CVPR
#2453

Learning to Solve PDEs on Neural Shape Representations

Anonymous CVPR submission

Paper ID 2453

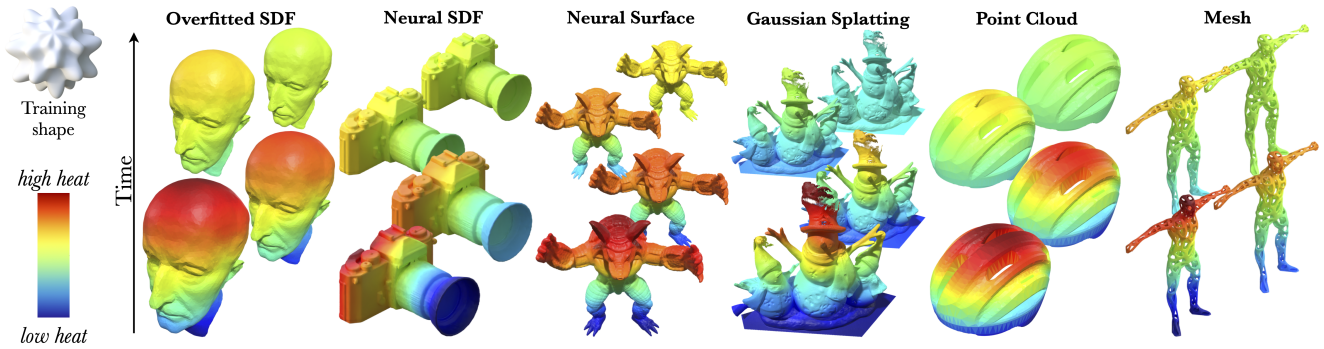

Figure 1. Our mesh-free, geometry-conditioned learned operator solves surface PDEs *directly in the neural domain* on multiple modalities, without mesh extraction, or per-instance optimization. Trained once on a single exemplar (SPIKE, top-left), the learned operator generalizes across unseen geometries, topologies, and input modalities. See supplemental for heat flow videos and also Poisson solves.

Abstract

Solving partial differential equations (PDEs) on shapes underpins many shape analysis and engineering tasks; yet, prevailing PDE solvers operate on polygonal/triangle meshes while modern 3D assets increasingly live as neural representations. This mismatch leaves no suitable method to solve surface PDEs directly within the neural domain, forcing explicit mesh extraction or per-instance residual training, preventing end-to-end workflows. We present a novel, mesh-free formulation that learns a local update operator conditioned on neural (local) shape attributes, enabling surface PDEs to be solved directly where the (neural) data lives. The operator integrates naturally with prevalent neural surface representations, is trained once on a single representative shape, and generalizes across shape and topology variations, enabling accurate, fast inference without explicit meshing or per-instance optimization while preserving differentiability. Across analytic benchmarks (heat equation and Poisson solve on sphere) and real neural assets across different representations, our method slightly outperforms CPM while remaining reasonably close to FEM, and, to our knowledge, delivers the first end-to-end pipeline that solves surface PDEs on both neural and classical surface representations. Code will be released on acceptance.

1. Introduction

Solving partial differential equations (PDEs) on surfaces is central to geometry processing, shape analysis, and many engineering tasks; examples include heat flow on surfaces, Poisson equation, and harmonic interpolation. Classical FEM-based solvers [11, 42] operate on (discretized) polygonal or triangle meshes, with well-understood accuracy and stability behavior. However, they cannot directly handle contemporary 3D assets that are increasingly represented as *neural shape representations* (e.g., point clouds or splats [36, 44], neural surfaces [29, 43], overfitted implicit shapes [15, 40], neural implicit fields [32, 44]). These representations are popular as they are differentiable, often topology-agnostic, and integrate naturally with modern learning and generative systems. This creates a mismatch: *mesh-centric PDE solvers do not operate in the domain where neural data lives*.

Handling surface PDEs *directly in the neural shape domain* removes mesh extraction, preserves end-to-end differentiability (crucial for inverse problems and PDE priors), and handles topology changes without intermediate re-meshing or reparameterization, while avoiding round-trip errors and engineering overhead. However, current workarounds either *extract a mesh* (e.g., marching cubes [23], dual contouring [14] and their neural variants [8, 9]) and shuttle results back, hindering differentiable pipelines; or *rely on per-instance residual training* (e.g., surface PINNs [12]),

which generalizes poorly across shape variation.

We introduce a geometry-aware neural PDE solver that targets the surface-structure component of PDEs and operates directly inside neural representations while retaining classical-solver fidelity and applying equally to classical surfaces. Inspired by the *Closest Point Method* (CPM) [39], our approach learns how surface geometry governs the extension of a surface field into a *narrow band*—the core operation of embedding-based solvers—allowing PDEs to be solved directly on their surface representations. A lightweight neural operator captures local geometric context (e.g., normals and principal curvature directions) and produces this extension. Trained once on a single shape, it generalizes across unseen geometries, modalities, and topologies, requires no meshing or per-instance optimization, and remains fully differentiable, making it suitable as a new neural PDE layer into existing neural training setups.

We demonstrate our neural PDE operator on different popular representations: neural surfaces, spherical neural surfaces [43], point clouds [16], overfitted occupancy fields [26], Gaussian splatting [16], as well as deep implicit fields [32]. We validate our results in two ways: (i) On spheres, we evaluate the heat and Poisson equations against analytical ground truth, and compare with both FEM and CPM [11, 39], to assess accuracy. (ii) For general shapes and modalities, we use dense-mesh FEM as reference to assess generalization and robustness. Our solver achieves competitive accuracy with zero meshing overhead and no extend–restrict shuttling, as required by CPM. Most importantly, we find that our method, once trained, generalizes surprisingly well across surface variations, topology, and meshing changes. As shown in Figure 1, a model trained on a single shape (the SPIKE) generalizes across diverse shapes and neural representations. We also ablate design choices and hyperparameters.

In summary, our main contributions are:

- Introducing a novel mesh-free, end-to-end differentiable solver for surface PDEs operating directly on both neural and classical surface representations.
- A lightweight, shape-conditioned network trained on a single shape that implicitly learns the narrow-band extension without per-shape optimization.
- Extensive evaluation showing generalization across unseen shapes, topologies, and representations, with competitive accuracy and speed on heat and Poisson equation.

2. Related Work

Classical and mesh-based methods. In Euclidean domains, PDEs are classically discretized by finite differences and Galerkin finite elements [25, 42]. Extending finite differences to curved manifolds typically requires embedding strategies, whereas Galerkin methods naturally generalize to arbitrary geometries via mesh-based formulations. On surfaces, *surface finite elements* (SFEM) discretize the manifold and apply intrinsic schemes on a triangulation, offering strong accuracy and stability guarantees under standard regularity and shape-regular mesh assumptions; see the survey [11]. Discrete differential geometry operators (e.g., the cotangent Laplacian) are also widely used for geometry processing and harmonic problems on meshes [10, 27]. The main limitations are geometric and practical: performance hinges on mesh quality, evolving or noisy geometries often require (re)meshing, and distortion/tangling can degrade conditioning, accuracy, and robustness. The main restriction being that such methods cannot directly be applied to current neural representations, without meshing.

Embedding and unfitted methods. Embedding methods solve surface PDEs in the ambient domain while enforcing surface constraints. We build on the *Closest Point Method* (CPM), which alternates extension and Cartesian updates on a narrow band and is valued for simplicity and robustness [39]; accuracy/flexibility have been boosted with high-order and meshfree RBF–FD stencils and least-squares implicit variants, including moving surfaces [33, 34]. Stochastic *Projected Walk on Spheres* offers discretization-free Monte Carlo solutions via repeated manifold projections [41], and CPM has been adapted to interior boundaries [17]. Unfitted FEM avoids explicit surface meshes by solving on a background grid: *CutFEM* stabilizes cut cells with ghost penalties [2, 3], while *TraceFEM* restricts spaces to an implicit level set and extends to evolving interfaces [20, 30]. Despite reduced meshing effort, these approaches still shuttle information between surface and grid, introducing overhead and potential bias, especially with implicit surfaces. In contrast, our method performs *grid-to-grid* updates without extend–restrict loops.

Learning-based solvers. Physics-Informed Neural Networks (PINNs) [38] impose PDE residuals and boundary terms in the training loss, enabling mesh-free forward/inverse solves but typically requiring *per-instance* optimization; they are sensitive to stiffness, boundary enforcement, residual weighting, and training stability at scale. Surface extensions (e.g., [12]) demonstrate feasibility on manifolds without meshing yet inherit the same optimization and runtime burdens. A complementary direction, *neural operator* (DeepONet, FNO) [18, 21, 24] and manifold variants [5, 35], amortizes solution maps across problem families but generally relies on supervision from classical solvers, assumes fixed discretizations/charts, and does not natively target neural implicit geometry. Closer to our goals, *implicit neural spatial representations* treat an implicit neural representation (INR) as the spatial discretization and evolve its weights over time to solve time-dependent PDEs [6, 7]. These methods show strong accuracy–memory trade-offs but still operate via global weight evolution and per-problem time integration, and cannot be directly used to unseen

shapes. In contrast, we learn a *local, geometry-conditioned update operator* that works directly in a *narrow band* around the surface. It takes geometric cues from diverse neural shape representations and performs a single forward update, avoiding per-instance training and mesh dependencies while retaining solver-level accuracy.

Neural shape representations. Modern 3D pipelines increasingly favor neural implicit/explicit representations over traditional meshes. Point clouds (e.g., PointNet/PointNet++ or splats [16]) provide a mesh-free sampling interface but lack continuity and differential structure by default [36, 37]. Neural *implicit* fields capture geometry as continuous functions: signed distance fields (DeepSDF [32]) and occupancy networks [26] model surfaces at effectively infinite resolution and are widely used for reconstruction and analysis. Overfitted implicit neural representations, such as SIREN [40], fit a single shape/scene as a coordinate MLP and expose smooth values and derivatives. For genus-0 surfaces, spherical neural surfaces map $\mathbb{S}^2$ to embedded shapes and expose intrinsic operators without meshing [43]. Scene appearance and volume are commonly modeled by neural radiance fields (NeRF) [28], with real-time explicit variants via 3D Gaussian splatting [16]. Triplane feature layouts (e.g., EG3D [4]) factor 3D into three orthogonal 2D feature planes that are both expressive and efficient for reconstruction and generation. Finally, recent latent encodings for neural fields, such as 3DShape2VecSet [44], represent shapes as sets of vectors tailored for generative modeling and downstream learning. These representations are differentiable and often topology-agnostic, making them suitable for our PDE solver that operates without mesh extraction (see Section 4).

3. Method

We propose a representation-agnostic solver that computes surface PDEs on *neural surfaces*, while remaining compatible with other geometric representations. Given a surface, which may be neural (e.g., Spherical Neural Surface, SDF or occupancy INR, overfitted implicit, or point cloud with normals), we first extract local geometric context such as normals and curvature tensor at sampled surface points. We then build a narrow Cartesian band around the embedded surface, following the principle of the *Closest Point Method* [39], and reformulate the surface PDE as a volumetric one defined within this neighborhood. Surface functions are extended to the band through a closest point extension that enforces normal constancy — the main assumption underlying embedding-based solvers. *This extension is modeled by a lightweight geometry-conditioned neural operator that learns it implicitly.* The operator acts locally across the surface, recognizing the underlying geometry from local features and grid stencils to produce local band functions, which are then assembled into a single global solution. This design helps generalize across a wide range of shape modalities, topologies, and surface functions.

Our training is local and data-efficient: patches from a single representative shape (see SPIKE in Figure 1) suffice to learn the operator. Two aspects are central to its construction: (i) structuring the architecture with geometric conditioning, and (ii) ensuring generalization to unseen functions at test time. At inference, the method first produces the extended band function by applying the local operator across all patches — each acting locally but contributing to a single global update of the field. The PDE is then solved directly within the band, as in [39], and for time-dependent problems, this process reduces to repeated global updates over time steps. The approach requires no meshing or per-instance optimization, remains fully differentiable, and integrates as a drop-in neural PDE layer.

Closest Point Method (CPM). The original method [39] embeds surface PDEs in a thin Cartesian *narrow band* around the surface $\mathcal{S}$. The equivalent volumetric PDE is then solved inside this band using standard numerical methods such as finite differences (FD), Runge–Kutta (RK) schemes and time integrators such as forward Euler for time-dependent problems. For the solution of the volumetric PDE to coincide with that of the original surface PDE when restricted to $\mathcal{S}$, the surface function must be extended into the band through a *closest point extension*, ensuring constancy along surface normals.

The CPM alternates between two simple operations: (i) a standard discrete volumetric *solve* in the *narrow band*, and (ii) a *re–extension* step to ensure normal constancy. At the core is the closest–point map cp sending a band point $x \in \mathbb{R}^3$ to its nearest point on the surface $\mathrm{cp}(x)$. A surface function $u_{\mathcal{S}}$ is extended to the band by $u(x) := u_{\mathcal{S}}(\mathrm{cp}(x))$. When u is (approx.) constant along normals, ambient derivatives restricted to surface $\mathcal{S}$ coincide with intrinsic ones, allowing replacement of surface operators (e.g., gradient/Laplacian) by standard finite–difference stencils during *solve* step.

However, each time step (involving a FD *solve*) generally breaks normal constancy, causing the field to vary along surface normals. Hence, CPM *re–extends* by overwriting the band value at x with the value at $\mathrm{cp}(x)$ (which is interpolated from the neighbouring grid points). In practice, CPM requires (i) constructing a band of width ε, (ii) efficient evaluation of closest points, and (iii) *re-extension* operation, and (iv) consistent handling of boundary conditions by tagging band cells whose projections lie on any boundary $\partial\mathcal{S}$. For details, see the original paper [39].

The CPM is simple, robust, and reuses off–the–shelf Cartesian solvers; however, it requires *extend–solve–reextend*, using surface information, at every iteration. Instead, we design a neural network that replaces CPM's *re-extension* step implicitly and naturally accommodates neural shape representations.

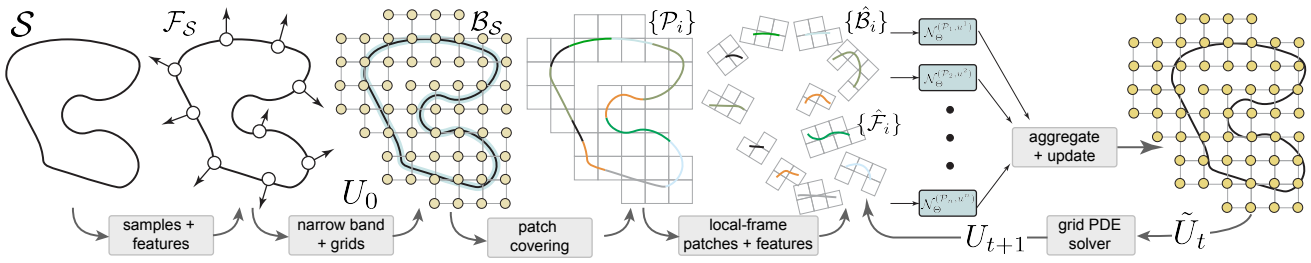

Figure 2. **Pipeline overview.** From a surface $\mathcal{S}$, we sample points and geometric features (normals, local features). Around an *ε-narrow band* around the shape, we gather Cartesian grids $\mathcal{B}_\mathcal{S}$ to store an initial field U_0 extended from surface values and covered by overlapping, surface-centred patches $\{\mathcal{P}_i\}$. Each patch is reoriented to its local frame, yielding $\{\hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i\}$, which are processed by our lightweight geometry-conditioned operators $\{\mathcal{N}_\Theta\}$ to produce local updates (see Figure 3). The local updates are smoothly aggregated to form the global band update $\tilde{U}_t$ and advanced with a standard grid PDE time step to get U_{t+1}. Iterating this loop yields surface PDE solutions directly in the neural domain via grid-to-grid updates, *without* mesh extraction or extend–restrict shuttling.

3.1. Algorithm Steps

We learn a local neural solver that operates around a *narrow band* of implicit or explicit surface and produces band updates that keep the function constant along surface normals. Our pipeline is representation-agnostic and applies to any surface that supports (i) sampling on the surface, and (ii) estimating local differential cues. In Section 4, we discuss how to estimate these for different neural shape representations.

Inputs and notation. Let $\mathcal{S} \subset \mathbb{R}^3$ be a smooth surface representation, either neural (e.g., SNS, SDF/occupancy INR) or traditional (e.g., point cloud, mesh). We denote the unit normal at $x \in \mathcal{S}$ by $\mathbf{n}(x)$, and the principal curvature directions by $(\mathbf{t}_1(x), \mathbf{t}_2(x))$, with $\mathbf{t}_1$ aligned to the maximum curvature and $\mathbf{t}_2$ to the minimum; when curvatures are equal (umbilic points), any orthogonal tangent pair is chosen and the order doesn't matter. A regular Cartesian grid G provides samples in a narrow ε-band around $\mathcal{S}$. We now describe our full pipeline (see Figure 2).

Pipeline Overview. (i) *Surface feature extraction.* Compute geometric descriptors on the surface on a set of surface samples as: $\mathcal{F}_\mathcal{S} := \big\{(x,\ \mathbf{n}(x),\ \mathbf{t}_1(x),\ \mathbf{t}_2(x)) \ \big|\ x \in \mathcal{S}\big\}$.

(ii) *Narrow-band construction.* Define a uniform volumetric grid G in a bounding box of $\mathcal{S}$ and retain grid nodes within distance ε of the surface: $\mathcal{B}_\mathcal{S} := \big\{y \in G \ \big|\ \mathrm{dist}(y, \mathcal{S}) \le \varepsilon\big\}$.

(iii) *Overlapping local patches.* Cover $\mathcal{S}$ and its *narrow band* $\mathcal{B}_\mathcal{S}$ with surface-centered patches. Each patch is anchored at a surface point $p_i^c \in \mathcal{S}$, which serves as its center, and is defined as: $\mathcal{P}_i := (\mathcal{L}_i,\ \mathcal{B}_i,\ \mathcal{F}_i)$, where $\mathcal{L}_i = (p_i^c, \mathbf{n}(p_i^c), \mathbf{t}_1(p_i^c), \mathbf{t}_2(p_i^c))$ defines a local frame at p_i^c used to express all subsequent quantities. $\mathcal{B}_i \subset \mathcal{B}_\mathcal{S}$ collects nearby band samples and $\mathcal{F}_i$ gathers surrounding surface features (points and normals in our implementation). All quantities within a patch are expressed in the local frame $\mathcal{L}_i$. Accordingly, coordinates of local band samples and local surface features (i.e., $\mathcal{B}_i$, $\mathcal{F}_i$) are transformed into this intrinsic coordinate system, and we denote their local-frame representations with a hat symbol (i.e., $\hat{\mathcal{B}}_i$, $\hat{\mathcal{F}}_i$).

(iv) *Learned band-to-band update (neural operator).* Given a scalar band field $U_t : \mathcal{B}_\mathcal{S} \to \mathbb{R}$ at time t, we denote by $u_t^i := U_t|_{\mathcal{B}_i}$ its restriction to the local band associated with patch $\mathcal{P}_i$. A lightweight neural operator $\mathcal{N}_\Theta$ consumes per-patch stencils (band values and local geometry) and updates the sampled field to enforce normal constancy: $\tilde{u}_t^i = \mathcal{N}_\Theta^{(\mathcal{P}_i, u_t^i)}(\hat{\mathcal{B}}_i)$. We denote by a tilde $(\tilde{\cdot})$ functions that are approximately constant along surface normals.

(v) *Aggregation of local predictions.* Local predictions are combined to reconstruct a global band field via smooth, proximity-weighted averaging as,

$$\tilde{U}_t(x) = \frac{\sum_{i: x \in \mathcal{B}_i} \exp(-\|x - p_i^c\|^2/T)\, \tilde{u}_t^i(x)}{\sum_{i: x \in \mathcal{B}_i} \exp(-\|x - p_i^c\|^2/T)},$$

with temperature $T > 0$ controlling the blending.

(vi) *Time evolution in the band.* Since $\tilde{U}_t$ is nearly constant along normals, intrinsic surface operators can directly be accurately approximated. We evolve $\tilde{U}_t$ using standard finite differences and forward Euler scheme (e.g., for heat/diffusion):

$$U_{t+dt}(x) = \tilde{U}_t(x) + dt\, \Delta \tilde{U}_t(x), \qquad x \in \mathcal{B}_\mathcal{S},$$

where Δ is the discrete Laplacian on G restricted to the band. Boundary conditions are imposed on band nodes whose projections lie on $\partial\mathcal{S}$, as in [39]. Note that U_{t+dt} carries no tilde, as there is no guarantee that the updated function remains constant along the surface normals.

(vii) *Iterate or reconstruct.* If additional steps are needed, return to the *learned band-to-band update* (step iv) and repeat the aggregate–evolve cycle. At any time, we 'readout' the surface solution by restricting the band field to $\mathcal{S}$, by interpolating with radial basis functions (Gaussian kernels in ours) for a smooth surface field.

3.2. Overlapping Local Patches

We decompose the surface $\mathcal{S}$ and its *narrow band* $\mathcal{B}_\mathcal{S}$ into overlapping, surface–centered local patches, each aggregating nearby band samples for grid-based updates and nearby

surface samples with geometric features for conditioning. A patch $\mathcal{P}_i$ is centered at a surface point $p_i^c \in \mathcal{S}$. Around p_i^c, we gather the k nearest band nodes to form a local stencil $\mathcal{B}_i$. In Section 4, we discuss choice of k for good accuracy–locality trade-off. Next, we take tight axis-aligned bounding box of $\mathcal{B}_i$ and dilate it by a small margin. All surface samples whose coordinates lie inside this enlarged box, together with their normal, constitute the surface-conditioning set $\mathcal{F}_i$, providing a broader geometric context around the local surface region. Thus, the tuple $\mathcal{P}_i := (\mathcal{L}_i, \mathcal{B}_i, \mathcal{F}_i)$, with $\mathcal{L}_i = (p_i^c, \mathbf{n}(p_i^c), \mathbf{t}_1(p_i^c), \mathbf{t}_2(p_i^c))$, defines one such patch. Using $\mathcal{L}_i$, we express all quantities of $\mathcal{B}_i$ and $\mathcal{F}_i$ in the local frame centered at p_i^c with basis $(\mathbf{n}(p_i^c), \mathbf{t}_1(p_i^c), \mathbf{t}_2(p_i^c))$, ensuring invariance to translation and rotation. Degenerate cases where curvature directions are ambiguous (e.g., umbilic regions where principal curvatures coincide) are naturally present in the training data and are further handled through data augmentation: random rotations of the local patch (encompassing both normal and tangent directions) enforce the network to learn rotational invariance.

We progressively generate patches across $\mathcal{S}$, expanding outward from an initial (surface) seed so that coverage naturally propagates over the surface (similar to floodfill restricted to the surface). This strategy yields a family of overlapping patches whose union covers the entire band, while maintaining controllable redundancy. The degree of overlap is controlled by a spacing parameter in our patch-placement procedure, which determines how far each new center is placed from the previous ones while ensuring that adjacent band regions still overlap. Smaller spacing increases redundancy and overlap, whereas larger spacing yields sparser coverage. Increasing either improves robustness but adds computational cost. In our implementation, nearest-neighbor queries on $\mathcal{B}_\mathcal{S}$ and $\mathcal{S}$ are accelerated with a KD-Tree.

Coverage condition. A potential issue arises from using a fixed number of neighbours k to define each patch: for small grid spacing Δx or large band width ε, some band points may lie too far from any surface center p_i^c, leading to incomplete coverage of the band $\mathcal{B}_\mathcal{S}$. This motivates us to seek a relation linking ε, Δx, and k.

This setting is closely related to the classical *Gauss circle problem* (and its three-dimensional analogue, the *Gauss sphere problem*, see [19]), which counts the number of lattice points contained in a ball of radius r. In three dimensions, neglecting higher-order terms, the number of grid points N_3 within a band of radius ε and spacing Δx is well approximated by the volume of a ball of radius $\varepsilon/\Delta x$:

$$N_3(\varepsilon/\Delta x) \approx \tfrac{4}{3}\pi \left(\frac{\varepsilon}{\Delta x}\right)^3.$$

Ensuring every band point is covered by at least one patch yields the condition, we arrive at:

$$\varepsilon \leq \Delta x \left(\tfrac{3k}{4\pi}\right)^{1/3}.$$

3.3. Learning a Neural Update Operator

3.3.1. Neural geometry encoder

Our neural geometry encoder is a lightweight network built from small MLPs with an attention-like interaction for local geometry adherence. It operates locally in the *narrow band* around the surface and updates the target function in a single step, *directly* on grid values.

Network architecture. As illustrated in Figure 3, our neural solver acts on local geometry attributes and updates the field within each narrow band $\mathcal{B}_i$. The inputs are the query point, the band points $\hat{\mathcal{B}}_i$, and their current field values u^i, together with the associated surface features $\hat{\mathcal{F}}_i$, all expressed in the local frame $\mathcal{L}_i$. The query point q attends to its neighboring band samples to obtain spatial weights, while $\hat{\mathcal{F}}_i$ modulates this aggregation so the update is *conditioned* on local geometry. The output is a weighted sum producing the updated value at q. To handle a variable number of surface features per patch, we apply mean pooling (similar to [36]), yielding a fixed-size descriptor. Formally, we encode the neural update as:

$$\begin{aligned} \mathcal{N}_\Theta : \mathbb{R}^3 \times \mathbb{R}^{k\times 3} \times \mathbb{R}^{N_i \times 6} \times \mathbb{R}^k \quad &\longrightarrow \quad \mathbb{R}, \\ (q, \hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i, u^i) \quad &\longmapsto \quad \mathcal{N}_\Theta(q, \hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i, u^i) \end{aligned} \tag{402}$$

where Θ contains the weights of the three MLPs encoder $(\theta_1, \theta_2, \theta_3)$ and a learnable scalar λ. For convenience, we define a compressed form of the network where the patch $\mathcal{P}_i = (\mathcal{L}_i, \hat{\mathcal{B}}_i, \hat{\mathcal{F}}i)$ and its current field u^i are fixed:

$$\mathcal{N}_\Theta^{(\mathcal{P}_i, u^i)} : q \longmapsto \mathcal{N}_\Theta(q, \hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i, u^i)$$

In practice, our implementation evaluates multiple queries simultaneously. Let $Q = \{q_1, \cdots, q_b\} \subset \mathbb{R}^3$; then $\mathcal{N}_\Theta^{(\mathcal{P}_i, u^i)}(Q) = \left\{\mathcal{N}_\Theta^{(\mathcal{P}_i, u^i)}(q)\right\}_{q\in Q}$. When solving PDE, we set $Q = \hat{\mathcal{B}}_i$ as band values are updated at band samples.

3.3.2. Training setup

Dataset construction. We train on a single representative surface, the SPIKE, represented by an SNS [43] $S_\omega : \mathbb{S}^2 \subset \mathbb{R}^3 \to$ SPIKE. Note that since our network only depends on first and second order quantities (i.e., normal and curvatures), the single SPIKE shape, having a good distribution of curvature profiles, is sufficient to train on – we test its generalization behavior in Sec. 4. Surface is split into patches $\{\mathcal{P}_i\}$, as Sec. 3.2. For each patch, we apply random rotations. Each band $\mathcal{B}_i$ is paired with closest points as,

$$\mathrm{cp}(x) := S_\omega\Big(\arg\min_{y\in\mathbb{S}^2} \|x - S_\omega(y)\|_2^2\Big), \quad x \in \mathcal{B}_i,$$

forming $\Pi_i := \{\mathrm{cp}(x) \mid x \in \mathcal{B}_i\}$. For supervision, we use monomials:

$$\mathcal{M} := \{\, (x, y, z) \mapsto x^i y^j z^k \mid i+j+k \leq 5 \,\}.$$

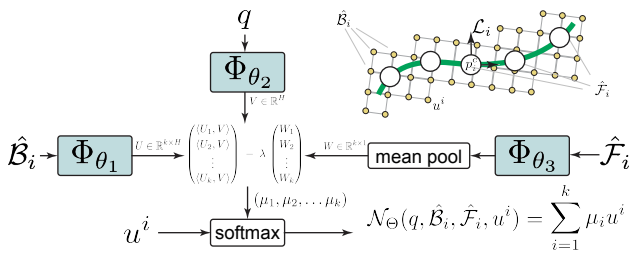

Figure 3. **Neural update operator (overview).** Given a query location q, the local band $\hat{\mathcal{B}}_i$ expressed in the local frame $\mathcal{L}_i$ centered around p_i^c for patch $\mathcal{P}_i$, with locally-transformed surface features $\hat{\mathcal{F}}_i$ (e.g., positions, normals), and current band values u^i at grid sites (time index t omitted for brevity; full notation u_t^i), our operator predicts updated function value at location q. Trainable components include compact MLP blocks ($\Phi_{\theta_1}, \Phi_{\theta_2}, \Phi_{\theta_3}$) and a scalar λ. The full network $\mathcal{N}_\Theta$ produces the updated band value at q, yielding a single geometry–conditioned grid-to-grid step.

Note that we rely on any (unseen) function to be sufficiently approximated by only their top few Taylor coefficients since PDE solutions are smooth; hence, learning over monomial functions turns out to be sufficient in our tests. For each such $g \in \mathcal{M}$, we evaluate $g(\mathcal{B}_i)$ and $g(\Pi_i)$, where the first serves as the network input and the second as the ground truth target, yielding pairs:

$$\mathcal{E}_i := \{ \, (g(\mathcal{B}_i),\, g(\Pi_i)) \;\big|\; g \in \mathcal{M} \, \}.$$

Thus, the training dataset is $\mathcal{D} := \{ \, (\hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i, \mathcal{E}_i) \, \}_{i=1}^{p}$.

Loss functions. We use two loss functions. The primary mean squared error enforces accurate function reconstruction:

$$L_{\mathrm{MSE}} = \frac{1}{k|\mathcal{D}||\mathcal{M}|} \sum_{\substack{(\hat{\mathcal{B}}_i, \hat{\mathcal{F}}_i, \mathcal{E}_i) \in \mathcal{D} \\ (g, g^{\mathrm{GT}}) \in \mathcal{E}_i}} \| \mathcal{N}_\Theta^{(\mathcal{P}_i, g)}(\hat{\mathcal{B}}_i) - g^{\mathrm{GT}} \|_2^2.$$

To check geometric consistency, we monitor a normal-consistency term enforcing constancy along surface normals, evaluated over different patches and functions $(\mathcal{P}, u)$:

$$L_{\mathrm{NC}} := \sum_q \left| \langle \nabla_q \, \mathcal{N}_\Theta^{(\mathcal{P}, u)}(q),\, \mathbf{n}(\mathrm{cp}(q)) \rangle \right|.$$

This term encourages the field gradient to remain orthogonal to the surface normals; when the dot products are close to zero then the field is nearly constant along the surface normals. The gradient $\nabla_q \mathcal{N}_\Theta^{(\mathcal{P}, u)}(q)$ is computed via automatic differentiation with respect to the query location q (other terms are detached). The overall objective is:

$$L = L_{\mathrm{MSE}} + \alpha \, L_{\mathrm{NC}}.$$

We conduct an ablation study (see supplemental) to assess the contribution of the normal-consistency term L_{NC} and to determine an appropriate weighting α. This analysis highlights how enforcing normal-aligned consistency improves accuracy across surfaces.

4. Evaluation

Baselines. We evaluate against two established families: (i) *Surface FEM (SFEM)* discretizes the PDE intrinsically on an explicit triangle mesh: unless stated otherwise, we use linear elements with the cotangent Laplacian and a consistent mass matrix [11, 42]. Meshes come either from the ground-truth surface or from marching cubes [22] on the same implicit surface used by our method, with multiple resolutions to probe convergence and mesh-quality effects. We do not aim to outperform FEM, whose solvers are highly mature and extensively optimized. Instead, we use FEM to provide reliable reference solutions and expected error levels, illustrating the behavior and capability of our method. Comparisons to FEM should therefore be viewed as grounding rather than competition.
(ii) *Closest Point Method (CPM)* solves in an Eulerian *narrow band* around the surface using standard Cartesian stencils and alternates *solve* and *re–extend* steps [39]. Closest-point projections and normals are computed from the same implicit geometry used by our method to ensure parity. We tested different interpolation schemes for the *re-extension* step (trilinear versus polynomial) and retained the polynomial one, consistent with the original CPM formulation, as it yielded the best accuracy–efficiency trade-off.

For fairness, all baselines share the same right-hand sides, initial data, and boundary conditions; we also matched resolution schedules and aligned stopping criteria (final time or steady-state residual).

PDEs. We benchmark *heat diffusion* ($\partial_t u = \Delta_{\mathcal{S}} u$) and *Poisson* ($\Delta_{\mathcal{S}} u = f$) on closed surfaces. For well-posedness, we initialize the heat equation with a prescribed initial condition and let it evolve until reaching a steady state. For Poisson, we choose a zero-mean function f over the surface and select the zero-mean solution on the surface, since the kernel of the Laplace–Beltrami operator on closed surfaces corresponds to constant functions. We report both boundary-free cases and settings with Dirichlet conditions on embedded curves (imposed identically for all methods).

Metrics. We report normalized mean absolute error (NMAE), normalized max error (NMaxE), normalized root mean square error (NMRSE). Input probe function ranges were normalized to $[-0.5, 0.5]$. See supplemental for details.

Shape representations. We evaluate across common shape encodings and derive the geometric cues needed by our operator in a consistent way. (i) *Meshes:* normals are area–weighted averages of incident face normals; mean curvature normals follow the cotangent discretization, and principal curvatures are obtained from discrete differential operators [10, 27]. (ii) *Point clouds:* we estimate normals via PCA of k-NN neighborhoods with sign disambiguation along a coarse viewpoint field. More advanced point normal prediction [13] may be used; we do not use curvature features in

this case. (iii) *Spherical Neural Surfaces (SNS):* The method provides direct access to normals and first/second fundamental forms by differentiating the mapping $S_\omega : \mathbb{S}^2 \subset \mathbb{R}^3 \to \mathbb{R}^3$; principal curvatures follow from the Weingarten map [43]. (iv) *Implicit SDF fields (DeepSDF/overfitted INRs):* normals are $\nabla\phi/\|\nabla\phi\|$ for these implicit fields [32]. However, although we could have used curvatures using level-set formulas [31], we found the estimates to be noisy; hence, we did not use curvature estimates in these cases. For occupancy fields [26, 40], we compute normals from the implicit gradient of the network near the isosurface. (iv) *GSplats:* after the training we treat all splats as a point cloud and filter out those that have high depth error. We extract the features using the same protocol as in (ii). See Figure 1.

Accuracy and Convergence on Spheres. We begin on the unit sphere, where closed-form solutions for heat diffusion and Poisson problems are available via spherical harmonics, enabling precise accuracy and convergence studies (see Chapter 6 [1]). We also compare across four sphere mesh resolutions—*coarse*, *medium*, *fine*, and *very fine* with approximately $0.1k$, $1k$, $10k$, $100k$ vertices, respectively.

Across resolutions, our solver matches CPM in accuracy (Table 1), even when CPM benefits from dense meshes, and follows similar error trends. Experiments further show that high-resolution SFEM provides a reliable proxy for ground truth (used later when analytic solutions are unavailable). Unlike mesh-centric pipelines, our errors are notably stable under remeshing and connectivity changes (see supplemental), indicating reduced sensitivity to sampling irregularities and local topology. Most importantly, our method operates natively in the neural implicit domain and can be used as a drop-in *neural PDE layer* within standard deep-learning frameworks.

Table 1. **Poisson on the sphere (analytic GT).** Error vs. resolution for SFEM, CPM, and our method. We report normalized mean (NMAE) and max (NMaxE) errors (lower is better); all methods use identical right–hand sides and evaluation grids. See the supplemental for the corresponding heat equation table.

Solver	Resolution	NMAE ↓	NMaxE ↓
SFEM	Coarse	1.05×10^{-2}	2.32×10^{-2}
	Medium	6.48×10^{-4}	1.90×10^{-3}
	Fine	2.84×10^{-4}	6.30×10^{-4}
	Very fine	1.11×10^{-4}	1.29×10^{-4}
CPM	Coarse	4.56×10^{-2}	1.36×10^{-1}
	Medium	1.49×10^{-2}	3.59×10^{-2}
	Fine	1.46×10^{-2}	3.49×10^{-2}
	Very fine	1.48×10^{-2}	3.52×10^{-2}
Ours	Coarse	2.75×10^{-2}	9.14×10^{-1}
	Medium	1.24×10^{-2}	2.99×10^{-2}
	Fine	1.33×10^{-2}	3.17×10^{-2}
	Very fine	1.32×10^{-2}	3.23×10^{-2}

Handling Different Shape Representations. We run our solver on multiple shape encodings (mesh, point cloud, SNS, SDF/occupancy INRs, Gaussian Splatting), using the same local features described above (positions and normals, with optional curvature), illustrated in Figure 1. Evaluation is performed on the steady-state solution, whose ground truth is analytic (given by the mean of the initial condition over the surface). Although a direct comparison is not strictly meaningful across different shapes and representations, similar trends are observed (see Table 2), with SNS yielding the best results—consistent with its superior geometric estimates. Generalization across modalities also indicates robustness to noisy geometric quantities, as different representations provide features of varying quality.

Table 2. Comparison of different **surface representations** for the heat equation on the steady state. The metric is the Normalized Root Mean Squared Error (NRMSE), computed against the analytic solution.

Representation / Shape	NRMSE ↓
Neural SDF / Camera	2.17×10^{-2}
Overfitted SDF / Max Planck Face	3.03×10^{-2}
Spherical Neural Surface / Armadillo	$\mathbf{1.88 \times 10^{-2}}$
Gaussian Splatting / Snowman	6.15×10^{-2}
Point Cloud / Hat	2.94×10^{-2}
Mesh / Holey Human	1.92×10^{-2}

Generalization across shapes. Our neural operator, trained once on a single shape, transfers seamlessly to *unseen* shapes and topologies across input modalities. As illustrated in Figure 4 and quantified in Table 3, it closely matches SFEM reference solutions for Poisson across diverse geometries (e.g., organic, CAD parts with sharp transitions, and thin-structure cases) with consistently low NRMSE. This amortized, geometry-conditioned behavior underpins cross-shape generalization. Nonetheless, errors tend to appear near regions where closest points are not unique (e.g., sharp

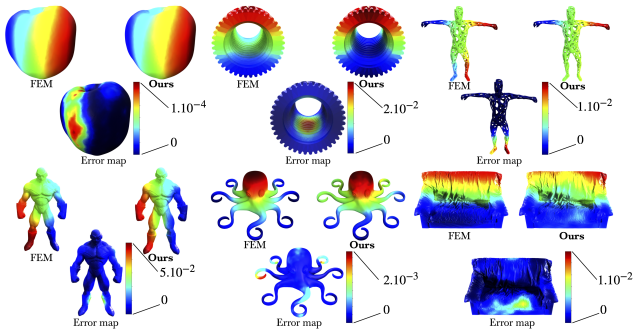

Figure 4. **Comparison to SFEM on diverse shapes.** For each object, left shows SFEM and right shows ours; the small inset below visualizes the pointwise error (ours vs. SFEM) with a hot–cold colormap. See color bar for error scale and the supplemental for per-shape statistics. (Error colormaps are normalized per instance.)

edges, thin parts, near the medial axis), a limitation inherited from the CPM formulation.

Table 3. Poisson equation results on different shapes (NRMSE ↓). Errors are computed against the FEM as ground truth solution.

Shape	NRMSE	Shape	NRMSE
Jared	5.12×10^{-2}	Sofa	2.65×10^{-2}
Octopus	1.13×10^{-2}	Holey Human	2.67×10^{-2}
Apple	3.20×10^{-3}	Fastener	3.95×10^{-2}

Boundary handling. As in CPM, Neumann conditions are naturally satisfied since our update enforces normal consistency. For exterior Dirichlet boundaries, we follow the CPM practice of clamping boundary values and updating only in the band. On the Max Planck head (see Fig. 5), we solve heat with homogeneous and sinusoidal Dirichlet data and compare to a high-resolution SFEM reference on the corresponding open surface. Errors remain low and stable; detailed plots and per-case statistics are provided in the supplemental. These results confirm that exterior Dirichlet conditions are handled effectively, leveraging CPM's boundary treatment, which fits naturally within our method.

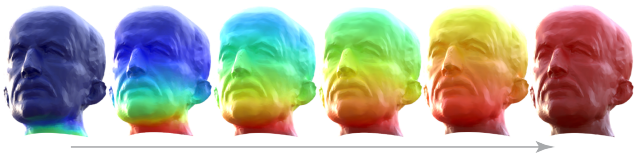

Figure 5. **Dirichlet boundaries on an open surface.** Heat diffusion on the Max Planck head cut at the neck (left to right) with boundary values clamped on the cut. See supplemental.

Ablations. Unless noted, all ablations use identical data, schedules, and hyperparameters; results across tables are not directly comparable. Overall, the studies support a simple, data-efficient design (see supplemental). *(i) Geometric consistency vs. data.* A small weight on L_{NC} reliably lowers errors; large weights bias toward trivial normal-invariant fields and hurt fidelity. Even without L_{NC}, the operator largely maintains normal consistency. *(ii) Local features.* Positions and normals matter most; accurate curvature adds marginal benefit. *(iii) Band receptive field.* Larger k yields diminishing returns: errors drop mildly then plateau, consistent with convex aggregation. A mid-range k ($\sim$300) balances accuracy, stability, and cost. *(iv) Model capacity.* Shallow, narrow MLPs suffice; deeper/wider variants bring marginal gains and may overfit. *(v) Learnable attention strength.* A single learnable scalar λ modestly but consistently improves accuracy, providing lightweight adaptation.

5. Conclusion

We presented a mesh-free solver that computes surface PDEs *directly* on neural implicit surfaces. Our geometry-conditioned local operator performs a direct grid-to-grid update in a *narrow band* around the target surface, remains fully differentiable, and generalizes across shapes and topologies without requiring meshing or per-instance optimization. On analytic benchmarks and real neural assets (across different representations), our method achieves competitive accuracy and runtime while eliminating mesh extraction and extend–restrict shuttling as required in the classical CPM method. The approach integrates naturally into learning pipelines, opening up options to directly add PDE neural layers for analysis, editing, and reconstruction involving neural shape representations. While our focus is on the *geometric* component of PDE solving—rather than optimizing high-order numerical schemes—this design makes our operator complementary to stronger discretizations and suitable for integration into more advanced solvers. This opens promising directions for coupling our geometry-aware update with advanced solvers, enabling PDE layers operating directly on neural surfaces while remaining compatible with classical surface representations.

Limitations and Future Work

Self-intersections and medial-axis neighborhoods. Near self-intersections and/or close to the medial axis, SDF gradients may become unreliable, which can degrade update quality. A pragmatic remedy is a *hybrid fallback*: detect ill-conditioned patches (e.g., via $|\nabla\phi|$ or curvature thresholds) and locally hand off to a classical local FEM/embedded solver; the operator remains applicable elsewhere. Eventually, ours relies on the neural implicit surface to be accurate on/near the surface.

Evolving surfaces and rebanding. When the underlying surface moves under the PDE (e.g., surface evolution under curvature flow), the *narrow band* must be rebuilt and resampled, reducing any amortization benefits. Incremental band updates, and reusing cached features can mitigate cost; extending the operator to predict both updates and band maintenance is also an interesting future direction.

Grid dependence and scale effects. Our operator is not fully discretization-invariant: performance can vary with grid spacing and band thickness. Scale-aware conditioning and multi-resolution training may improve robustness. Learning on a modest range of resolutions generalizes in practice. Extending our coverage condition, it will be interesting to derive a relation between the sampling density and implicit surface quality to further guide the grid and sampling processes.

Finally, in this work, we focus on Poisson solves and heat equation; strongly anisotropic or stiff systems may require tailored stabilization or implicit time-stepping. Incorporating learnable preconditioners, implicit updates, or operator splitting within our framework is a promising future work.

CVPR #2453

References

[1] Kendall E. Atkinson and Weimin Han. *Spherical Harmonics and Approximations on the Unit Sphere: An Introduction*. Springer, Berlin, Heidelberg, 2012. 7

[2] Erik Burman. Cut finite element methods. *Acta Numerica*, 34:1–153, 2025. 2

[3] Erik Burman, Susanne Claus, Peter Hansbo, Mats G. Larson, and Andre Massing. CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015. 2

[4] Eric R. Chan, Connor Z. Monteiro, Petr Kellnhofer, Gordon Wetzstein, and Angjoo Kanazawa. Efficient geometry-aware 3d generative adversarial networks. In *Proc. CVPR*, 2022. Tri-plane 3D GAN (EG3D). 3

[5] Gengxiang Chen, Xu Liu, Qinglu Meng, Lu Chen, Changqing Liu, and Yingguang Li. Learning neural operators on riemannian manifolds, 2023. 2

[6] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes, 2023. 2

[7] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes. In *International Conference on Machine Learning (ICML)*, 2023. 2

[8] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Trans. Graph.*, 40(6), 2021. 1

[9] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics*, 41(4):1–13, 2022. 1

[10] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH*, pages 317–324, Los Angeles, USA, 1999. ACM Press/Addison-Wesley. 2, 6

[11] Gerhard Dziuk and Charles M. Elliott. Finite element methods for surface PDEs. *Acta Numerica*, 22:289–396, 2013. 1, 2, 6

[12] Zhiwei Fang, Justin Zhang, and Xiu Yang. A physics-informed neural network framework for partial differential equations on 3d surfaces: Time-dependent problems, 2021. 1, 2

[13] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018. 6

[14] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3): 339–346, 2002. 1

[15] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy J. Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH*, pages 1–9. ACM, 2022. 1

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 2, 3

[17] Nathan King, Ruben Jones, and Christopher Batty. A closest point method for pdes on manifolds with interior boundary conditions. *ACM TOG*, 2024. 2

[18] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces, 2021. 2

[19] Eberhard Krätzel. *Lattice Points*. Kluwer Academic Publishers, 1988. 5

[20] Christoph Lehrenfeld, Maxim A. Olshanskii, and Xianmin Xu. A stabilized trace finite element method for partial differential equations on evolving surfaces. *SIAM Journal on Scientific Computing*, 2018. 2

[21] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020. arXiv preprint; later versions appeared at ICLR venues. 2

[22] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 6

[23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 1

[24] Lu Lu, Pengzhan Jin, Guofei Pang, Zongqi Zhang, and George E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. 2

[25] Sandip Mazumder. *Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods*. Academic Press, Imprint of Elsevier, London San Diego Waltham, MA Oxford, 2016. 2

[26] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. CVPR*, 2019. 2, 3, 7

[27] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 35–57. Springer, Berlin, Heidelberg, 2003. 2, 6

[28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421. Springer, 2020. 3

[29] Luca Morreale, Noam Aigerman, Vladimir G Kim, and Niloy J Mitra. Neural surface maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2021. 1

[30] Maxim A. Olshanskii and Arnold Reusken. Trace finite element methods for PDEs on surfaces, 2016. 2

[31] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2003. 7

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, 2019. 1, 2, 3, 7

[33] Argyrios Petras, Leevan Ling, and Steven J. Ruuth. An rbf-fd closest point method for solving pdes on surfaces. *Journal of Computational Physics*, 375:1170–1190, 2018. 2

[34] Argyrios Petras, Leevan Ling, and Steven J. Ruuth. A least-squares implicit RBF-FD closest point method and applications to pdes on moving surfaces. *Journal of Computational Physics*, 381:146–161, 2019. 2

[35] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021. 2

[36] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 1, 3, 5

[37] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. NeurIPS*, 2017. 3

[38] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. 2

[39] Steven J. Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3):1943–1961, 2008. 2, 3, 4, 6

[40] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 1, 3, 7

[41] Ryusuke Sugimoto, Nathan King, Toshiya Hachisuka, and Christopher Batty. Projected walk on spheres: A monte carlo closest point method for surface pdes, 2024. 2

[42] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems*. Number 25 in Springer Series in Computational Mathematics. Springer, Berlin, 2nd ed edition, 2006. 1, 2, 6

[43] Romy Williamson and Niloy J. Mitra. Neural geometry processing via spherical neural surfaces. *Eurographics*, 2025. 1, 2, 3, 5, 7

[44] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 42(4), 2023. 1, 3