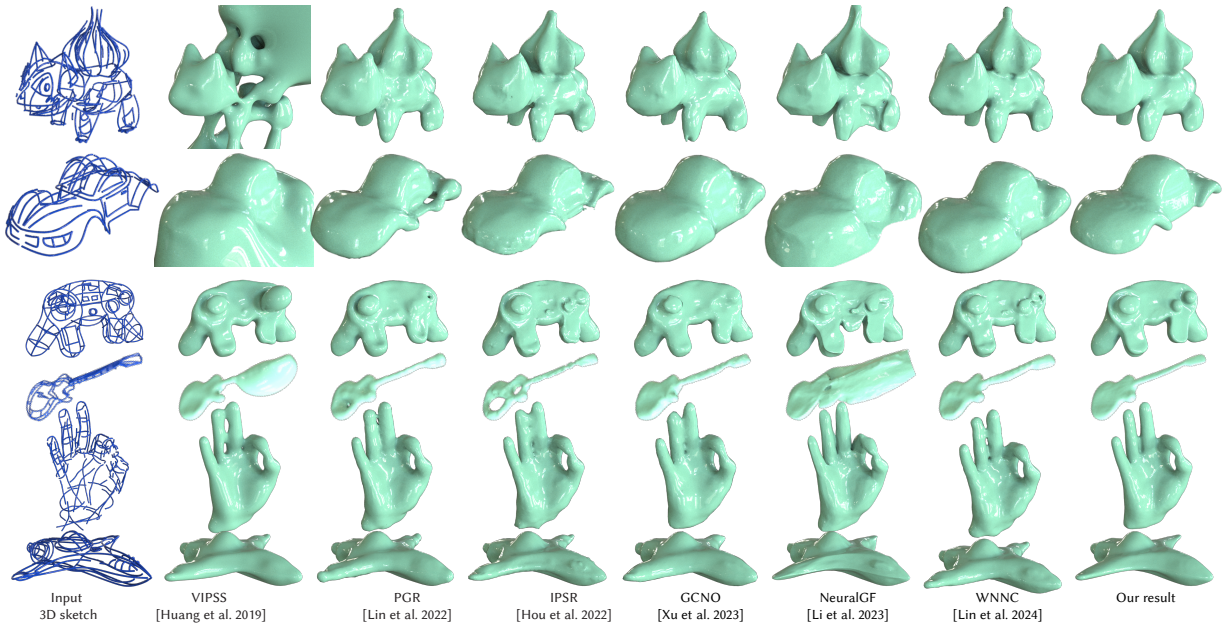


Fig. 11. A comparison of our method with previous works on rough 3D sketches. Top to bottom: 'bulbasaur', 'buggy', 'gamepad', 'guitar', 'ok hand', 'spaceship'

in VR. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017). https://api.semanticscholar.org/CorpusID:10652844

Matan Atzmon and Yaron Lipman. 2019. SAL: Sign Agnostic Learning of Shapes From Raw Data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 2562–2571. https://api.semanticscholar.org/CorpusID:208267630

Matan Atzmon and Yaron Lipman. 2020. SAL++: Sign Agnostic Learning with Derivatives. *ArXiv* abs/2006.05400 (2020). https://api.semanticscholar.org/CorpusID:219558695

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Pratap Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *ACM Symposium on User Interface Software and Technology*. https://api.semanticscholar.org/CorpusID:7294545

Yizhak Ben-Shabat, Chamin Pasidu Hewa Koneputugodage, and Stephen Gould. 2021. DiGS : Divergence guided shape implicit neural representation for unoriented point clouds. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 19301–19310. https://api.semanticscholar.org/CorpusID:235490568

Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Cláudio T. Silva. 2017. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum* 36 (2017). https://api.semanticscholar.org/CorpusID:5902682

Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. 2012. Design-driven quadrangulation of closed 3D curves. *ACM Trans. Graph.* 31, 6, Article 178 (Nov. 2012), 11 pages. doi:10.1145/2366145.2366197

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM SIGGRAPH 2009 papers* (2009). https://api.semanticscholar.org/CorpusID:2936733

M. Bordegoni and C. Rizzi. 2011. *Innovation in Product Design: From CAD to Virtual Prototyping*. Springer. http://books.google.ca/books?id=gXS-7JIJEwEC

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. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001). https://api.semanticscholar.org/CorpusID:11863186

Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, S. Lovegrove, and Richard A. Newcombe. 2020. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. In *European Conference on Computer Vision*. https://api.semanticscholar.org/CorpusID:214623267

Frederic Cordier, Hyewon Seo, Mahmoud Melkemi, and Nickolas S. Sapidis. 2013. Inferring mirror symmetric 3D shapes from sketches. *Computer-Aided Design* 45, 2 (2013), 301–311. doi:10.1016/j.cad.2012.10.013 Solid and Physical Modeling 2012.

Qiujie Dong, Huibiao Wen, Rui Xu, Shuangmin Chen, Jiaran Zhou, Shiqing Xin, Changhe Tu, Taku Komura, and Wenping Wang. 2024a. NeurCross: A Neural Approach to Computing Cross Fields for Quad Mesh Generation. *ACM Trans. Graph.* 44 (2024), 108:1–108:17. https://api.semanticscholar.org/CorpusID:278326833

Qiujie Dong, Rui Xu, Pengfei Wang, Shuangmin Chen, Shiqing Xin, Xiaohong Jia, Wenping Wang, and Changhe Tu. 2024b. NeurCADRecon: Neural Representation for Reconstructing CAD Surfaces by Enforcing Zero Gaussian Curvature. *ACM Trans. Graph.* 43, 4, Article 51 (July 2024), 17 pages. doi:10.1145/3658171

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy Jyoti Mitra, and Michael Wimmer. 2020. Points2Surf Learning Implicit Surfaces from Point Clouds. In *European Conference on Computer Vision*. https://api.semanticscholar.org/CorpusID:226298441

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. 2005. Robust moving least-squares fitting with sharp features. *ACM SIGGRAPH 2005 Papers* (2005). https://api.semanticscholar.org/CorpusID:473208

A. Gahan. 2010. *3D Automotive Modeling: An Insider's Guide to 3D Car Modeling and Design for Games and Film*. Elsevier Science. http://books.google.ca/books?id=_VQS1jTnuWMC

Google. 2016. Tilt Brush. https://github.com/googlevr/tilt-brush

Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. FlowRep: Descriptive Curve Networks for Free-Form Design Shapes. *ACM Transaction on Graphics* 36, 4 (2017). doi:10.1145/3072959.3073639

GravitySketch. 2017. Gravity Sketch | 3D sketching and design software. https://www.gravitysketch.com/

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:211259068

Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Frédo Durand, and Adrien Bousseau. 2019. OpenSketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph.* 38, 6, Article 232 (Nov. 2019), 16 pages. doi:10.1145/3355089.3356533

Hugues Hoppe, Tony DeRose, Tom Duchamp, John Alan McDonald, and Werner Stuetzle. 1992. Surface reconstruction from unorganized points. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992). https://api.semanticscholar.org/CorpusID:1122241

Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. 2022. Iterative poisson surface reconstruction (iPSR) for unoriented points. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 13. https://api.semanticscholar.org/CorpusID:250956751

Jiahui Huang, Haoxiang Chen, and Shihui Hu. 2022. A Neural Galerkin Solver for Accurate Surface Reconstruction. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 16. https://api.semanticscholar.org/CorpusID:254097072

Jin Huang, Y. Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary aligned smooth 3D cross-frame field. *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011). https://api.semanticscholar.org/CorpusID:13935900

Zhiyang Huang, Nathan A. Carr, and Tao Ju. 2019. Variational implicit point set surfaces. *ACM Transactions on Graphics (TOG)* 38 (2019), 1 – 13. https://api.semanticscholar.org/CorpusID:196834922

ZhangJin Huang, Yuxin Wen, ZiHao Wang, Jinjuan Ren, and Kui Jia. 2024. Surface Reconstruction From Point Clouds: A Survey and a Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 12 (2024), 9727–9748. doi:10.1109/TPAMI.2024.3429209

Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields. *ACM Transactions on Graphics (TOG)* 34 (2015), 1 – 16. https://api.semanticscholar.org/CorpusID:18892160

Arieh Iserles. 2008. *A First Course in the Numerical Analysis of Differential Equations* (2 ed.). Cambridge University Press.

Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas A. Funkhouser. 2020. Local Implicit Grid Representations for 3D Scenes. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 6000–6009. https://api.semanticscholar.org/CorpusID:214606025

Misha Kazhdan, Ming Chuang, Szymon Rusinkiewicz, and Hugues Hoppe. 2020. Poisson Surface Reconstruction with Envelope Constraints. *Computer Graphics Forum* 39 (2020). https://api.semanticscholar.org/CorpusID:220116886

Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*. https://api.semanticscholar.org/CorpusID:14224

Michael M. Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Trans. Graph.* 32 (2013), 29:1–29:13. https://api.semanticscholar.org/CorpusID:1371704

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

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. 2021. Characterizing possible failure modes in physics-informed neural networks. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:237386144

Qing Li, Feng, Kanle Shi, Yuedong Gao, Yi Fang, Yu-Shen Liu, and Zhizhong Han. 2023. NeuralGF: Unsupervised Point Normal Estimation by Learning Neural Gradient Function. *ArXiv* abs/2311.00389 (2023). https://api.semanticscholar.org/CorpusID:264833398

Guying Lin, Lei Yang, Congyi Zhang, Hao Pan, Yuhan Ping, Guodong Wei, Taku Komura, John Keyser, and Wenping Wang. 2023. Patch-Grid: An Efficient and Feature-Preserving Neural Implicit Surface Representation. *ArXiv* abs/2308.13934 (2023). https://api.semanticscholar.org/CorpusID:261242882

Siyou Lin, Zuoqiang Shi, and Yebin Liu. 2024. Fast and Globally Consistent Normal Orientation based on the Winding Number Normal Consistency. *ACM Trans. Graph.* 43, 6, Article 189 (Nov. 2024), 19 pages. doi:10.1145/3687895

Siyou Lin, Dong Xiao, Zuoqiang Shi, and Bin Wang. 2022. Surface Reconstruction from Point Clouds without Normals by Parametrizing the Gauss Formula. *ACM Transactions on Graphics* 42 (2022), 1 – 19. https://api.semanticscholar.org/CorpusID:251257800

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 2020. Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces. *ArXiv* abs/2011.13495 (2020). https://api.semanticscholar.org/CorpusID:227209266

Baorui Ma, Yu-Shen Liu, Matthias Zwicker, and Zhizhong Han. 2022. Surface Reconstruction from Point Clouds by Learning Predictive Context Priors. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 6316–6327. https://api.semanticscholar.org/CorpusID:248376851

Mayra Donaji Barrera Machuca, Paul Asente, Wolfgang Stuerzlinger, Jingwan Lu, and Byungmoon Kim. 2018. Multiplanes: Assisted Freehand VR Sketching. *Proceedings of the 2018 ACM Symposium on Spatial User Interaction* (2018). https://api.semanticscholar.org/CorpusID:52942811

Pierre Malraison. 2012. N-sided surfaces: A survey. (11 2012), 9.

Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2018. Occupancy Networks: Learning 3D Reconstruction in Function Space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), 4455–4465. https://api.semanticscholar.org/CorpusID:54465161

Günay Orbay and Levent Burak Kara. 2012. Technical Section: Sketch-based surface design using malleable curve networks. *Computers & Graphics* 36 (2012), 916–929. https://api.semanticscholar.org/CorpusID:13047314

David Palmer, David Bommes, and Justin Solomon. 2020. Algebraic Representations for Volumetric Frame Fields. *ACM Trans. Graph.* 39, 2, Article 16 (April 2020), 17 pages. doi:10.1145/3366786

Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. 2015. Flow aligned surfacing of curve networks. *ACM Trans. Graph.* 34, 4, Article 127 (July 2015), 10 pages. doi:10.1145/2766990

Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 165–174. https://api.semanticscholar.org/CorpusID:58007025

Yesom Park, Taekyung Lee, Jooyoung Hahn, and Myungjoo Kang. 2023. p-Poisson surface reconstruction in curl-free flow from point clouds. *ArXiv* abs/2310.20095 (2023). https://api.semanticscholar.org/CorpusID:264820180

Songyou Peng, Chiyu Max Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. 2021. Shape As Points: A Differentiable Poisson Solver. *ArXiv* abs/2106.03452 (2021). https://api.semanticscholar.org/CorpusID:235358422

Shervin Rasoulzadeh, Michael Wimmer, and Iva Kovacic. 2023. Strokes2Surface: Recovering Curve Networks From 4D Architectural Design Sketches. *Computer Graphics Forum* 43 (2023). https://api.semanticscholar.org/CorpusID:259138804

Nicolas Ray and Dmitry Sokolov. 2015. On Smooth 3D Frame Field Design. https://api.semanticscholar.org/CorpusID:14395367

Enrique Rosales, Chrystiano Araújo, Jafet Rodriguez, Nicholas Vining, Dongwook Yoon, and Alla Sheffer. 2021. AdaptiBrush. *ACM Transactions on Graphics (TOG)* 40 (2021), 1 – 15. https://api.semanticscholar.org/CorpusID:245015600

Bardia Sadri and Karan Singh. 2014. Flow-complex-based shape reconstruction from 3D curves. *ACM Trans. Graph.* 33 (2014), 20:1–20:15. https://api.semanticscholar.org/CorpusID:15971290

Ryan M. Schmidt, Azam Khan, Karan Singh, and Gordon Kurtenbach. 2009. Analytic drawing of 3D scaffolds. *ACM SIGGRAPH Asia 2009 papers* (2009). https://api.semanticscholar.org/CorpusID:727544

Ruwen Schnabel, Patrick Degener, and R. Klein. 2009. Completion and Reconstruction with Primitive Shapes. *Computer Graphics Forum* 28 (2009). https://api.semanticscholar.org/CorpusID:7779419

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. *ArXiv* abs/2006.09661 (2020). https://api.semanticscholar.org/CorpusID:219720931

Smoothstep. 2021. Quill. https://quill.art/

Alexander Sorkine-Hornung and Leif Kobbelt. 2006. Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *Eurographics Symposium on Geometry Processing*. https://api.semanticscholar.org/CorpusID:95114

Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Sprynski. 2016. Surfacing curve networks with normal control. *Comput. Graph.* 60 (2016), 1–8. https://api.semanticscholar.org/CorpusID:46253376

Anandhu Sureshkumar, Amal Dev Parakkat, Georges-Pierre Bonneau, Stefanie Hahmann, and Marie-Paule Cani. 2025. VRSurf: Surface Creation from Sparse, Unoriented 3D Strokes. *Comput. Graph. Forum* 44 (2025). https://api.semanticscholar.org/CorpusID:277572716

Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. 2009. Curve skeleton extraction from incomplete point cloud. *ACM SIGGRAPH 2009 papers* (2009). https://api.semanticscholar.org/CorpusID:1664563

Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. 2021. SA-ConvONet: Sign-Agnostic Optimization of Convolutional Occupancy Networks. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), 6484–6493. https://api.semanticscholar.org/CorpusID:237347240

Hui Tian, Chenyang Zhu, Yifei Shi, and Kai Xu. 2024. SuperUDF: Self-Supervised UDF Estimation for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (2024), 5965–5975. doi:10.1109/TVCG.2023.3318085

Peng-Shuai Wang, Yang Liu, and Xin Tong. 2022. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 15. https://api.semanticscholar.org/CorpusID:248525003

Sifan Wang, Yujun Teng, and Paris Perdikaris. 2021. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM J. Sci. Comput.* 43 (2021), A3055–A3081. https://api.semanticscholar.org/CorpusID:263888201

Zixiong Wang, Yunxiao Zhang, Rui Xu, Fan Zhang, Peng Wang, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. 2023. Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian. *ACM Transactions on Graphics (TOG)* 42 (2023), 1–14. https://api.semanticscholar.org/CorpusID:261529950

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Trans. Graph.* 33, 4, Article 131 (July 2014), 13 pages. doi:10.1145/2601097.2601128

Rui Xu, Zhiyang Dou, Ningna Wang, Shiqing Xin, Shuangmin Chen, Mingyan Jiang, X. Guo, Wenping Wang, and Changhe Tu. 2023. Globally Consistent Normal Orientation for Point Clouds by Regularizing the Winding-Number Field. *ACM Transactions on Graphics (TOG)* 42 (2023), 1 – 15. https://api.semanticscholar.org/CorpusID:258298598

Emilie Yu, Rahul Arora, Jakob Andreas Bærentzen, Karan Singh, and Adrien Bousseau. 2022. Piecewise-smooth surface fitting onto unstructured 3D sketches. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 16. https://api.semanticscholar.org/CorpusID:249319904

Emilie Yu, Rahul Arora, Tibor Stanko, Jakob Andreas Bærentzen, Karan Pratap Singh, and Adrien Bousseau. 2021. CASSIE: Curve and Surface Sketching in Immersive Environments. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021). https://api.semanticscholar.org/CorpusID:232134021

Paul Zhang, Josh Vekhter, Edward Chaoho Chien, David Bommes, Etienne Vouga, and Justin M. Solomon. 2020. Octahedral Frames for Feature-Aligned Cross Fields. *ACM Transactions on Graphics (TOG)* 39 (2020), 1 – 13. https://api.semanticscholar.org/CorpusID:219055106

Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. 2022. Learning Consistency-Aware Unsigned Distance Functions Progressively from Raw Point Clouds. *ArXiv* abs/2210.02757 (2022). https://api.semanticscholar.org/CorpusID:271293868

Yixin Zhuang, Ming Zou, Nathan A. Carr, and Tao Ju. 2013. A general and efficient method for finding cycles in 3D curve networks. *ACM Transactions on Graphics (TOG)* 32 (2013), 1 – 10. https://api.semanticscholar.org/CorpusID:14101050