DiffusionNET: Discretization Agnostic Learning on Surfaces

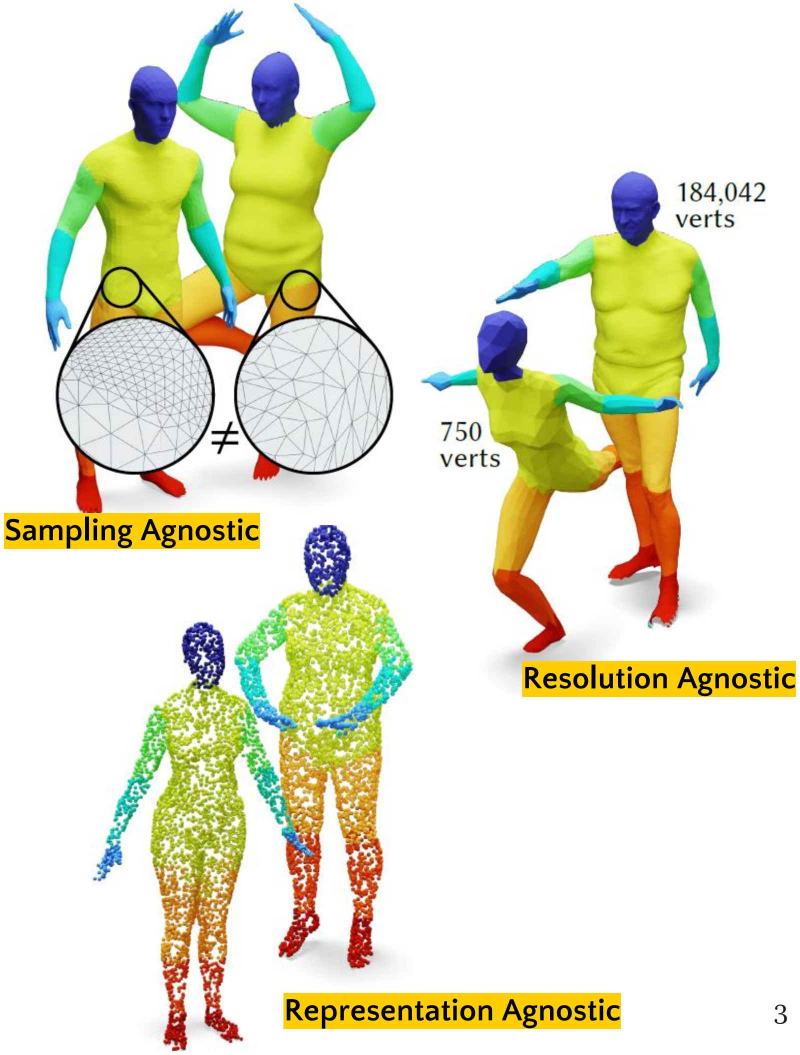
Recent Advances in 3D Computer Vision

Til Stotz (03757560) Til.Stotz@tum.de Robotics, Cognition, Intelligence WS22/23

Table of Contents

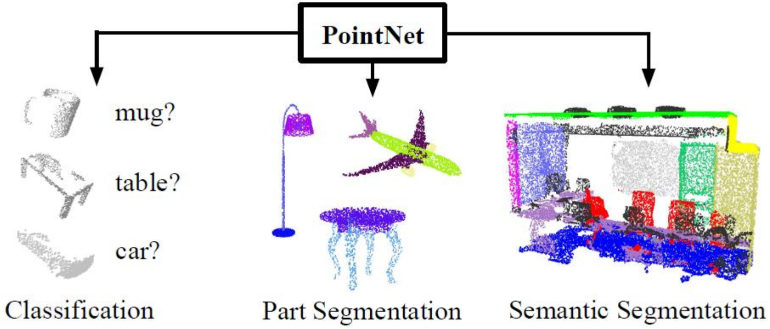
01 What is DiffusionNET?

- Plug-and-play neural network for deep learning on 3D surfaces
- Simple, robust and efficient networks
- Ingredients: standard MLP, diffusion and spatial gradient features



02 Related Work: (1) PointNET

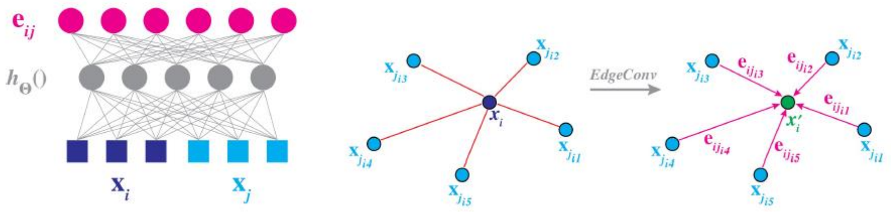
- Neural network architecture for point sets
- High performance on classification, segmentation
- Drawbacks:
  - **Point clouds only**
  - **Not suitable for deformable shapes**

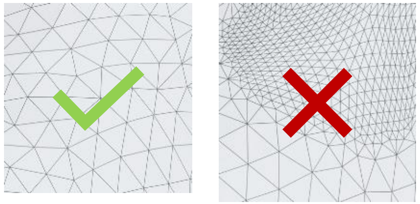


Source: Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1, 2 (2017), 4.

02 Related Work: (2) Dynamic Graph CNN

- CNN-based neural network
- Propagates information along edges
- Drawbacks:
  - **Bad performance under remeshing**
  - **Worse propagation on dense point clouds**

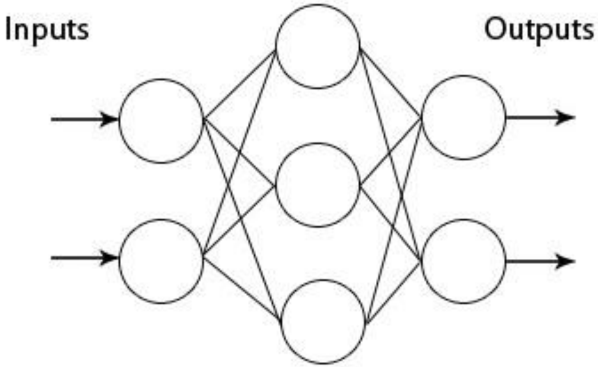




Source: Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. ACM Transactions on Graphics (TOG) (2019).

03 Method: (1) Multi-layer Perceptron

- Consider D features at every vertex V
- Transform features for all vertices via $f\colon \mathbb{R}^D \to \mathbb{R}^D$ using **standard MLP**
- Can‘t capture spatial structure, no communication between vertices



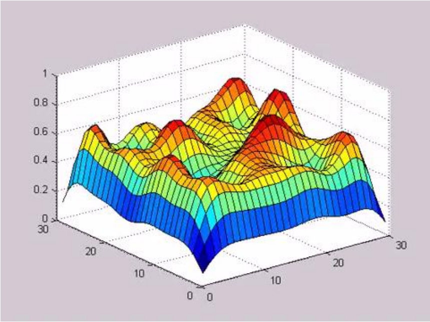
03

Method: (2) What is Diffusion?

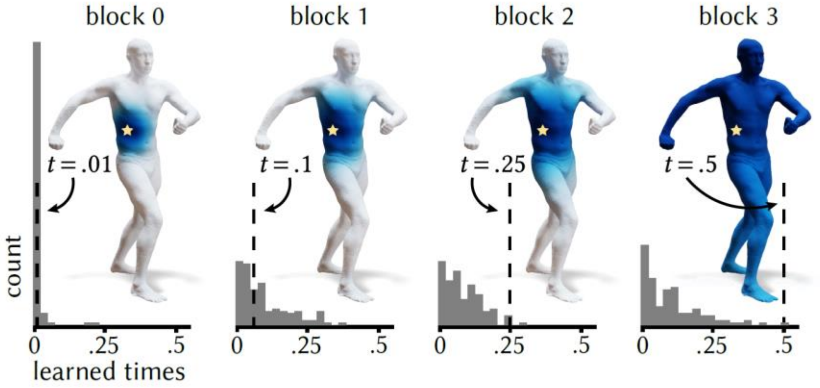
- Purpose: Spatially propagate features
- Intuition: Use **heat equation** to model diffusion

Continuous	Discrete
$\frac{d}{dt} u_t = \triangle u_t$	$\frac{d}{dt} u_t = -M^{-1} L u_t$ $M \in \mathbb{R}^{VxV}, L \in \mathbb{R}^{VxV}$

- Learn diffusion time t
- Implemented by diffusion layer



https://www.youtube.com/watch?v=twBcpxrWm5E



03 Method: (2) Computing Diffusion

Euler methods

- Easy to implement
- Solve sparse linear systems
- **May not scale well to large problems**

$$h_t(u) = (M + tL)^{-1} M u$$

Spectral acceleration

- Based on Laplacian eigenfunctions
- Diffusion reduces to **elementwise exponentiation**

$$h_t(u) = \Phi \begin{bmatrix} e^{-\lambda_0 t} \\ e^{-\lambda_1 t} \\ \dots \end{bmatrix} \odot (\Phi^T M u)$$

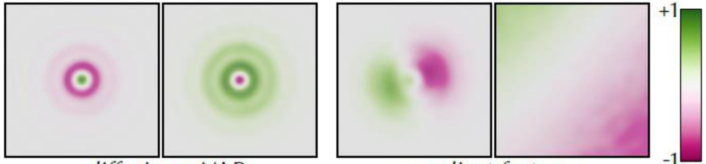
03 Method: (2) Computing Diffusion ctd.

1. Calculate eigenvalue decomposition of L, M: $L\phi_i = \lambda_i M \phi_i$
2. Stack eigenvectors in Φ
3. Project u into spectral basis, express diffusion, reproject:

$$h_t(u) := \underbrace{\Phi}_{\text{Reprojection}} \underbrace{\begin{bmatrix} e^{-\lambda_0 t} \\ e^{-\lambda_1 t} \\ \dots \end{bmatrix}}_{\text{Calculate diffusion for time t}} \odot (\underbrace{\Phi^T M u}_{\text{Project into spectral basis}})$$

03

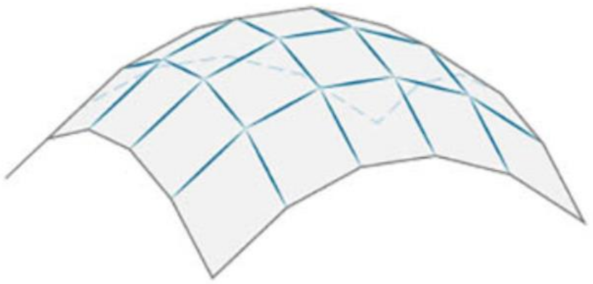
Method: (3) Spatial Gradient Features



diffusion + MLP + gradient features

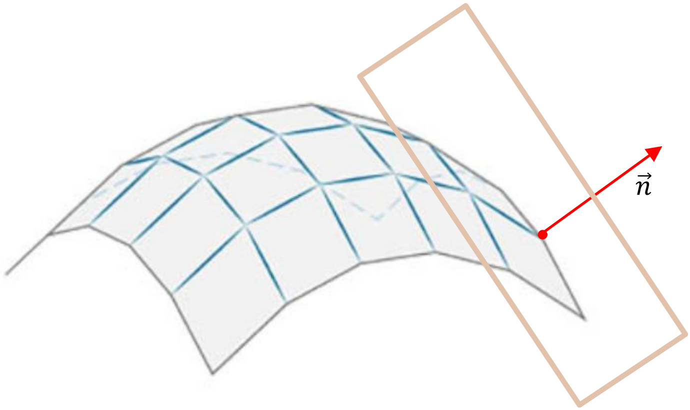
- Diffusion layer: Radially-symmetric filters only
- Spatial gradient features: **Directional filters**
- Procedure:
  - Precompute gradients
  - Learn features from dot product between gradient pairs
  - Gradients first undergo learned scaling/rotation

03 Method: (3) Spatial Gradient Features



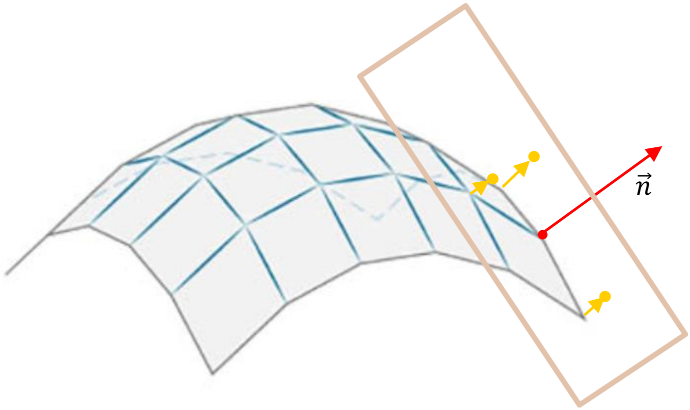
- Given a mesh or point cloud with V vertices, for all V:

03 Method: (3) Spatial Gradient Features



- Given a mesh or point cloud with V vertices, for all V:
  - Define tangent plane

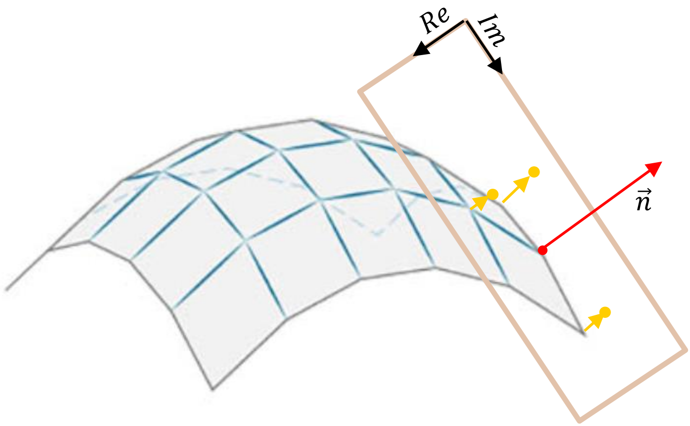
03 Method: (3) Spatial Gradient Features



- Given a mesh or point cloud with V vertices, for all V:
  - Define tangent plane
  - Project neighboring vertices

03

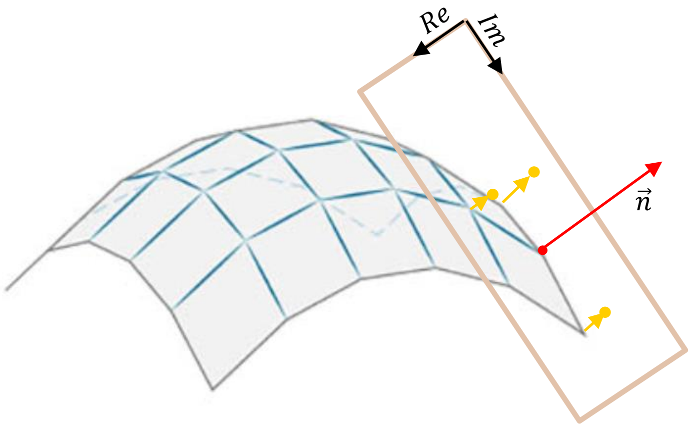
Method: (3) Spatial Gradient Features



- Given a mesh or point cloud with V vertices, for all V:
  - Define tangent plane
  - Project neighboring vertices
  - Estimate gradient in tangent plane
  - Express gradients as complex number

03

Method: (3) Spatial Gradient Features



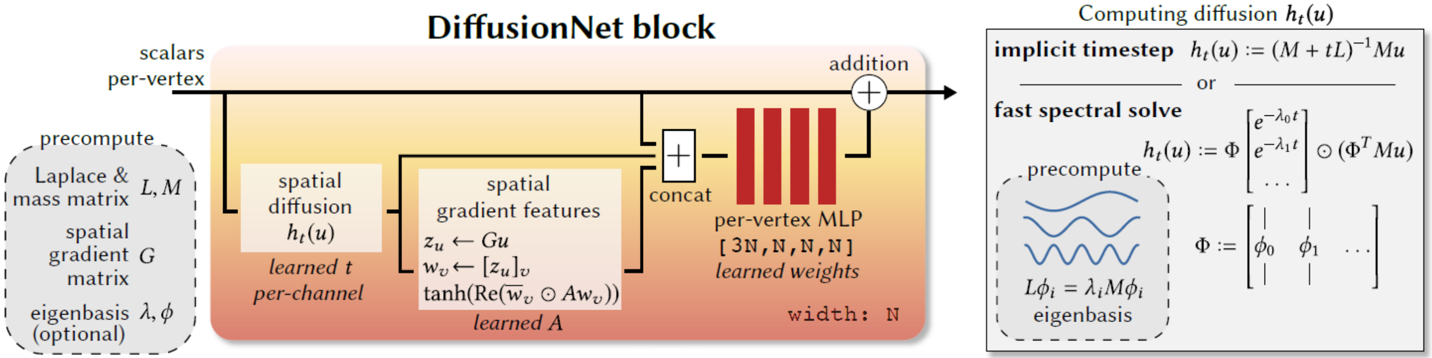
- Introduce gradient operator $G \in \mathbb{C}^{V x V}$
- Calculate spatial gradient $z_u \in \mathbb{C}^V$:

$$z_u = Gu$$

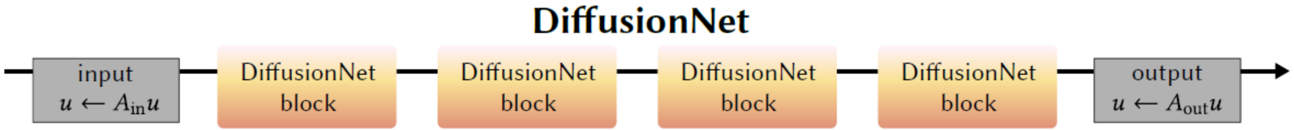
- Stack gradients to form $w_v \in \mathbb{C}^D$
- Learn scalar features $g_v \in \mathbb{R}^D$ via:

$$g_v(i) = \tanh(Re\{\sum_{j=1}^{D} \overline{w_v}(i) A_{ij}\, w_v(j)\})$$

Method: (4) DiffusionNET Architecture



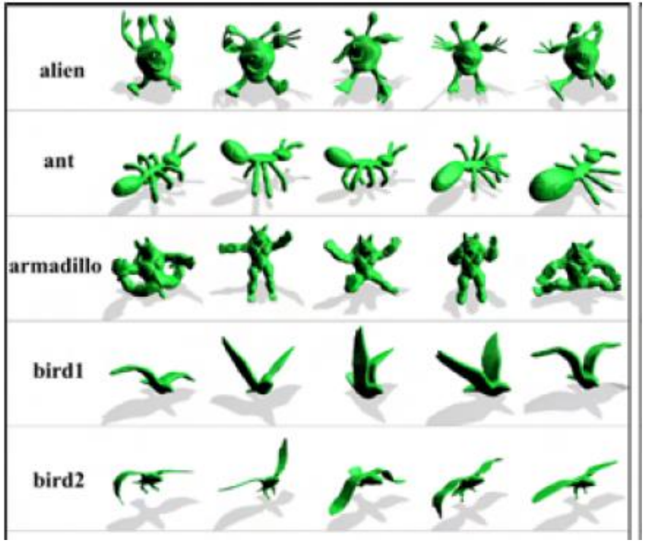
03 Method: (4) DiffusionNET Architecture



- Feature selection:
  - Basic: 3D vertex coordinates (xyz)
  - Rigid & non-rigid invariance: Heat kernel signatures (HKS)
- HKS: network invariant to **orientation-preserving deformations**

04 Experiment Results: (1) Classification

- New state-of-the-art in classification on SHREC-11:

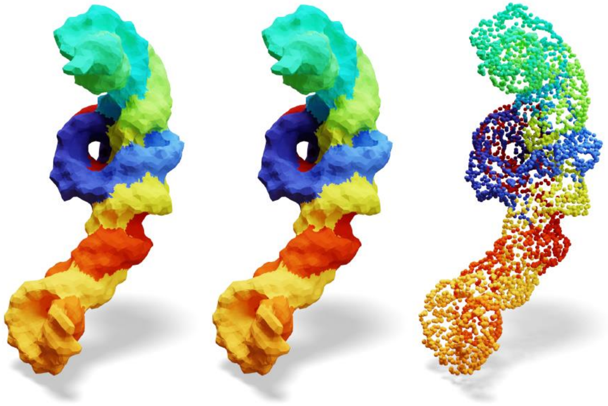


Method	Accuracy
GWCNN [Ezuz et al. 2017]	90.3%
MeshCNN† [Hanocka et al. 2019]	91.0%
HSN† [Wiersma et al. 2020]	96.1%
MeshWalker† [Lahav and Tal 2020]	97.1%
PD-MeshNet† [Milano et al. 2020]	99.1%
HodgeNet† [Smirnov and Solomon 2021]	94.7%
FC† [Mitchel et al. 2021]	99.2%
DiffusionNet - xyz†	99.4%
DiffusionNet - xyz	99.0%
DiffusionNet - hks†	99.5%
DiffusionNet - hks	**99.7%**

Source: Lian, Zhouhui & Godil, Afzal & Bustos, Benjamin & Daoudi, Mohamed & Hermans, Jeroen & Kawamura, Shun & Kurita, Yukinori & Lavoué, Guillaume & Nguyen, Hien & Ohbuchi, Ryutarou & Ohkita, Yuki & Ohishi, Yuya & Porikli, Fatih & Reuter, Martin & Sipiran, Ivan & Smeets, Dirk & Suetens, Paul & Tabia, Hedi & Vandermeulen, Dirk. (2011). SHREC '11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes.. Eurographics Workshop on 3D Object Retrieval. 79-88. 10.2312/3DOR/3DOR11/079-088.

04 Experiment Results: (2) Molecular Segmentation

- New state-of-the-art in segmentation tasks
- RNA mesh consists of 14k vertices
- Training: 38ms, requiring 2.2GB of GPU memory
- Accurate results for **meshes and point clouds**



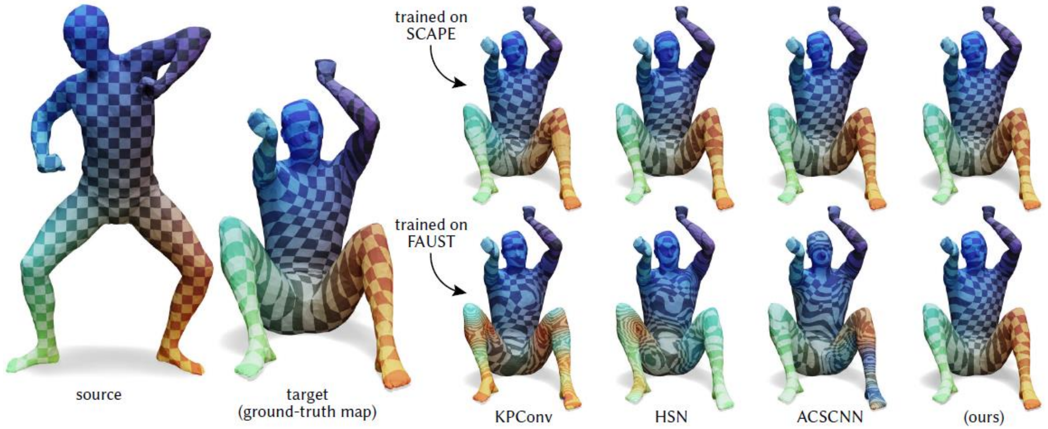
ground truth mesh prediction point cloud prediction

04 Experiment Results: (3) Human Segmentation

- DiffusionNET directly trains on mid-size inputs
- Scales well to large meshes due to spectral acceleration
- Most other approaches inapplicable to large meshes

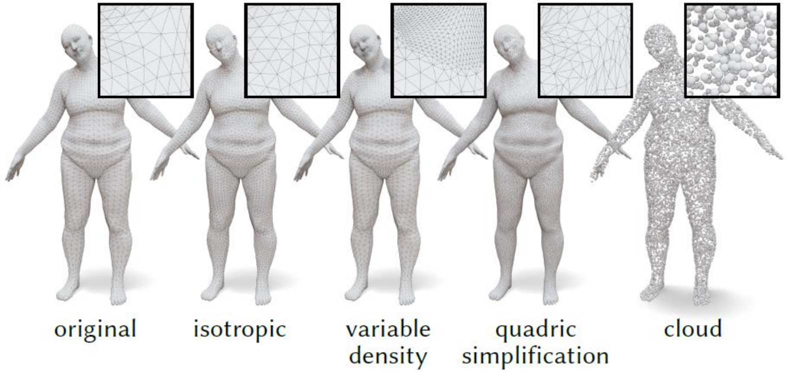
Method		small 752 vert	medium 10k vert	large 184k vert
DiffusionNet (spectral)	*pre:*	288ms	3.55sec	69.5sec
	train:	19ms	25ms	379ms
	infer:	7ms	10ms	154ms
DiffusionNet (direct)	*pre:*	104ms	—	—
	train:	329ms	—	—
	infer:	81ms	—	—
MeshCNN [Hanocka et al. 2019]	*pre:*	85ms	1.13sec	—
	train:	269ms	2.97sec	—
	infer:	194ms	2.71sec	—
HSN [Wiersma et al. 2020]	*pre:*	905ms	162sec	—
	train:	188ms	1.08sec	—
	infer:	68ms	389ms	—
HodgeNet [Smirnov et al. 2021]	*pre:*	*n/a*	*n/a*	—
	train:	752ms	7.61sec	—
	infer:	645ms	6.87sec	—

04 Experiment Results: (3) Functional Correspondence

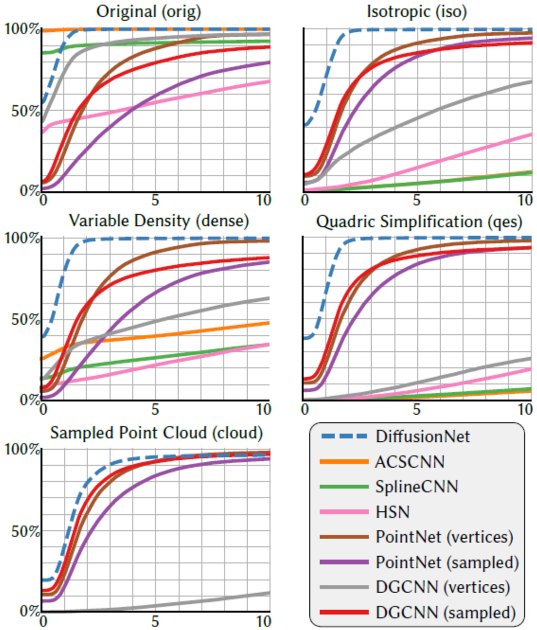


DiffusionNET outperforms other methods in nonrigid correspondence

04 Experiment Results: (4) Discretization Agnostic Learning



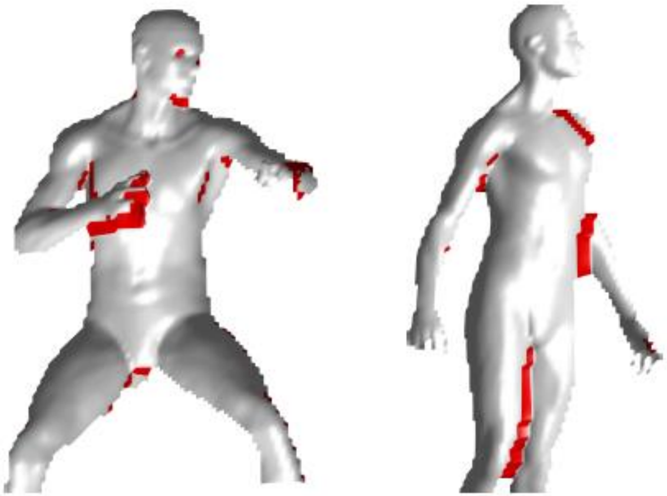
Different discretization techniques



Performance of different networks

04 Limitations

No diffusion for disconnected components



Diffusion at topological merges

Source: Z. L·ahner, E. Rodola, M. M. Bronstein, D. Cremers, O. Burghard, L. Cosmo, A. Dieckmann, R. Klein and Y. Sahillioglu, SHREC'16: Matching of Deformable Shapes with Topological Noise, Proc. of Eurographics Workshop on 3D Object Retrieval (3DOR), May 2016.

Thanks!

Any questions?