PoissonNet: A Local-Global Approach for Learning on Surfaces

ARMAN MAESUMI, Brown University, USA
TANISH MAKADIA, Brown University, USA
THIBAULT GROUEIX, Adobe Research, USA
VLADIMIR G. KIM, Adobe Research, USA
DANIEL RITCHIE, Brown University, USA
NOAM AIGERMAN, Université de Montréal, CA

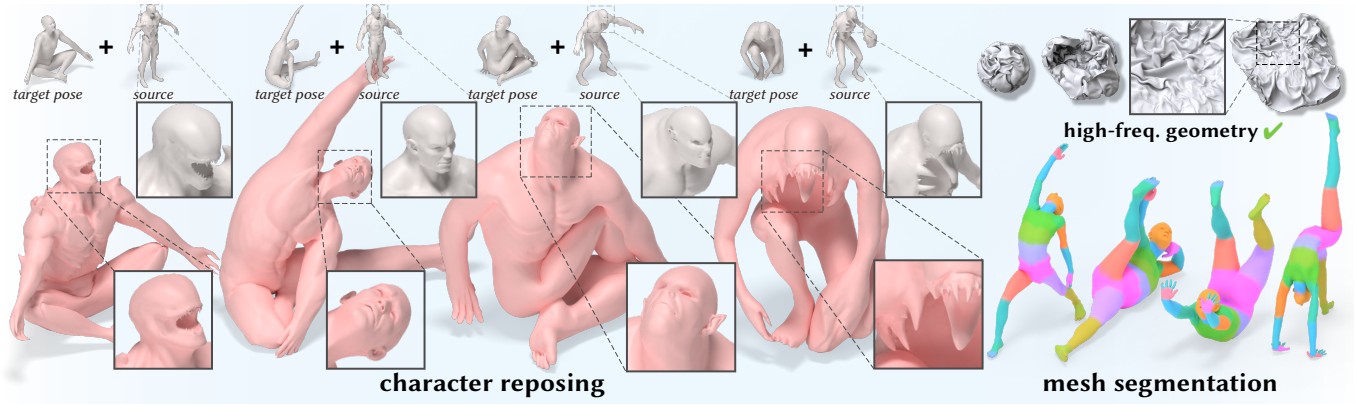


Fig. 1. We develop a general neural architecture for learning on surfaces that uses a local-global construction, resulting in a framework that is highly accurate for processing detailed meshes while being more efficient than comparable methods. *Left:* We train PoissonNet on source-to-target shape deformation for humanoid characters. Our model is able to generalize to in-the-wild geometries while preserving fine details and obeying the provided pose parameters (see reference *target pose* exemplars). *Bottom right:* Our method is general and can be applied broadly to learning tasks on surfaces—e.g. for finely segmenting human bodies. *Top right:* PoissonNet is able to represent extremely high-frequency geometry, such as a crumpling paper ball with 300k faces.

Many network architectures exist for learning on meshes, yet their constructions entail delicate trade-offs between difficulty learning high-frequency features, insufficient receptive field, sensitivity to discretization, and inefficient computational overhead. Drawing from classic local-global approaches in mesh processing, we introduce PoissonNet, a novel neural architecture that overcomes all of these deficiencies by formulating a local-global learning scheme, which uses Poisson's equation as the primary mechanism for feature propagation. Our core network block is simple; we apply learned *local* feature transformations in the gradient domain of the mesh, then solve a Poisson system to propagate scalar feature updates across the surface *globally*. Our local-global learning framework preserves the features's full frequency spectrum and provides a truly global receptive field, while remaining agnostic to mesh triangulation. Our construction is efficient, requiring far less compute overhead than comparable methods, which enables scalability—both in the size of our datasets, and the size of individual training samples. These qualities are validated on various experiments where, compared to previous intrinsic architectures, we attain state-of-the-art performance on semantic segmentation and parameterizing highly-detailed animated surfaces. Finally, as a central application of PoissonNet, we show its ability to learn deformations, significantly outperforming state-of-the-art architectures that learn on surfaces. https://github.com/ArmanMaesumi/poissonnet

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

ACM Reference Format:
Arman Maesumi, Tanish Makadia, Thibault Groueix, Vladimir G. Kim, Daniel Ritchie, and Noam Aigerman. 2025. PoissonNet: A Local-Global Approach for Learning on Surfaces. *ACM Trans. Graph.* 44, 6, Article 1 (December 2025), 16 pages. https://doi.org/10.1145/3763298

1 Introduction

Recent years have seen an explosion in techniques for deep learning on surfaces represented as triangle meshes. As opposed to point clouds and voxel grids, meshes are a representation that can easily encode highly-detailed geometry, provide an appropriate discretization of a Riemannian manifold, encode explicit topological information, and enable computations on the underlying surface (e.g. via finite elements). For these reasons, meshes remain the primary choice of representation for a wide array of applications in graphics.

Current state-of-the-art learning methods for meshes follow an *intrinsic* approach by employing the surface's differential operators (e.g., the Laplacian, gradient, curl, divergence). At each block of the neural network, the differential operators are used to transform intermediate feature fields (e.g. by taking the divergence of a vector

Authors' Contact Information: Arman Maesumi, arman.maesumi@gmail.com, Brown University, USA; Tanish Makadia, tanish_makadia@brown.edu, Brown University, USA; Thibault Groueix, thibault.groueix.2012@polytechnique.org, Adobe Research, USA; Vladimir G. Kim, vova.g.kim@gmail.com, Adobe Research, USA; Daniel Ritchie, daniel_ritchie@brown.edu, Brown University, USA; Noam Aigerman, noamaigerman@gmail.com, Université de Montréal, CA.

ACM 1557-7368/2025/12-ART1
https://doi.org/10.1145/3763298

Method	Full Spectrum	Spatial Support‡	Triangulation Agnostic	Precompute	Inference	Scalable
PoissonNet (ours)	Yes	Global *(inhom.)*	Yes	Single factorization	Fast	Yes
DiffusionNet (direct) [2022]	Yes	Learned *(hom.)*	Yes	Many factorizations	Slow	Mesh-bound
DiffusionNet (spectral) [2022]	Truncated	Learned *(hom.)*	Yes	Eigenbases	Fast	Dataset-bound
DeltaConv† [2022]	Yes	Local + pooling	No	K-NN	Fast	Yes
HodgeNet [2021]	Truncated	Local	Yes	Eigenbases	Slow	Dataset-bound
Harmonic Surface Net [2020]	Yes	Local	No	Parallel transport	Slow	Mesh-bound
MeshCNN [2019]	Yes	Local + pooling	No	N/A	Slow	Mesh-bound

Table 1. A bird's-eye view of trade-offs associated with various methods for learning on surfaces. Columns correspond to: **Full Spectrum**: whether features are propagated in their full frequency spectrum, or spectrally truncated; **Spatial Support**: the effective receptive field of each atomic block in the network; **Triangulation Agnostic**: changes in triangulation produce near-identical outputs; **Precompute**: amount and type of required per-mesh precomputation; **Inference**: per-sample inference latency (ignoring precompute); and **Scalable**: ability to scale up training or model size – affected by amount of precompute (dataset-bound) and inference efficiency (mesh-bound). Green entries indicate desirable extremes, red entries indicate undesirable extremes (spectral truncation, expensive per-mesh precomputation, slow runtime), with intermediate hues reflecting partial trade-offs. **PoissonNet** is the first method to simultaneously encompass: feature propagation in the full eigenspectrum, global spatial support (via a sparse Poisson system that is *efficiently* solvable across our network), agnosticism to mesh discretization, and ability to scale in the number of data samples and mesh size, thereby addressing the fundamental limitations of prior intrinsic architectures. †DeltaConv is a point-based method, though it operates using similar constructs as the other methods in this table. Its entry *"No"* under *Triangulation Agnostic* reflects lack of discretization invariance: local K-NN neighborhoods (and thus the layer's behavior) change with sampling density. ‡ *hom.* and *inhom.* indicate that DiffusionNet's heat equation is homogeneous, whereas Poisson's equation is inhomogeneous (see Section 2 for discussion).

field of features [Wiersma et al. 2022]); alternatively, the differential operators may define a partial differential equation (PDE) whose solution serves as the transformed signal [Gao et al. 2024; Sharp et al. 2022]. This approach provides the means to treat the surface with proper tools from differential geometry, and endows these methods with crucial properties, such as *triangulation agnosticism* (different discretizations of the same shape lead to similar outputs).

There are many ways to incorporate differential operators in a learning framework, often resulting in intricate constructions, which in turn exhibit particular deficiencies that reoccur across all existing methods, e.g., a limited receptive field due to locality of the chosen operators; inability to represent features in their full frequency spectrum; sensitivity to surface discretization; and, expensive computation and memory footprints.

In this paper, we devise a simple and straightforward intrinsic learning approach that overcomes the above issues. We achieve this by formulating our network through *Poisson's equation* — a particular PDE, which we argue is a natural choice for this task. Poisson's equation is one of the most ubiquitous PDEs in graphics, appearing in cornerstone algorithms such as As-Rigid-As-Possible [Sorkine and Alexa 2007], Poisson Surface Reconstruction [Kazhdan et al. 2006], in image editing [Fattal et al. 2002; Pérez et al. 2023], and recently as a proxy to learning deformations' gradients [Aigerman et al. 2022]. Surprisingly, no work has leveraged Poisson's equation for feature learning on meshes; rather, its use has been limited to end-stage "integration" steps common in aforementioned algorithms.

Intuitively, Poisson's equation acts as a bridge between the gradients of signals, and the signals themselves: if the gradient operator transforms signals into their spatial gradients, then Poisson's equation can be seen as its inverse—conceptually akin to an integration operator (see Section 3 for further discussion).

Using this fact, we design a network architecture that alternates between the gradient domain and the functional domain, similar to classic local-global algorithms in graphics. Concretely, each block of our network takes the gradient of the signal (i.e. features), applies *local* learned transformations in the gradient domain, and then solves Poisson's equation to obtain *global* (i.e. not localized in their receptive field) feature updates in the scalar domain, thereby placing gradients as *first class citizens* in our framework.

Operating primarily in the gradient domain is a crucial property for learning over meshes. Indeed, throughout the literature, we observe that previous intrinsic learning methods consistently identify the gradient operator as a critical component of their architectures, with some noting the framework significantly underperforms without gradient features (e.g. Figure 6 in Sharp et al. [2022]).

Whereas prior methods for learning on surfaces trade off along several key properties (see Table 1), our method yields the first network that satisfies them simultaneously:

(1) **Full spectrum.** Our network features retain their native frequency components without any spectral truncation, preserving high-frequency details while avoiding expensive precomputation of eigenbases.
(2) **Global receptive field.** As an integral-like operator, our proposed network block efficiently propagates local feature updates across the entire surface.
(3) **Triangulation agnosticism.** The core mechanism in our network approximates a well-defined object: the continuous Poisson's equation. This allows PoissonNet to produce near-identical predictions under changes in mesh discretization (subdivision, simplification, remeshing, corruption, etc.).
(4) **Efficient computational footprint.** Our method is scalable: PoissonNet can operate on high-resolution meshes and forgo lengthy pre-computation before training and inference, which facilitates training on large datasets.

We empirically validate these properties on a range of canonical applications, such as shape segmentation, deformation, and learning

high-frequency signals on surfaces. In all experiments, PoissonNet achieves state-of-the-art performance, while remaining far more efficient than previous methods with comparable capabilities. Code is available at: https://github.com/ArmanMaesumi/poissonnet

2 Related Work

Learning on surfaces. The first works to apply deep learning to surfaces considered point clouds sampled from the surface, starting with the seminal PointNet [Qi et al. 2017a], later extended to use a local receptive field [Qi et al. 2017b; Qian et al. 2022; Wang et al. 2019] and attention mechanisms [Wu et al. 2024, 2022; Yu et al. 2022; Zhang et al. 2022; Zhao et al. 2021]; see [Guo et al. 2020] for a survey. However, without knowledge of connectivity, point clouds cannot encode highly-detailed geometry, and the network may get confused by nearby points representing geodesically distant regions.

To leverage the surface's connectivity, several approaches aim to generalize the notion of a convolution, either by flattening local patches to the plane [Boscaini et al. 2016; Bronstein et al. 2017; Fey et al. 2018; Masci et al. 2015; Monti et al. 2017; Simonovsky and Komodakis 2017], or via notions of equivariance [De Haan et al. 2020; He et al. 2020; Mitchel et al. 2021; Poulenard and Ovsjanikov 2018; Sun et al. 2020; Wiersma et al. 2020; Yang et al. 2021]. Others treat the mesh as a graph [Simonovsky and Komodakis 2017], or apply a Recurrent Neural Network with random walks on the mesh [Lahav and Tal 2020]. MeshCNN [Hanocka et al. 2019] learns task-specific pooling strategies along with edge collapses. In contrast to our method, these methods are not *triangulation agnostic*, i.e., changes in surface triangulation produce drastically different outputs, causing such networks to 1) learn spurious features that do not generalize to out-of-distribution geometries, and 2) mistakenly couple the mesh's resolution with the network's receptive field. We note that, in this context, triangulation agnosticism does not imply that a given architecture is invariant to all discretizations of an underlying surface; e.g. our method is still subject to discretization error.

Intrinsic learning on meshes. In order to perform triangulation-agnostic learning on surfaces, many works turn to differential operators derived from the meshes themselves, whose constructions are guaranteed to be triangulation agnostic. One popular approach is to leverage the spectral domain, performing a Fourier-like transform using the eigenmodes of the Laplace-Beltrami operator. Several methods leverage the Functional Maps Framework [Ovsjanikov et al. 2012] in a deep learning setting [Attaiki et al. 2021; Donati et al. 2020; Halimi et al. 2019; Litany et al. 2017; Roufosse et al. 2019; Yi et al. 2017]. HodgeNet [Smirnov and Solomon 2021] extends spectral learning to vector fields and area forms. However, the spectral basis is represented as a dense matrix, which has a large memory footprint and is slow to compute (see Table 1). Hence, these methods operate in the low-frequency part of the spectrum (the first k eigenfunctions), hindering their ability to represent high-frequency signals. By contrast, PoissonNet does not rely on a spectral basis, and hence can capture the full frequency spectrum of signals.

DiffusionNet [Sharp et al. 2022] stands as the work closest to ours. Similar to our method, it propagates signals over surfaces by solving a PDE: the homogeneous heat equation. This PDE is often approximated via a single implicit integration step. In a learning framework, however, this becomes highly inefficient, as each learned diffusion time induces a distinct linear system, each needing a factorization that cannot be precomputed. As such, DiffusionNet instead solves its PDE in the spectral domain, restricting to the lower frequency range, which, as explained above, leads to loss of expressivity (see Fig. 4). Since the heat equation acts only as a radially-symmetric filter (see Fig. 6 in their paper), DiffusionNet uses the gradient operator to re-introduce directionality into their filters post-factum. Moreover, the homogeneous heat equation converges to a constant global average at steady state, erasing all structure when propagating features globally. By contrast, we directly solve Poisson's equation, which by construction: offers nontrivial global feature couplings (rather than producing constant signals); incorporates directional features directly into the PDE; and forgoes spectral bases entirely. These properties translate to superior performance and efficiency in several experiments, as discussed in Section 6.

DeltaConv [Wiersma et al. 2022] stands as another close work, as it defines convolutions on surfaces by combining *local* differential operators. However, since the operators are local, these convolutional layers have a local receptive field, which geometrically shrinks as the mesh is refined—requiring a deeper network for propagation of signals across the mesh, implying the method is not agnostic to sampling density. This is in contrast to our method, in which each layer has a global receptive field that approximates a well-defined continuous operation; *i.e.* , different triangulations of the same underlying geometry produce nearly identical outputs, leading to triangulation agnosticism. Additionally, DeltaConv is published as a point-based method, *i.e.* , their operators are defined through K-nearest neighbors, yielding artifacts in our experiments.

Neural Jacobian Fields (NJF) [Aigerman et al. 2022] also stands as inspiration for our method—though it is not a general learning architecture, but rather a method for learning *deformations* of meshes. NJF trains a standard MLP to predict deformation gradients (Jacobians) as a neural field. Poisson's equation is then used in the final layer to produce a mapping from the deformation's gradient. In NJF, Poisson's equation is not a means to propagate learned features over meshes. By contrast, in our method, each consecutive network block solves Poisson's equation to globally propagate learned features over the domain. Additionally, NJF predicts an *extrinsic* signal (i.e. not defined in a local coordinate frame), which is then projected into the mesh's tangent space. This is in contrast to our intrinsic transformations of gradients in the mesh's tangent space, which is critical for learning tasks, as it makes each network block rotation-invariant, leading to more efficient learning and robustness.

Learning Mesh Deformations. As a primary benchmark and application for our network, we show that PoissonNet can serve as a strong backbone for learning mesh deformations. Mesh deformation is a long-standing problem in geometry processing with applications in animation [Sumner and Popović 2004], registration [Bogo et al. 2014a], and geometric modeling [Gao et al. 2019].

Gradient-domain computation commonly appears in this scenario, using maps' Jacobians for surface parameterization [Aigerman and Lipman 2013; Du et al. 2020; Kovalsky et al. 2014; Lévy et al. 2002; Li et al. 2018; Lipman 2012; Liu et al. 2008; Myles and Zorin 2013; Rabinovich et al. 2017; Schüller et al. 2013; Smith and Schaefer 2015;

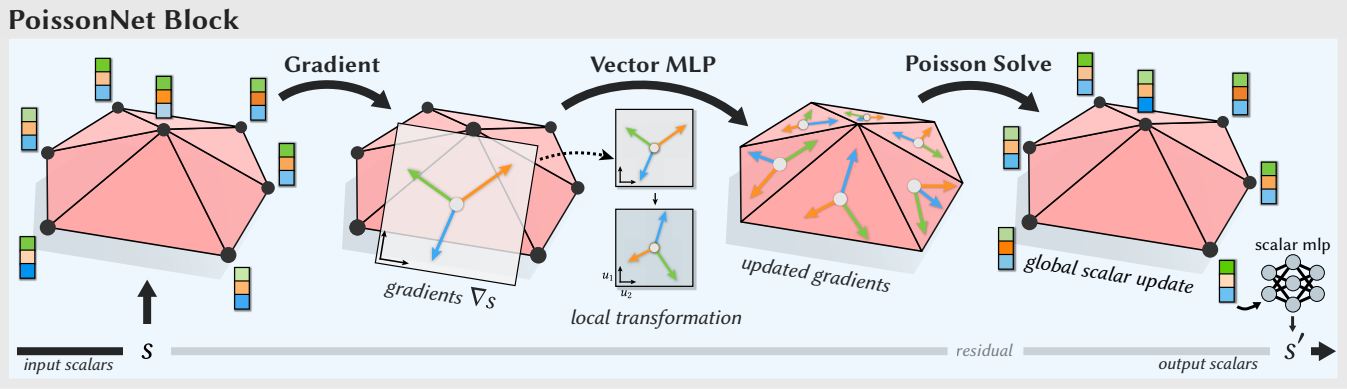


Fig. 2. We illustrate our *PoissonNet block* on a surface patch of a triangle mesh. Our block begins by computing spatial gradients of the incoming scalar features—for demonstrative purposes we depict the signal as having three channels (shown as green, orange, blue) with color indicating signal intensity. The gradient features are transformed locally in each tangent basis (denoted by basis vectors u_1, u_2) via a Vector MLP, which induces a linear combination of rotated and scaled gradients on each face. We then solve a global Poisson system using the transformed vector fields, producing new scalar features on vertices that are updated locally using a scalar (per-vertex) MLP. This process is repeated for N blocks, thereby producing a final feature representation on the shape.

Weber and Zorin 2014] or for deformation [Lipman et al. 2004; Sorkine et al. 2004; Sumner and Popović 2004; Yu et al. 2004].

Many other ways to parameterize deformations as *rigs* have been developed over the years [Fulton et al. 2019; Jacobson et al. 2011, 2014; Ju et al. 2005; Kavan et al. 2008; Lipman et al. 2008], which have been used in a deformation-learning setting, e.g., using skeletons [Holden et al. 2015; Li et al. 2021; Liu et al. 2025; Xu et al. 2020, 2019], handles [Liu et al. 2021], or cages [Sun et al. 2024; Wang et al. 2020]. Some methods combine rig-driven deformation with non-linear residuals per discrete element [Bailey et al. 2020, 2018; Romero et al. 2021; Yin et al. 2021; Zheng et al. 2021].

Alternatively, for given template meshes [Bogo et al. 2014a; Osman et al. 2020; Varol et al. 2017; Zuffi et al. 2017] one can directly predict a fixed-size tensor of vertex coordinates [Anguelov et al. 2005a; Bogo et al. 2016; Shen et al. 2021], possibly by directly learning the deformations' gradients [Gao et al. 2018; Tan et al. 2018].

We evaluate PoissonNet as a backbone, used together with a final layer provided by NJF [Aigerman et al. 2022], discussed above, as it provides a triangulation-agnostic method for predicting deformations, which has since proven highly effective in scenarios such as temporal sequences [Muralikrishnan et al. 2024], face rigging [Qin et al. 2023], and text/image-driven generative deformation [Gao et al. 2023; Kim et al. 2025; Yoo et al. 2024].

3 Preliminaries

Tangent spaces and local coordinates. We consider meshes with vertices $\mathbf{V}$ and triangles $\mathbf{F}$. Each triangle $\mathbf{t} \in \mathbf{F}$ defines its own *tangent space*, denoted $T_\mathbf{t}$ — a 2-dimensional linear subspace of $\mathbb{R}^3$, consisting of all vectors tangent to $\mathbf{t}$. We choose an (arbitrary) orthonormal basis $\mathcal{B}_\mathbf{t} = \{U_1, U_2\}$, $U_i \in \mathbb{R}^3$ which serves as the tangent space's *local coordinate frame*: any vector $v \in \mathbb{R}^3$ that lies on triangle $\mathbf{t}$ is a *tangent vector*, which can be written equivalently as a 2-vector, $\tilde{v} \in \mathbb{R}^2$ in the local coordinate system of $\mathcal{B}_\mathbf{t}$, as the unique vector satisfying $\sum_i \tilde{v}_i U_i = v$. While we derive intuition from the geometric 2-d tangent plane, we will alternatively treat it as the complex plane $\mathbb{C}$, with each tangent vector defined as $v \in \mathbb{C}$.

Piecewise linear functions and their gradients. Hence, we follow the standard definition of piecewise linear functions (i.e. linear finite elements): we consider functions that are scalars assigned to the mesh's vertices. We denote such a function as $s \in \mathbb{R}^{|\mathbf{V}|}$, with s_i being the scalar value associated with vertex i. Such scalar values on the vertices of a single triangle, s_i, s_j, s_k uniquely define an affine function $a(p) : \mathbf{t} \to \mathbb{R}$ over the triangle, which interpolates these three values at the triangle's vertices, i.e., $a(v_i) = s_i$. Thus, the signal s defines a piecewise-linear function over the triangles of the mesh, i.e., it is a linear function when restricted to one of the triangles. Therefore, its gradient is constant within the triangle and is a tangent vector that we denote (in the local coordinate system $\mathcal{B}_\mathbf{t}$) as $f \in \mathbb{R}^2$. The gradient $f_\mathbf{t}$ of a specific triangle $\mathbf{t}$ can be obtained from the vertex values by applying the linear gradient operator which we denote $\nabla_\mathbf{t}$, i.e., $f_\mathbf{t} = \nabla_\mathbf{t}(s_i, s_j, s_k)$. We use the notation f to refer to the tensor of stacked gradient vectors over all the faces of the mesh. Finally, we consider multiple simultaneous signals, called *channels*, $s^1, s^2, ..., s^c$, where each channel s^i is a scalar field, whose corresponding gradient field is f^i.

Poisson's equation. As with many graphics applications that operate in the gradient domain, our method relies on the variational formulation of Poisson's equation. Given a set of tangent vectors f, the variational perspective of the Poisson equation finds *the scalar function whose gradient best matches* f. In the continuous setting, this translates to the following least squares variational objective

$$u = \min_s \int_\Omega ||\nabla s - f||^2 dA. \tag{1}$$

Since our domain is a *mesh*, f is constant over each face. Hence, the above integrand becomes constant over each triangle, making the least squares problem reduce to a sparse linear system (see Eq. 5).

4 PoissonNet

Our method builds on the insight that using Poisson's equation as the core mechanism in an intrinsic learning architecture leads to several desirable properties. Poisson's equation can be solved *efficiently*

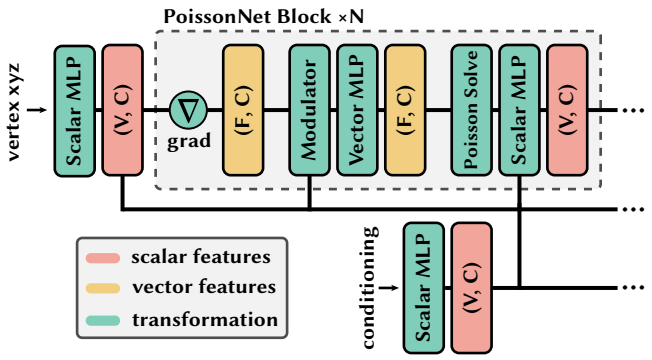


Fig. 3. PoissonNet stacks identical blocks that transform features locally in the gradient domain and globally in the functional domain.

without resorting to lossy spectral approximations while serving as a truly global operator over the surface. These characteristics make PoissonNet a strong backbone for learning highly detailed signals on surfaces, where spectral approximations fail; learning deformations, where global operators are necessary; and common semantic tasks, such as mesh segmentation.

Our architecture is comprised of *PoissonNet blocks* (depicted in Figures 2 and 3), which interleave two simple operations on the incoming scalar features s:

(1) *Local step in the gradient domain (Sec 4.1).* Compute the gradient of the incoming signal, ∇s, and locally (per-face) apply a learned transformation, producing gradient field f.
(2) *Global step (Sec 4.2).* Solve Poisson's equation using f, inducing a scalar update, u, that globally couples features on all vertices. Finally, output a learned transformation on s and u.

In the following sections we describe these steps in detail.

4.1 Local step in the gradient domain

Given a scalar feature field with C channels defined on the vertices of a mesh, $s \in \mathbb{R}^{|V|\times C}$, we first compute its corresponding gradient field, $f \in \mathbb{C}^{|F|\times C}$, using the intrinsic gradient operator

$$f \coloneqq \nabla s. \tag{2}$$

These vector-valued quantities reside in the tangent space of each triangle, which is denoted by an orthonormal basis $\{u_1, u_2, n\}$ where u_1 and u_2 span the tangent plane and n is the face normal. We express these gradients as a complex number $z = a + bi$ with a and b being the coefficients of u_1 and u_2, and hence transformations of these quantities can be represented by products with complex numbers, which induces a scaling and rotation within the tangent plane.

As is common when learning transformations of vector-valued quantities, we parametrize gradient transformations with learned complex weight matrices [Gao et al. 2024; Sharp et al. 2022; Wiersma et al. 2020, 2022]. Geometrically, the transformed gradient features become linear combinations of rotated and scaled gradients at each face. Notably, the choice of basis at each face is arbitrary up to a rotation (i.e. any orthonormal basis in the triangle can be chosen). Hence, it is desirable for our gradient transformation to maintain equivariance under linear coordinate transformations. As proposed by Wiersma et al. [2020], we apply non-linearities to gradient magnitudes only, which preserves equivariance by ensuring that the gradient's directional component transforms consistently with the underlying coordinate system — i.e. the phase of each gradient feature transforms linearly. The gradient transformation can be written succinctly on the i-th face as

$$\mathbf{f}_i \leftarrow \mathbf{W}\mathbf{f}_i \odot \frac{\sigma(\mathbf{r}+\mathbf{b})}{\mathbf{r}} \tag{3}$$

where $\mathbf{f}_i \in \mathbb{C}^C$ are the incoming gradient features on the face, and $\mathbf{W} \in \mathbb{C}^{C\times C}$ is a learned complex weight matrix. We write $\mathbf{r} \in \mathbb{R}^C$ as the vector holding magnitudes of each element of $\mathbf{W}\mathbf{f}_i$. Finally, $\mathbf{b} \in \mathbb{R}^C$ is a learned bias parameter, and σ denotes a non-linearity; $\odot$ and division by $\mathbf{r}$ are taken elementwise.

To enrich our gradient transformations, we additionally use the original scalar signal, s, to modulate the incoming gradients (before the transformation above). This allows the network to more discriminately transform gradient features using the scalar signal as intrinsic positional information. Let $s_{\text{face}} \in \mathbb{R}^{F\times C}$ denote the scalar features averaged onto faces. We modulate the phase and magnitude of gradient features via the element-wise product

$$f \leftarrow (\sigma(\boldsymbol{\gamma}) + \epsilon) f e^{i\boldsymbol{\theta}} \quad \text{where } \boldsymbol{\gamma}, \boldsymbol{\theta} = \text{MLP}(s_{\text{face}}) \tag{4}$$

where the scale factors $\boldsymbol{\gamma}$ and angular rotations $\boldsymbol{\theta}$ are given by a small multi-layer perceptron acting point-wise on s_{face}. We apply a softplus activation, σ, to the scale factors; adding a small epsilon ensures positivity.

4.2 Globally propagating gradient features

Once gradient-domain features have been transformed, we integrate them back into scalar features via a Poisson solve. Concretely, let $\mathbf{f} \in \mathbb{R}^{2F\times C}$ represent the stacking of components of all face-based transformed gradient features. We recover a global update to the scalar features by solving on each channel the sparse linear system

$$\mathbf{L}u = \nabla^T \mathbf{M} f, \tag{5}$$

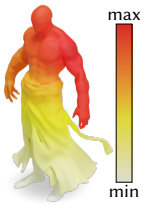
where $\mathbf{L}$ is the cotangent Laplacian [Pinkall and Polthier 1993], $\mathbf{M}$ is the mesh's mass matrix, and ∇^T represents the divergence operator. Since the solution u is unique only up to an additive constant, we nullify its mean, centering the solution at zero. Divergence being a coordinate-free operator means that the Poisson solution is invariant to the choice of tangent bases. Finally, we apply a point-wise MLP to the concatenation of the input features s and Poisson solution u. The feature update on the i-th vertex becomes

$$s_i \leftarrow \text{MLP}([s_i, u_i, c_i]). \tag{6}$$

where c_i are experiment-specific conditional features (see Sec. 6).

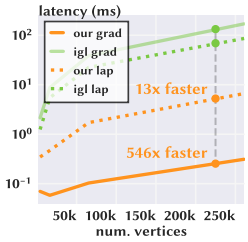
Since the Poisson equation is an elliptic PDE, it can be solved efficiently without approximate timestepping or spectral methods, and its discretization admits a single pre-factorable sparse linear system that can be reused across all network blocks. These qualities allow PoissonNet to 1) efficiently scale, both in the size of datasets and the meshes themselves (i.e. the number of vertices); and 2) forgo lossy spectral approximations that are used in previous methods.

Remark. Applying Poisson's equation to our network's gradient features is analogous to that of a global attention mechanism with fixed geometry-dependent weights. The inverse Laplacian $\mathbf{L}^{-1}$ implicitly defines a Green's function $G(i, j)$ that weights the contribution of the divergence at vertex j to the update at vertex i. In this view, $G(i, j)$ serves as a global attention kernel over the mesh, aggregating gradient-domain signals from all triangles into each vertex's scalar feature update—the inset visualizes G w.r.t a vertex on the character's left shoulder. This construction is efficient, in that the attention kernel is defined through sparse mesh operators (rather than materializing a quadratic attention matrix), and has the added benefit of naturally adapting to the underlying mesh geometry.



5 Implementation

Efficient construction of operators. We employ a custom PyTorch CUDA extension for the construction of our discrete mesh operators; namely the Laplacian, gradient, and mass matrices, allowing our training pipeline to forgo lengthy precomputation and instead compute necessary operators efficiently on-the-fly during training. This greatly reduces friction in experimentation and allows our method to be applied directly to large datasets. Notably, our method does not require precomputing a Laplacian eigenbasis, which often relies on CPU-based generalized eigendecomposition routines that are too slow to use during training and may take several hours to precompute even for moderately sized datasets. These qualities make PoissonNet more practical for large scale training and rapid experimentation, and additionally more flexible in pipelines with non-static training examples (e.g. when applying data augmentation to meshes). The inset compares our CUDA kernels against LibIGL [Jacobson et al. 2018]. Our CUDA kernels emit sparse mesh operators directly using PyTorch's COO representation and support batching for homogeneous meshes (i.e. those with identical connectivity structure).



Solving Poisson's equation. We discretize the Poisson equation as in Equation 5. Our Poisson systems are solved using a shared Cholesky factorization of the Laplacian matrix, $\mathbf{L}$, across all network blocks and channels, and hence the simultaneous per-channel linear systems are efficiently solved in parallel. We use Cholespy [Nicolet et al. 2021], a CUDA-based sparse linear solver. Our Laplacian uses zero Neumann boundary conditions. Following Poisson's variational form (Eq. 1), the inhomogeneous Neumann condition, $\partial u/\partial n = f \cdot n$, appears naturally, with n being the outward boundary normal. The Poisson solution, u, is centered to obtain a unique solution.

6 Results and Experimentation

In the following section, we evaluate PoissonNet on several applications, comparing it to current state-of-the-art in learning on meshes. We focus on methods that perform intrinsic learning using differential operators, as the limitations of previous approaches have been demonstrated. See Section 2 for a full discussion of these methods.

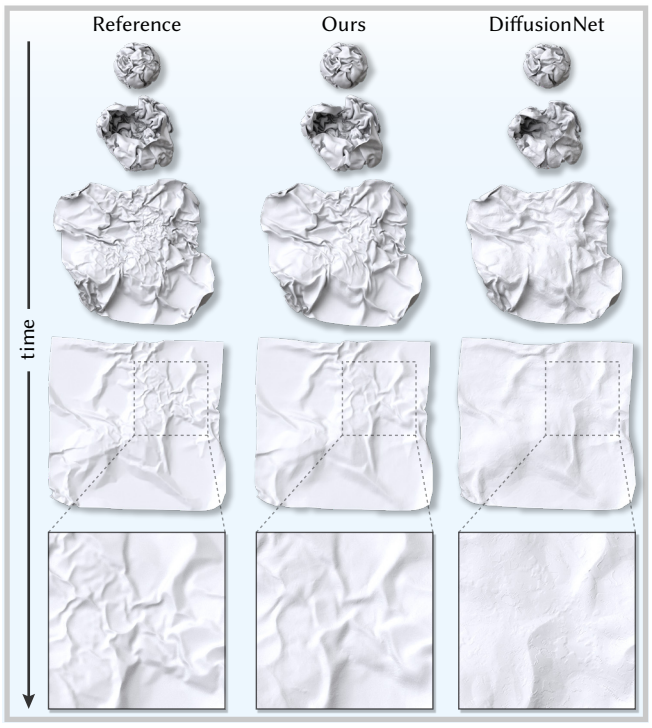
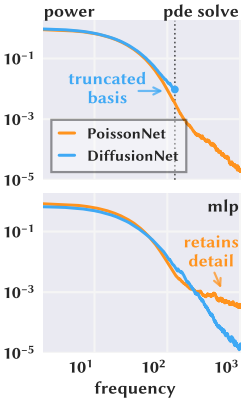


Fig. 4. Comparison of architectures for representing highly-detailed signals. The networks are used to parametrize the evolution of a crumpling paper through time w.r.t. a rest configuration (see supplemental video). The mesh has 300k triangles and is available on TurboSquid [IndefinGaming 2025].

Experimental setup. Across experiments we employ the same PoissonNet with varying numbers of network blocks, using 128-width blocks in all experiments. We use xyz vertex coordinates as our input features unless otherwise specified. Data augmentation is applied when applicable; we specifically augment shape orientation and global scale. Our experiments primarily compare to DiffusionNet [Sharp et al. 2022] and DeltaConv [Wiersma et al. 2022], as they are leading methods for learning on surfaces using differential operators. We include further details in Section A.

6.1 Analysis of Full-Spectrum Learning

Figure 4 demonstrates that our method is capable of representing extremely rich geometric signals. Here, we train PoissonNet to represent the evolution of an animated crumpling paper ball that has 300k faces [IndefinGaming 2025]. We parametrize the sequence by a scalar time t, which is used to condition the input of each network block's MLP (as in Eq. 6). To further challenge our method, we only use a total of ~650k parameters, whose memory footprint (i.e., compression ratio) constitutes 2% percent of the original sequence size; nevertheless, our method manages to preserve most of the fine details of the geometry.



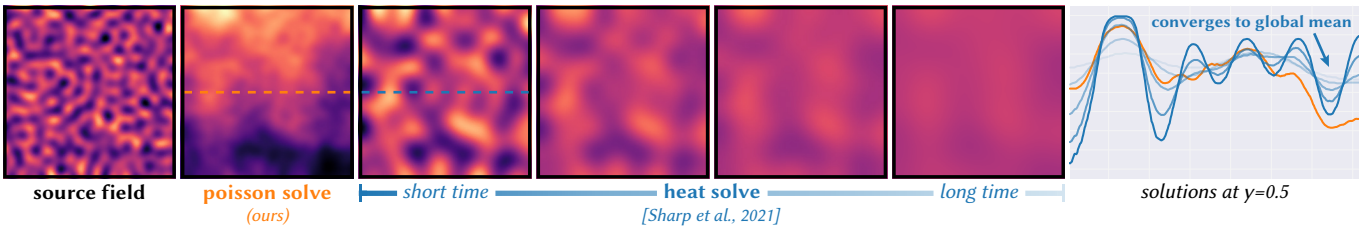
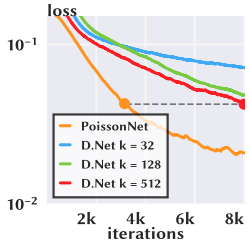


Fig. 5. Poisson and heat solutions on an input scalar field. The Poisson equation (inhomogeneous) retains structure; whereas the heat equation (homogeneous) converges to a global mean, losing all structure over long-time heat flow. *Right plot*: solution values along a cross-section at $y = 0.5$ (dotted lines).

We compare our method's performance to that of DiffusionNet [Sharp et al. 2022] by training it with the same number of parameters. Due to the limitations of the heat equation discussed in Section 2 and Figure 5, DiffusionNet exhibits clear loss of detail and over-smoothing, struggling to represent the high-frequency wrinkles of the crumpled paper. The first inset further compares power spectra of the feature maps learned by both networks, confirming that our network is able to use higher frequency features to represent the evolving geometry, while DiffusionNet encounters the expected issues that arise from the use of the heat equation. Additionally, the inset loss plot shows the clear effect of DiffusionNet's eigenbasis size (denoted by k) on training dynamics, as compared to our method. Both methods employ the *NJF* head described in Section 6.2.



6.2 Shape Deformation

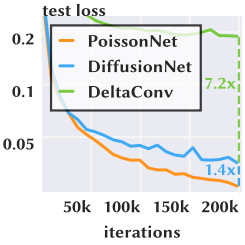
We demonstrate that PoissonNet is capable of accurate, global reasoning, by learning to repose arbitrary humanoid character models without canonical poses or rigs, which requires global understanding of input geometry (joint articulation is an inherently long-range phenomenon, acting on kinematic chains). In particular, we accrue a dataset of 16k source-target mesh pairs generated by the SMPL-X human body model, using poses from the MOYO dataset [Pavlakos et al. 2019; Tripathi et al. 2023]. These poses are comprised of motion-captured yoga sequences—containing "pretzel-like" contortions of human bodies, which are significantly more challenging to repose than traditional body poses. We deform a given source mesh into the target pose, conditioning the network on the SMPL-X pose parameters of the target. Our network uses five PoissonNet blocks, totaling 1.4 million parameters.

To conduct a fair comparison between network backbones, we employ the *NJF* deformation head proposed by Aigerman et al. [2022], which is a state-of-the-art method for parametrizing deformations. Briefly, the *NJF* head receives as input three gradient fields associated with the gradients of the deformation map's x, y, z components—i.e., a per-face Jacobian, $J_i \in \mathbb{R}^{2\times 3}$. The *NJF* head then produces a final deformation by solving Poisson's equation w.r.t the predicted Jacobians. We modify each architecture to predict the necessary Jacobian field (see Section A.2 for details), and supervise the predicted deformations using *NJF*'s proposed loss,

$$\mathcal{L}_{\text{NJF}} = \sum m_i \cdot \|v_i^{\text{tar}} - u_i\|^2 + \sum \alpha_i \cdot \|J_i^{\text{tar}} - \nabla u_i\|^2, \qquad (7)$$

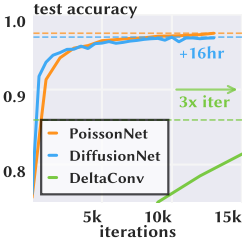
where m, α hold lumped vertex masses and face areas, and u is the solution to Equation 5. The ground truth vertex positions and Jacobians are denoted as v^{tar} and $J^{\text{tar}} := \nabla v^{\text{tar}}$ respectively.

In Figures 1 and 12 we show qualitative results of our method, and Figure 6 compares PoissonNet to that of a DiffusionNet backbone. Our network not only faithfully captures deformations of SMPL-X human bodies, but also boasts remarkable generalization to in-the-wild character models. We find that DiffusionNet is unable to retain high-frequency details in these shapes, often distorting their hands and faces (see Figure 6). The inset plot reflects a similar conclusion: our model converges more quickly and reaches a much lower loss than the alternative backbones. We note that DeltaConv was unable to converge to a meaningful result on this benchmark, likely due to its KNN-based differential operators, which are unreliable for surfaces with nearly-touching parts (e.g. in yoga poses). Finally, we show that PoissonNet is even able to transfer motion capture sequences to out-of-distribution characters, generating realistic motion sequences (see Figure 7 and supplemental video).



6.3 Semantic Segmentation

We demonstrate that PoissonNet surpasses state-of-the-art performance on semantic segmentation of meshes, while remaining far more efficient than previous methods. In particular, we train a 3-block PoissonNet (650K parameters) on segmentations of the yoga motion capture shapes used in Section 6.2 (totaling 32k training samples). We use the canonical SMPL-X segmentation map to delimit 27 unique body parts, distinguishing symmetric body parts (e.g. *left*/*right* forearm are separate classes). Our network achieves 97.03% test accuracy; DiffusionNet achieves 96.12% while requiring an additional *16 hours of precomputation and 160gb of memory overhead due to the need for eigenbases*; and DeltaConv attains 88.2% but is unable to reliably distinguish between left/right-sided parts, likely due to its local construction (see inset accuracy plot). Each method shows negligible variability in peak accuracy between runs ($< 0.5\%$). We additionally train our network on the human body dataset of Maron et al. [2017]. This dataset is an amalgamation of human meshes obtained from various sources



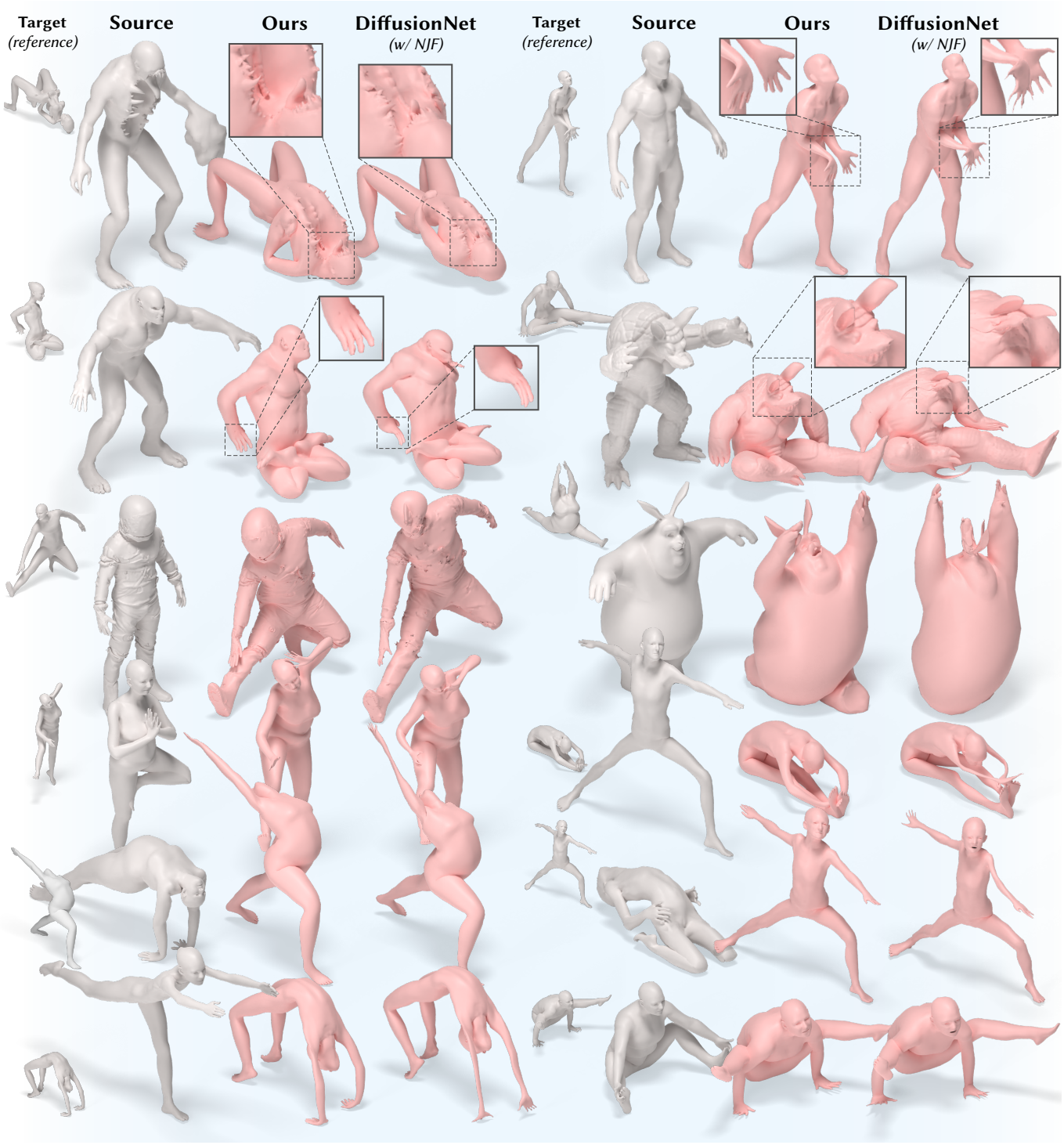


Fig. 6. We demonstrate that PoissonNet surpasses state-of-the-art intrinsic learning backbones for representing source-to-target deformations of humanoid characters. Here, we compare our deformations to that of DiffusionNet [Sharp et al. 2022], using diverse poses from the MOYO dataset [Tripathi et al. 2023]. PoissonNet is able to faithfully match target poses while retaining intricate surface details from the source geometry, even for models that are substantially out of distribution (e.g. the top left mutant, and the Buck Bunny). We observe that DiffusionNet is unable to retain surface details, causing many body parts to become distorted (see close-ups). We additionally show in-distribution examples in the bottom half of the figure—of which these distortions remain noticeable.

Table 2. Comparison of methods on SHREC11 shape classification.

Method	Accuracy
MeshCNN[Hanocka et al. 2019]	91.0%
HSN[Wiersma et al. 2020]	96.1%
MeshWalker[Lahav and Tal 2020]	97.1%
PD-MeshNet[Milano et al. 2020]	99.1%
HodgeNet[Smirnov and Solomon 2021]	94.7%
FC[Mitchel et al. 2021]	99.2%
DiffusionNet[Sharp et al. 2022]	99.5%
DeltaConv[Wiersma et al. 2022]	99.6%
PoissonNet (ours)	100.0%

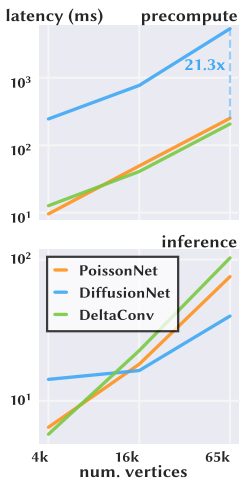
[Adobe 2016; Anguelov et al. 2005b; Bogo et al. 2014b; Giorgi et al. 2007; Vlasic et al. 2008]. The meshes are segmented into eight unique body parts. We report our test-time accuracy in Table 4 alongside many previous methods as they were reported by Wiersma et al. [2022]. Our network matches state-of-the-art performance on this benchmark. Predicted segmentation maps for both datasets are shown in Figures 1 and 13.

6.4 Classification

We train PoissonNet on the SHREC11 shape classification benchmark [Lian et al. 2011], which contains 30 categories of shapes with 20 shape variations in each category. We employ a 3-block PoissonNet, identical to that of Section 6.3. Following previous methods [Ezuz et al. 2017; Hanocka et al. 2019; Sharp et al. 2022; Wiersma et al. 2022], we train and test on simplified meshes, using just 10 examples per class for training, and averaging peak test accuracy over five training runs. Our network achieves a perfect accuracy of 100% on the held out samples. Results are summarized in Table 2.

6.5 Analysis of Architectural Properties

Training & Inference Efficiency. PoissonNet's construction is efficient, making it straightforward to apply to large datasets and meshes with tens of thousands of vertices. Our method circumvents costly precomputation while being accurate and maintaining high throughput. These benefits extend to PoissonNet's forward latency on large meshes. In Table 3, we compare the training efficiency of our network with previous state-of-the-art methods on the experiment detailed in Section 6.2. Additionally, the inset figures compare the latency of these networks on meshes of increasing size. PoissonNet provides the best trade-off between precompute time, throughput, and accuracy. For fair comparison, we endow DiffusionNet with our CUDA kernels for precomputing Laplacian and gradient operators.



We additionally compare the memory footprint of our Cholesky factorizations to methods that rely on Laplacian eigenbases. Although Cholesky factorization, in general, leads to possibly dense triangular matrices, we empirically find that our Cholesky factors scale within reasonable constant factors of $O(n \log n)$ nonzero elements, with n being the number of mesh vertices and constant factors ranging from 3 to 6. For example, the Armadillo mesh subdivided to $n = 800k$ vertices yields a decomposition requiring <200MB memory when accounting for sparse matrix storage overhead. A moderately-sized eigenbasis on the same mesh yields comparable memory usage; e.g., $k = 128$ eigenvectors requires $n \cdot k \approx 390$MB.

Table 3. Compute efficiency on our reposing experiment, alongside single-mesh forward latency. Our method has minimal precompute while maintaining high throughput and accuracy. By comparison, *DiffusionNet (spectral)*'s eigenbasis precompute takes several hours and demands large memory overhead, while *DiffusionNet (direct)* is too expensive to train. For fair comparison, both PoissonNet and DiffusionNet use our operator CUDA kernels.

Total compute expenditure			
Method	**Precompute**	**Train**	**Total**
PoissonNet	< 1 min	9058 batch/hr	22.1hr
DiffusionNet *(spectral)*	+9hr +80GB mem.	11438 batch/hr	26.5hr
DeltaConv	< 1 min	9015 batch/hr	22.2hr
Single mesh forward latency (*precompute+inference* in ms)			
Method	**4k verts**	**16k verts**	**65k verts**
PoissonNet	9.6+6.5ms	49.4+18.2ms	251+75.9ms
DiffusionNet *(spectral)*	245+14.2ms	770+16.4ms	5340+39.9ms
DeltaConv	12.7+5.8ms	40.8+22.7ms	207+103ms

Robustness. In Fig. 8 we show that PoissonNet's predictions remain stable under changes to triangulation; e.g. corruption, simplification, subdivision, and partial surfaces. Additionally, PoissonNet is more robust to surface holes compared to DiffusionNet, which introduces significant distortion around the hole (see inset). We attribute this to properties of Poisson's equation discussed by Stein et al. [2018]. Namely, the natural boundary conditions that appear in our inhomogeneous Poisson equation indicate that our feature fields will not become overly distorted around surface holes.

Learning local signals. Given that PoissonNet's spatial filter is a purely global operator, one may wonder if the network is suitable for learning local, multi-scale signals; e.g., could PoissonNet predict the heat kernel signatures (HKS) [Sun et al. 2009] on a given shape? We indeed find that our network is capable of learning such signals by training a 3-block PoissonNet on heat kernel signatures computed on shapes from the Thingi10k dataset [Zhou and Jacobson 2016] that have been pre-processed by fTetWild [Hu et al. 2020]. We retain all shapes that contain a single connected component and have more than 500 vertices, leaving 7500 shapes. We use an 85-15 train-test split. For each shape, we sample 16 HKS feature fields using time values logarithmically spaced in the interval $[0.01, 1]$. Each channel

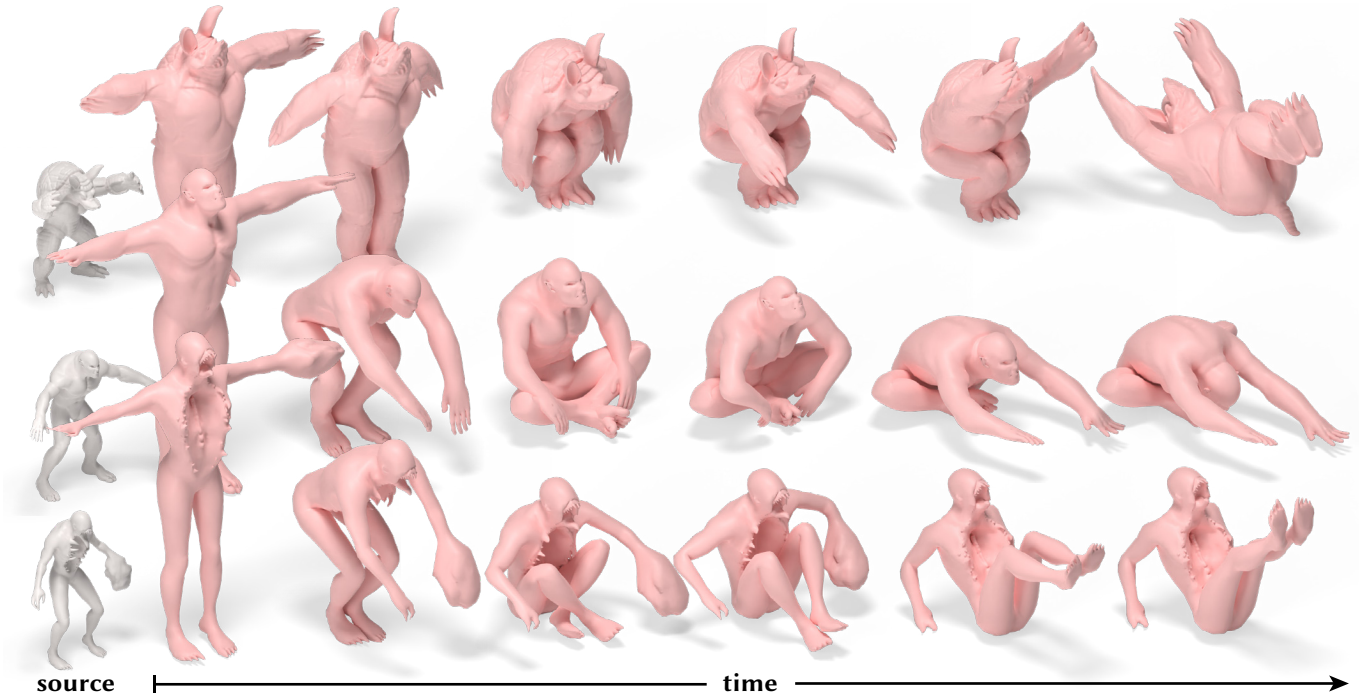


Fig. 7. Our reposing network learns a smooth parametrization of humanoid poses that generalizes to out-of-distribution shapes. We repose several characters using pose sequences from the MOYO dataset [Tripathi et al. 2023] and observe smoothly varying, realistic movements in accordance with that of a human.

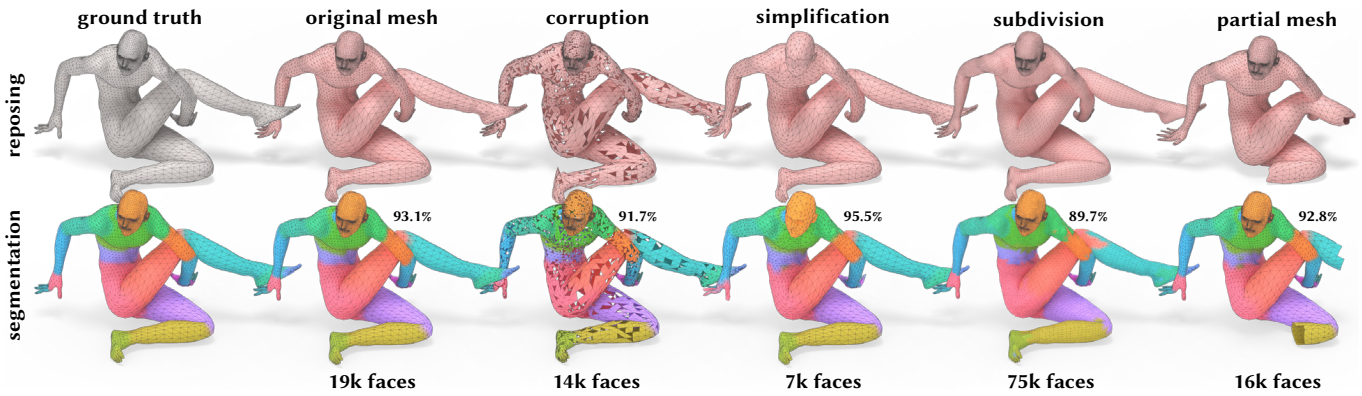


Fig. 8. Our network is robust to changes in discretization. We apply our networks to a mesh under various such changes and observe stable predictions. From left to right: ground truths, our baseline prediction; and our predictions under corruption, quadric decimation, subdivision, and a partial (incomplete) mesh. For segmentations we provide accuracies w.r.t. the ground truth—in the case of topological changes, we use nearest-vertex matching to compute accuracy.

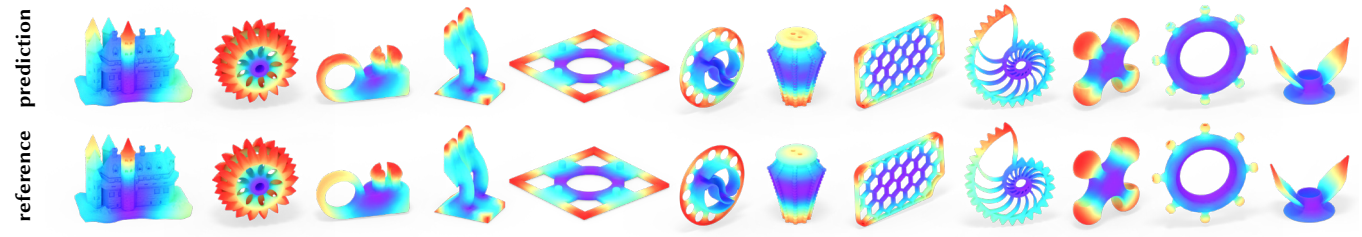


Fig. 9. Outputs from our PoissonNet trained on multi-scale heat kernel signatures. Despite using a purely global operator, PoissonNet is able to capture HKS fields across a diverse array of shapes. All shapes shown are from the held out test set.

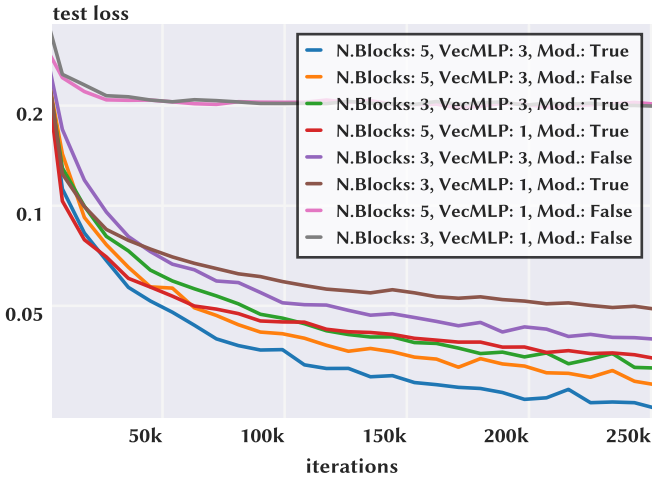


Fig. 10. Hyperparameter sweep over various PoissonNet configurations for our shape deformation experiment. *N.Blocks* denotes number of PoissonBlocks, *VecMLP* denotes the number of layers in each block's Vector MLP, and *Mod* denotes the usage of our proposed vector feature modulation step given by Equation 4, which is applied before each Vector MLP.

is independently normalized to $[0, 1]$ to stabilize learning; hence, our HKS predictions are meaningful up to a global scaling. Our network predicts all feature maps simultaneously and is supervised using a standard MSE loss. Figure 9 shows that PoissonNet is able to represent heat kernel signatures on a diverse array of out-of-distribution shapes.

Ablating design choices. Figure 10 shows the effect of various design decisions with respect to the shape deformation experiment in Section 6.2. We observe that, all else being equal, employing our modulation step (Eq. 4) improves network performance; similarly, using larger Vector MLPs (more layers) and more PoissonNet blocks steadily improves performance. The blue curve represents our primary network used in all results.

7 Conclusion

We have demonstrated that PoissonNet's local-global approach sidesteps several trade-offs associated with previous intrinsic approaches — resulting in a network that is efficient, scalable, and robust to out-of-distribution geometry. It serves as a practical tool enabling several learning applications, including animation of intricately detailed character models (without rigs), semantic segmentation, and compression of high-frequency geometry.

Limitations. Our method is not well suited for learning on shapes with multiple connected component; as our intrinsic Poisson equation will operate indendently on each component. This drawback is also shared by other intrinsic methods based on differential operators [Sharp et al. 2022; Smirnov and Solomon 2021; Wiersma et al. 2020]. The utility of these networks would be greatly expanded if they generalized to multi-component meshes, which dominate the corpus of in-the-wild models. Facilitating this learning in a principled way, without ad-hoc KNN-based solutions, remains an unsolved problem. Moreover, extremely poor discretizations (e.g. sliver triangles) may lead to numerical instability for each of these approaches; e.g., by distorting the solution to Poisson's equation. Next, although PoissonNet requires less computational resources than comparable methods, it is still too slow to enable real-time applications on very large meshes. Finally, while each block of PoissonNet has global support, it has a decaying receptive field; i.e., the coupling between vertices diminishes with distance. We consider the investigation of more general classes of signal propagation using different PDEs, such as the *wave equation*, as important future work.

Acknowledgments

This material is based upon work supported by: National Science Foundation Graduate Research Fellowship under Grant No. 2040433; NSERC Discovery grant RGPIN-2024-04605, "Practical Neural Geometry Processing"; FRQNT Établissement de la relève professorale 365040, "Calcul rapide et léger des déformations à l'aide de réseaux neuronaux"; and a gift from Adobe. Part of this work was done while Arman Maesumi was an intern at Adobe Research. The authors thank Qingnan Zhou for providing preprocessed Thingi10k data, as well as Nick Sharp, Alec Jacobson, and Derek Liu for fruitful discussions.

References

Adobe. 2016. Adobe Mixamo 3D characters. www.mixamo.com..

Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. 2022. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *SIGGRAPH* (2022).

Fig. 11. Feature maps given by the Poisson solves across a pre-trained three block PoissonNet. We visualize a fixed channel for all feature maps and find their qualitative appearance to be similar across shapes.

Noam Aigerman and Yaron Lipman. 2013. Injective and bounded distortion mappings in 3D. 32, 4 (2013).

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005a. SCAPE: Shape Completion and Animation of People. In *SIGGRAPH*.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005b. SCAPE: Shape Completion and Animation of People. *ACM Trans. Graph.* 24, 3 (2005), 408–416.

Souhaib Attaiki, Gautam Pai, and Maks Ovsjanikov. 2021. Dpfm: Deep partial functional maps. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 175–185.

Stephen W Bailey, Dalton Omens, Paul Dilorenzo, and James F O'Brien. 2020. Fast and deep facial deformations. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 94–1.

Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. 2018. Fast and Deep Deformation Approximations. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 119:1–12. doi:10.1145/3197517.3201300 Presented at SIGGRAPH 2018, Los Angeles.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*. Springer International Publishing.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014a. FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014b. FAUST: Dataset and evaluation for 3D mesh registration. *CVPR* (2014).

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems* 29 (2016).

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. 2020. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425* (2020).

Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. 2020. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8592–8601.

Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Trans. Graph.* (2020).

Danielle Ezuz, Justin Solomon, Vladimir G. Kim, and Mirela Ben-Chen. 2017. GWCNN: A Metric Alignment Layer for Deep Shape Analysis. *Computer Graphics Forum* 36, 5 (2017), 49–57. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13244 doi:10.1111/cgf.13244

Raanan Fattal, Dani Lischinski, and Michael Werman. 2002. Gradient domain high dynamic range compression. *ACM Trans. Graph.* 21, 3 (July 2002), 249–256. doi:10.1145/566654.566573

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. 2018. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 869–877.

Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin, and Alec Jacobson. 2019. Latent-space Dynamics for Reduced Deformable Simulation. *Computer Graphics Forum* (2019).

Alexander Gao, Maurice Chu, Mubbasir Kapadia, Ming C Lin, and Hsueh-Ti Derek Liu. 2024. An intrinsic vector heat network. *arXiv preprint arXiv:2406.09648* (2024).

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic Unpaired Shape Deformation Transfer. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2018)* 37, 6 (2018), To appear.

Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. 2019. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–15.

William Gao, Noam Aigerman, Thibault Groueix, Vladimir G. Kim, and Rana Hanocka. 2023. TextDeformer: Geometry Manipulation using Text Guidance. *SIGGRAPH (Conference track)* (2023).

Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. 2007. Shape retrieval contest 2007: Watertight models track. *SHREC competition* 8, 7 (2007), 7.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence* 43, 12 (2020), 4338–4364.

Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. 2019. Surface networks via general covers. (2019), 632–641.

Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. 2019. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4370–4379.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Trans. Graph.* 38, 4, Article 90 (July 2019), 12 pages. doi:10.1145/3306346.3322959

Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. 2020. CurvaNet: Geometric deep learning based on directional curvature for 3D shape analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2214–2224.

Daniel Holden, Jun Saito, and Taku Komura. 2015. Learning an Inverse Rig Mapping for Character Animation. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. Association for Computing Machinery, New York, NY, USA, 165–173. doi:10.1145/2786784.2786788

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (July 2020), 18 pages. doi:10.1145/3386569.3392385

IndefinGaming. 2025. Animated Paper Crumpling. https://www.turbosquid.com/3d-models/animated-paper-crumpling-1794996.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78.

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH 2014 Courses*.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Siggraph 2005 Papers* (2005), 561–566.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)* 27, 4 (2008), 1–23.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Goslar, DEU, 61–70.

Hyunwoo Kim, Itai Lang, Thibault Groueix, Noam Aigerman, Vladimir G. Kim, and Rana Hanocka. 2025. MeshUp: Multi-Target Mesh Deformation via Blended Score Distillation. *3DV* (2025).

Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2014. Controlling singular values with semidefinite programming. *ACM Trans. Graph.* (2014).

Alon Lahav and Ayellet Tal. 2020. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–13.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. Vol. 21. 362–371.

Minchen Li, Danny Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint Optimization of Surface Cuts and Parameterization. *SIGGRAPH Asia* (2018).

Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. 2021. Learning Skeletal Articulations with Neural Blend Shapes. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1.

Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. 2011. SHREC'11 track: shape retrieval on non-rigid 3D watertight meshes. In *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval* (Llandudno, UK) *(3DOR '11)*. Eurographics Association, Goslar, DEU, 79–88.

Yaron Lipman. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.* (2012).

Yaron Lipman, David Levin, and Daniel Cohen-Or. 2008. Green coordinates. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–10.

Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.P. Seidel. 2004. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.* 181–190.

Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. 2017. Deep Functional Maps: Structured Prediction for Dense Shape Correspondence. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. 2025. RigAnything: Template-Free Autoregressive Rigging for Diverse 3D Assets. *arXiv preprint arXiv:2502.09615* (2025).

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Computer Graphics Forum*. 1495–1504.

Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. 2021. DeepMetaHandles: Learning Deformation Meta-Handles of 3D Meshes with Biharmonic Coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12–21.

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4, Article 71 (July 2017), 10 pages. doi:10.1145/3072959.3073616

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings*

of the IEEE international conference on computer vision workshops. 37–45.
MeshyAI. 2025. . https://www.meshy.ai/.
Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-Dual Mesh Convolutional Neural Networks. 33 (2020), 952–963. https://proceedings.neurips.cc/paper_files/paper/2020/file/0a656cc19f3f5b41530182a9e03982a4-Paper.pdf
Thomas W. Mitchel, Vladimir G. Kim, and Michael Kazhdan. 2021. Field Convolutions for Surface CNNs. (October 2021), 10001–10011.
Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5115–5124.
Sanjeev Muralikrishnan, Niladri Dutt, Siddhartha Chaudhuri, Noam Aigerman, Vladimir Kim, Matthew Fisher, and Niloy J Mitra. 2024. Temporal Residual Jacobians for Rig-Free Motion Transfer. (2024), 93–109.
Ashish Myles and Denis Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Trans. Graph.* 32, 4 (2013).
Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021). doi:10.1145/3478513.3480501
Ahmed A A Osman, Timo Bolkart, and Michael J. Black. 2020. STAR: A Sparse Trained Articulated Human Body Regressor. In *European Conference on Computer Vision (ECCV)*. 598–613. https://star.is.tue.mpg.de
Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–11.
Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 10975–10985.
Patrick Pérez, Michel Gangnet, and Andrew Blake. 2023. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 577–582.
Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36.
Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Trans. Graph.* 37, 6, Article 236 (Dec. 2018), 14 pages. doi:10.1145/3272127.3275102
Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.
Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).
Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 23192–23204. https://proceedings.neurips.cc/paper_files/paper/2022/file/9318763d049edf9a1f2779b2a59911d3-Paper-Conference.pdf
Dafei Qin, Jun Saito, Noam Aigerman, Groueix Thibault, and Taku Komura. 2023. Neural Face Rigging for Animating and Retargeting Facial Meshes in the Wild. In *SIGGRAPH 2023 Conference Papers*.
Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 4 (2017).
Cristian Romero, Dan Casas, Jesus Perez, and Miguel A. Otaduy. 2021. Learning Contact Corrections for Handle-Based Subspace Dynamics. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)* 40, 4 (2021). http://gmrv.es/Publications/2021/RCPO21
Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. 2019. Unsupervised deep learning for structured shape matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1617–1627.
Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* (2013).
Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)* 41, 3 (2022), 1–16.
Siyuan Shen, Yin Yang, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. 2021. High-order differentiable autoencoder for nonlinear model reduction. *ACM Transactions on Graphics*.
Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3693–3702.
Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: learning spectral geometry on triangle meshes. *ACM Trans. Graph.* 40, 4, Article 166 (July 2021), 11 pages. doi:10.1145/3450626.3459797
Jason Smith and Scott Schaefer. 2015. Bijective parameterization with free boundaries. *ACM Trans. Graph.* 34, 4 (2015).
Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*.
O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Nice, France) *(SGP '04)*. 175–184.
Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018. Natural Boundary Conditions for Smoothing in Geometry Processing. *ACM Trans. Graph.* 37, 2, Article 23 (May 2018), 13 pages. doi:10.1145/3186564
Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)* 23, 3 (2004), 399–405.
Bo Sun, Thibault Groueix, Chen Song, Qixing Huang, and Noam Aigerman. 2024. TutteNet: Injective 3D Deformations by Composition of 2D Mesh Deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21378–21389.
Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. 2009. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 1383–1392.
Zhiyu Sun, Ethan Rooke, Jerome Charton, Yusen He, Jia Lu, and Stephen Baek. 2020. Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 204–216.
Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018. Variational Autoencoders for Deforming 3D Mesh Models. In *CVPR*.
Shashank Tripathi, Lea Müller, Chun-Hao P. Huang, Taheri Omid, Michael J. Black, and Dimitrios Tzionas. 2023. 3D Human Pose Estimation via Intuitive Physics. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 4713–4725. https://ipman.is.tue.mpg.de
Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. 2017. Learning from Synthetic Humans. In *CVPR*.
Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. 2008. Articulated Mesh Animation from Multi-View Silhouettes. *ACM Trans. Graph.* 27, 3 (2008), 97.
Yifan Wang, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural Cages for Detail-Preserving 3D Deformations. In *CVPR*.
Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)* 38, 5 (2019), 1–12.
Ofir Weber and Denis Zorin. 2014. Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph.* (2014).
Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. 2020. CNNs on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)* 39, 4 (2020).
Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. 2022. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–10.
Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024. Point Transformer V3: Simpler Faster Stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4840–4851.
Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. 2022. Point Transformer V2: Grouped Vector Attention and Partition-based Pooling. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 33330–33342. https://proceedings.neurips.cc/paper_files/paper/2022/file/d78ece6613953f46501b958b7bb4582f-Paper-Conference.pdf
Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. 2020. RigNet: Neural Rigging for Articulated Characters. *ACM Trans. on Graphics* 39 (2020).
Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. 2019. Predicting Animation Skeletons for 3D Articulated Models via Volumetric Nets. In *2019 International Conference on 3D Vision (3DV)*.
Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. 2021. Continuous Geodesic Convolutions for Learning on 3D Shapes. 134–144. doi:10.1109/WACV48630.2021.00018
Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. 2017. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 2021. 3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations. In *Proceedings of International Conference on Computer Vision (ICCV)*.
Seungwoo Yoo, Kunho Kim, Vladimir G. Kim, and Minhyuk Sung. 2024. As-Plausible-As-Possible: Plausibility-Aware Mesh Deformation Using 2D Diffusion Priors. *CVPR* (2024).
Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 19313–19322.

Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2004. Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH 2004 Papers*. 644–651.

Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. 2022. PVT: Point-voxel transformer for point cloud learning. *International Journal of Intelligent Systems* 37, 12 (2022), 11985–12008. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.23073 doi:10.1002/int.23073

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. 2021. Point Transformer. (October 2021), 16259–16268.

Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbic. 2021. A Deep Emulator for Secondary Motion of 3D Characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5932–5940.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 2017. 3D Menagerie: Modeling the 3D Shape and Pose of Animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

A Experimental and Implementation Details

Below we provide details for all experiments outlined in the main text, including the employed PoissonNet architectures, details of compared methods, and training hyperparameters.

A.1 Analysis of Full-Spectrum Learning

We train a 128-width PoissonNet with two blocks using an additional *NJF* head. Each block uses a VectorMLP of two layers. We train our network on all frames of the crumpling paper ball dataset (see Sec. C.2) using a learning rate of 0.001 and virtual batch size of 8. The employed DiffusionNet baseline uses similar parameters; however, for fair comparison we endow DiffusionNet with wider blocks, using a width of 192 to equalize the total number of parameters, and an *NJF* head is provided. As with our shape deformation experiment, we validated the baseline with and without an NJF head—in this particular experiment, we found the results to be moderately better with the head included. The paper crumpling sequence is parametrized by a single scalar time value, which is appended to the scalar MLPs of both networks.

Analysis of network features. In Section 6.1 we compare the power spectra of PoissonNet's learned features compared to that of DiffusionNet. In particular, we extract feature maps from the last network block (before the appended *NJF* heads) for both networks, and project these features into an eigenbasis given by the crumpled paper's Laplace-Beltrami operator. The eigenbasis has size $K = 1024$. Let $\mathbf{f} \in \mathbb{R}^{K \times C}$ denote the projection of a feature map with C channels into the spectral basis. We compute the power spectra per-channel c, p_k^c, over all $\mathbf{f}_k$ with $k \in [1, K]$ via, $p_k^c = |\mathbf{f}_k^c|^2$, normalizing the power channel-wise, $\hat{p}_k^c = p^c / \sum_j^K p_j^c$. Finally, we compute the maximum channel-wise power, taking that as our final power spectrum. The first inset in Section 6.1 shows power spectra for feature maps produced by the networks' PDEs and MLP layers respectively. Our method produces features with more power in the high-frequency range, notably our MLP's features retain higher frequencies in the band that is truncated by DiffusionNet's spectral PDE solve.

Table 4. Comparison of methods on the human mesh segmentation task of Maron et al. [2017]. Table is provided by Wiersma et al. [2022].

Method	Accuracy
PointNet++ [Qi et al. 2017b]	90.8
MDGCNN [Poulenard and Ovsjanikov 2018]	88.6
DGCNN [Wang et al. 2019]	89.7
SNGC [Haim et al. 2019]	91.0
HSN [Wiersma et al. 2020]	91.1
MeshWalker [Lahav and Tal 2020]	**92.7**
CGConv [Yang et al. 2021]	89.9
FC [Mitchel et al. 2021]	92.5
DiffusionNet - xyz [Sharp et al. 2022]	90.6
DiffusionNet - hks [Sharp et al. 2022]	91.7
DeltaConv [Wiersma et al. 2022]	92.2
PoissonNet - xyz	90.7
PoissonNet - hks	91.1

A.2 Shape Deformation

In our shape deformation experiments we use a 128-width PoissonNet with five blocks, using VectorMLPs with three layers. We train using a learning rate of 0.0005 and batch size of 16. The employed DiffusionNet baseline uses equal number of blocks with a larger width of 192 to equalize parameter count. We experimented with various configurations for DeltaConv, which we discuss in a later paragraph. Finally, all networks are endowed with an additional *NJF* head to parametrize the deformation—we experiment with alternative choices, discussed in paragraph *NJF Head* below.

Experimental setup. We employ our full dataset of 16k source-target SMPL-X human body pairs with poses sourced from the MOYO dataset [Pavlakos et al. 2019; Tripathi et al. 2023]—see Section C.1 for details. Each network is conditioned on all 153 SMPL-X pose parameters—parameters for eyes are excluded. We experimented with two choices for injection of these parameters into the tested backbones: 1) by simply included these parameters in the input layer of the networks, 2) by concatentation of the parameters in each network block's MLP. We found that the second option is superior for all backbones. We employ standard data augmentation techniques to further improve generalization; in particular, we apply random shifts and scalings to the training shapes. In the case of DiffusionNet, which uses precomputed eigenbases, we are mindful to update them accordingly when scaling training shapes dynamically.

NJF Head. We experiment with two parametrizations of the deformation map: 1) a direct prediction of the target vertices (i.e. networks predict the deformed *xyz* coordinates directly), and 2) appending an *NJF* head to the end of each network. We find that using an *NJF* head is superior across all backbones, especially for generalization to out-of-distribution geometries. We implement identical *NJF* heads for each method. Concretely, the head expects input features on vertices, which are then mapped to corresponding gradient features on faces using the intrinsic gradient operator. We transform these features using a Vector MLP that maps the input gradient features into a 2-vector field for each of the target coordinate channels (i.e. *x,y,z*)—these serve as the predicted mapping's Jacobian fields, which are then integrated using Poisson's equation to produce the final deformed vertices. We additionally add the source shape's *xyz* gradients to the *NJF* head's predicted Jacobians, serving as a skip connection in the gradient domain. We re-iterate that identical constructions were tested across backbones in these experiments.

DeltaConv Baseline. We employed various configurations for the DeltaConv backbone used in Section 6.2, ranging from networks that matched our parameter count of ~1.5 million, to deeper networks totaling ~2.5 million parameters. We additionally, disabled the BatchNorm layers in DeltaConv's MLPs, as they hindered deformation performance. However, we found that all configurations were unable to converge to a comparable result as compared to our method and DiffusionNet. We attribute this to two properties inherent in DeltaConv's construction: 1) the use of point-based K-NN surface operators, which are extremely noisy for surfaces that have near-touching geometry. This is especially relevant in our case, as our dataset contains yoga poses, where ground truth geometry is often "kissing", or even intersecting; 2) DeltaConv's network blocks

Fig. 12. We further demonstrate that PoissonNet can be used to repose characters that are produced by generative models. Here, we show a diverse selection of characters generated by MeshyAI [2025] that have been reposed using PoissonNet with pose inputs from the MOYO dataset [Tripathi et al. 2023].

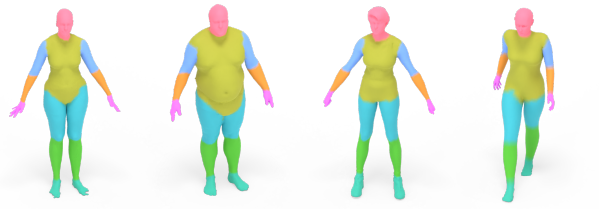


Fig. 13. PoissonNet's segmentations on the dataset of Maron et al. [2017].

use *local* mesh operators, whereas our method features fully global support in each network block. For this reason, we experimented with DeltaConv backbones that were comparatively deeper (i.e. to enlarge its effective receptive field). However, we note that doing so only marginally improved performance, and in some cases actually hindered performance due to over-parametrization. We note that the employed DeltaConv architecture did not use spatial downsampling or upsampling layers (as in a UNet-style network), as this architecture variant was not available in the open source code. Using spatial pooling may improve performance, as it would effectively broaden the network's receptive field; However, critical limitations would remain (i.e. sensitivity to sampling density).

A.3 Semantic Segmentation on MOYO Dataset

We train a 128-width PoissonNet with three blocks, using VectorMLPs with a single layer. We train using a learning rate of 0.001 and batch size of 16. The DiffusionNet baseline uses blocks of 176 width to equalize parameter count, and the DeltaConv baseline was configured similarly. The inset loss plot in Section 6.3 shows that our method converges to a higher test accuracy, while being far more compute efficient than the compared methods. Namely, DiffusionNet required an additional 16 hours of precompute and 160gb of memory overhead to compute and store the dense eigenbases over the entire segmentation dataset. Meanwhile, DeltaConv required 3 times as many training iterations to converge to a lower final test accuracy. The total wallclock time of our method was only 2 hours.

B Ablating design decisions

In Figure 10 we show validation loss curves for various hyperparameter configurations on our shape deformation experiment. We specifically highlight that our vector feature modulation step (given by Eq. 4) clearly improves performance on this benchmark. This gain in performance is representative across all of our experiments.

C Creation of Datasets

C.1 MOYO Dataset

We employ the MOYO dataset [Tripathi et al. 2023] in our shape deformation and semantic segmentation experiments (see Sections 6.3 and 6.2). In particular, we generate a dataset of 32k training and 4k validation SMPL-X [Pavlakos et al. 2019] human bodies using yoga poses sampled from MOYO. Because MOYO contains temporal motion captures, there are many near-identical frames that are redundant for our experiments; hence, we employ a greedy farthest point sampling routine to select a subset of the poses that are most distinct. We then generate human bodies by randomly sampling SMPL-X body shapes—we aggressively sample body parameters using a Gaussian distribution with a standard deviation of 5. This provides body models with more diverse geometry, which aids in generalization. The dataset is generated with pairs of source-target models with identical body shapes. For our semantic segmentation experiment, we use the canonical SMPL-X vertex segmentation map. In particular, we use all 27 classes—we exclude the eyes/eyeball classes. We use all 32 thousands training examples in both experiments; for shape deformation this constitutes 16 thousand source-target pairs.

C.2 Crumpling Paper Ball

We use the simulated crumpling paper ball published by IndefinGaming [2025] to benchmark our network. The sequence is comprised of 118 frames, each with a canonical mesh topology of 300k faces.