An Introduction to Deep Learning on Meshes
SIGGRAPH COURSE 2021
Rana Hanchan & Hsueh-Ti Derek Liu
Real world success of deep learning

- Reverse image search: Alibaba Pailitao
- Facial recognition: Facebook Photo Tags
- Speech recognition / Language processing: Apple Siri, Google translate
Meshes are popular in computer graphics

- fast to render
- adaptive
- efficient to texture
- intuitively deformable
- physics simulation

Sources:
- Sorkine & Alexa 2007
- Sawhney & Crane 2017
- Li et al. 2020
Combining the power of deep learning & meshes

for many applications in geometry processing

Modeling  Editing  Reconstruction  Shape Analysis
Challenges for deep learning on meshes

Representation

Data Accessibility
Irregular
Unordered

\[<x, y, z>\]

\[<1.2, 3.1, -0.7>\]

\[<1.2, 3.4, -0.8>\]

\[<1.5, 3.4, -0.6>\]

\[<1.5, 3.1, -0.7>\]

\[<1.5, 3.7, -0.7>\]

\[<1.7, 3.4, -0.7>\]
Inconsistent
Unoriented
Challenges for deep learning on meshes

Representation

Data Accessibility
Large Warehouses of 3D Mesh Data
High bar for geometric computation
High bar for geometric computation

non-manifold
not watertight

Source: geometry central
High bar for geometric computation

non-manifold  not watertight  intersections

Source: Sacht et. al 2013
High bar for geometric computation

Source: Takayama et. al 2014

- non-manifold
- not watertight
- intersections
- face orientation
Hard to Create 3D Data

Blender

SolidWorks
Curse of dimensionality
Mesh Convolutional Neural Networks

Machine Learning & Geometry Processing

Learning from a Single Mesh
Mesh Convolutional Neural Networks

Machine Learning & Geometry Processing

Learning from a Single Mesh
Voxel (3D Pixel) Representation

convolution
Voxel (3D Pixel) Representation

pooling
Large Memory Cost

O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis

PENG-SHUN WANG, Tsinghua University and Microsoft Research Asia
YANG LII, Microsoft Research Asia
YU-XIAO GUO, University of Electronic Science and Technology of China and Microsoft Research Asia
CHUN-YU SUN, Tsinghua University and Microsoft Research Asia
XIN TONG, Microsoft Research Asia

1 INTRODUCTION

With recent advances in low-cost 3D acquisition devices and 3D modeling tools, the amount of 3D models created by end users has been increasing rapidly. Analyzing and understanding these 3D shapes, such as for classifications, segmentation, and retrieval, becomes more and more important for many computer graphics and vision applications. A key technique for these shape analysis tasks is to extract features of 3D models that can sufficiently characterize their shapes and parts.

In the computer vision field, convolutional neural networks (CNNs) are widely used for image classification and have demonstrated their advantages over manually-crafted solutions in most image analysis and understanding tasks. However, it is a non-trivial task to adopt a CNN designed for 2D images to 3D shapes or vice versa. A set of methods convert the 3D shape to an equivalent point cloud representation and apply a CNN to them. Instead, base methods [Noh and Shin 2015; Wu et al. 2015] extract a 3D shape as an indicator function or distance function sampled over dense voxels and apply a CNN. In this paper, we propose an efficient 3D shape representation method based on Octrees and demonstrate that it is possible to use a 3D CNN on the new shape representation.
Not a smooth representation
Estimating surface quantities on voxels

Caissard et al. 2019

Lachaud et al. 2020
Not store \{0,1\}
Not store \{0,1\}

signed distance function (SDF)

\[
\begin{align*}
R &> 0 \\
R &= 0 \\
R &< 0
\end{align*}
\]
SDF to sub-pixel resolution

[Dai et al. 2017]
Get rid of the voxel grid

0 or 1

a real value

$f(x,y,z) = a \text{ real value}$
Neural Signed Distance Function

\[ f_\theta(p_x, p_y, p_z) = \text{signed distance} \]
An Alternative Shape Representation

missing basic ingredients:
- differential operators
- differential quantities
- shape editing
- ...

-
Use image convolution on surfaces

• Convolution
• Pooling
• Leverage image data

---

Su et al. 2015
Leverage Image Data
Extending to local tasks is hard
Global Parameterization
Seamless Parameterization

Aigerman & Lipman 2015
Seamless Parameterization

Aigerman & Lipman 2015
Challenges

- Not unique
- Orientation
- Cannot avoid distortion
Local Parameterization
Exponential Maps

Schmidt et al. 2006
Which direction to use?

Consider all orientations  
[Masci et al. 2015]

Pick one direction at a time  
[Poulenard & Ovsjanikov 2018]

Rotation-Equivariant  
[Wiersma et al. 2020]
**Spectral Convolution**

Convolution in the spatial domain is the **point-wise product** in the spectral domain.
Spectral Convolution (e.g., [Defferrard et al. 2016])

\[ y = \mathcal{T}^{-1} \left( w \odot \mathcal{T}(x) \right) \]

- filter weights
- spatial signal
- inverse FT
- FT
Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard  Xavier Bresson  Pierre Vandergheynst
EPFL, Lausanne, Switzerland
{michael.defferrard,xavier.bresson,pierre.vandergheynst}@epfl.ch

Abstract

In this work, we are interested in generalizing convolutional neural networks (CNNs) from low-dimensional regular grids, where image, video and speech are represented, to high-dimensional irregular domains, such as social networks, brain connectomes or words’ embedding, represented by graphs. We present a formulation of CNNs in the context of spectral graph theory, which provides the necessary mathematical background and efficient numerical schemes to design fast localized convolutional filters on graphs. Importantly, the proposed technique offers the same linear computational complexity and constant learning complexity as classical CNNs, while being universal to any graph structure. Experiments on MNIST and 20NEWS demonstrate the ability of this novel deep learning system to learn local, stationary, and compositional features on graphs.
Do not generalize to other shapes
(even the same shape with a different connectivity)
Do not generalize to other shapes
(even the same shape with a different connectivity)

different Fourier bases
Synchronizing Spectral Spaces

SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation

Li Yi¹ Hao Su¹ Xingwen Guo² Leonidas Guibas¹
¹Stanford University ²The University of Hong Kong

Abstract

In this paper, we study the problem of semantic annotation on 3D models that are represented as shape graphs. A functional view is taken to represent localized information on graphs, so that annotations such as part segment or keypoint are nothing but 0-1 indicator vertex functions. Compared with images that are 2D grids, shape graphs are irregular and nonisomorphic data structures. To enable the prediction of vertex functions on them by convolutional neural networks, we resort to spectral CNN method that enables weight sharing by parameterizing kernels in the spectral domain spanned by graph laplacian eigenbases. Under this setting, our network, named SyncSpecCNN, strive to overcome two key challenges: how to share coefficients and conduct multi-scale analysis in different parts of the graph for a single shape, and how to share information across related but different shapes that may be represented by very different graphs. Towards these goals, we introduce a spectral parameterization of dilated convolutional kernels and a spectral transformer network. Experimentally we tested our SyncSpecCNN on various tasks, including 3D shape part segmentation and 3D keypoint prediction. State-of-the-art performance has been achieved on all benchmark datasets.

Figure 1. Our SyncSpecCNN takes a shape graph equipped with vertex functions (i.e., spatial coordinate function) as input and predicts a per-vertex label. The framework is general and not limited to a specific type of output. We show 3D part segmentation and 3D keypoint prediction as example outputs here.

It is not straightforward to apply traditional deep learning approaches to 3D models because a mesh representation can be combinatorially irregular and does not permit the optimizations exploited by convolutional approaches, such as weight sharing, which depend on regular grid structures. In this paper we take a functional approach to represent information about shapes, starting with the observation that a shape part is itself nothing but a 0-1 indicator function defined on the shape.

Our basic problem is to learn functions on shapes. We start with example functions provided on a given shape and learn a classification or regression function from it.
Different representations
Why not point cloud convolution?
Why not graph convolution?
Surface Meshes
Neural Networks on Meshes

Example Outputs

- Global shape descriptor
- Probability to collapse an edge
- Displacement per vertex
- Segmentation label per-face
Y = ReLU(2L + 2W)
Fully-connected
The set of assumptions that we encode into our network, which make it better suited for the task
Inductive Bias

- Good
- Robust to irrelevant variations of the input
Local tasks

Global tasks

≈

≠
Local tasks

Global tasks
Local task

predict values per mesh element
Local task

predict values per mesh element

Faces

\[ f_0, f_1, f_2, f_3, f_4, \ldots, f_N \]

seat
legs
back
seat
legs
\ldots
seat
Local task

predict values per mesh element

Faces

\[ f_0, f_1, f_2, f_3, f_4, \ldots, f_N \]

Fully connected network not suitable for this task
Inspiration: image segmentation

Shared weights are a good inductive bias
Convolution

Shared-weights

Convolutional Filter

Output

Image

\( \theta_0 \quad \theta_1 \quad \theta_2 \)

\( \theta_3 \quad \theta_4 \quad \theta_5 \)

\( \theta_6 \quad \theta_7 \quad \theta_8 \)
Convolutions on meshes
Learn over intrinsic patches

local parameterizations
Intrinsic Techniques

MDGCNN. Poulenard & Ovsjanikov [SIGGRAPH Asia 2018]

Surface Networks via General Covers. Haim et. al [ICCV 2019]
Toric Covers. Maron et. al [SIGGRAPH 2017]

HodgeNet. Smirnov & Solomon [SIGGRAPH 2021]

CNNs on Surfaces. Wiersma & Eisemann [SIGGRAPH 2020]
Convolutions on meshes
Mesh edges have 4 edge-neighbors

**Vertices**

<\(x, y, z\)>

**Edges**

<\(v_i, v_j\)>

**Faces**

<\(v_i, v_j, v_k\)>

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Learn Filters on Edge Features

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Learn Filters on Edge Features

Different ordering of edge indices

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Mesh Convolution Order

Face normal
• Consistent ordering in each face
• Two valid orderings

Build symmetric features
\(\text{e} \rightarrow (\text{a+c, l}a\text{-cl, b+d, l}b\text{-dl})\)
Input edge features

Relative geometric features

Invariant to rigid transformations
Recap: learning local descriptors

Input mesh → Convolution → Convolution → Convolution → ... → Per edge attribute
Incorporating more context
Inspiration: image pooling

Input (4x4)

\[
\begin{array}{cccc}
4 & 6 & 1 & 1 \\
1 & 3 & 1 & 3 \\
4 & 0 & 0 & 8 \\
8 & 5 & 4 & 0 \\
\end{array}
\]

Output (2x2)
Mesh pooling via edge collapse

Classic Edge Collapse

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Mesh pooling via edge collapse
Mesh pooling via edge collapse

$p = \text{avg}(a, b, e)$

$q = \text{avg}(c, d, e)$
Mesh unpooling

\[ p = \text{avg}(a, b, e) \]

\[ q = \text{avg}(c, d, e) \]

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Different tasks

Different simplifications
MeshCNN
MeshCNN Segmentation Overview

Mesh Conv → Mesh Pooling → Mesh Conv → Mesh Pooling → Mesh Conv → Mesh Unpooling → Mesh Conv → Mesh Unpooling
MeshCNN Classification Overview

Mesh Conv → Mesh Pooling → Mesh Conv → Mesh Pooling → Mesh Conv → Mesh Unpooling → Fully Connected → Fully Connected

Human
Shape Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Split 16</th>
<th>Split 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshCNN</td>
<td>98.6%</td>
<td>91.0%</td>
</tr>
<tr>
<td>GWCNN</td>
<td>96.6%</td>
<td>90.3%</td>
</tr>
<tr>
<td>GI</td>
<td>96.6%</td>
<td>88.6%</td>
</tr>
<tr>
<td>SN</td>
<td>48.4%</td>
<td>52.7%</td>
</tr>
<tr>
<td>SG</td>
<td>70.8%</td>
<td>62.6%</td>
</tr>
</tbody>
</table>
Cube Engraving Classification

Which engraving does this have?

64.26% Point cloud

92.16% Mesh
Intermediate Mesh Pooling

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Human Segmentation

<table>
<thead>
<tr>
<th>Method</th>
<th># Features</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshCNN</td>
<td>5</td>
<td><strong>92.30%</strong></td>
</tr>
<tr>
<td>SNGC</td>
<td>3</td>
<td>91.02%</td>
</tr>
<tr>
<td>Toric Cover</td>
<td>26</td>
<td>88.00%</td>
</tr>
<tr>
<td>PointNet++</td>
<td>3</td>
<td>90.77%</td>
</tr>
<tr>
<td>DynGraphCNN</td>
<td>3</td>
<td>89.72%</td>
</tr>
<tr>
<td>GCNN</td>
<td>64</td>
<td>86.40%</td>
</tr>
<tr>
<td>MDGCNN</td>
<td>64</td>
<td>89.47%</td>
</tr>
</tbody>
</table>

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Convolutions on half-edges

Neural Subdivision. Liu et al [SIGGRAPH 2020]
Half-edge input features

Edge vectors

Differential coordinates

Neural Subdivision. Liu et al [SIGGRAPH 2020]
Input mesh coarsening

Neural Subdivision. Liu et al [SIGGRAPH 2020]
Learning on random walks

MeshWalker. Lerner & Tal [SIGGRAPH Asia 2020]
Learning on random walks

Walk step: \( \frac{V}{50} \)

Walk step: \( \frac{V}{7} \)

Walk step: \( \frac{V}{2.5} \)

MeshWalker. Lerner & Tal [SIGGRAPH Asia 2020]
Convolutions on mesh faces

Input features

Face-based convolution

Deep Geometric Texture Synthesis [SIGGRAPH 2020]
Input mesh untexturing
Convolutions on primal/dual mesh graphs

Mesh

Primal Graph

Dual Graph

Primal-Dual Mesh Convolutional Neural Network. Milano et. al [NeurIPS 2020]
Mesh Pooling via Edge Contraction

Edge contraction in $\mathcal{P}(\mathcal{M})$

Face merge in $\mathcal{M}$

Primal-Dual Mesh Convolutional Neural Network. Milano et. al [NeurIPS 2020]
Enhanced mesh pooling

MeshCNN Fundamentals. Barda et al [ArXiv 2021]
Enhanced mesh pooling

Original mesh pooling

Enhanced mesh pooling

MeshCNN Fundamentals. Barda et. al [ArXiv 2021]
Convolution and pooling from subdivision

Subdivision-based Mesh Convolutional Neural Networks. Hu et. al [ArXiv 2021]
Invariance to rigid transformations

still
It’s a bunny
Invariance to rigid transformations

... sometimes this can be too restrictive

Umetani & Schmidt [SIGGRAPH Asia 2013]
Expressiveness vs. Generalization
Break shift-invariance

Gain expressive power
Helps better fit the data

Fast Fourier Features Tancik et al. [2020]
Triangulation robustness

Perform simple augmentations
Triangulation robustness in segmentation

MeshCNN: a Network with an Edge [SIGGRAPH 2019]
Triangulation robustness in deformation

Without augmentation

With augmentation

Neural Blend Shapes [SIGGRAPH 2021]
Triangulation robustness via diffusion
Mesh Convolutional Neural Networks

Machine Learning & Geometry Processing

Learning from a Single Mesh
A Fundamental Tool: MeshCNN
A Small Component

meshCNN → build dV → + → sample → dist → X

C

loss
A Small Component

meshCNN → build dV → sample → dist → X

C → meshCNN

loss
Image Processing

SIFT
[Lowe 1999]
Image Processing

SIFT [Lowe 1999]

learned features
E.g., Optimal Flow via Pyramid

Sun et al. 2018
A tool in our toolbox

- Fourier analysis
- Differential geometry
- Optimization
- Linear solvers
- Machine learning
A tool in our toolbox

Fourier analysis
optimization
differential geometry
linear solvers
machine learning
No-free-lunch

push the limits

new possibilities
No-free-lunch

push the limits
new possibilities

quality
expensive training
generalization
large training data
Overview

- Fourier analysis
- Optimization
- Differential geometry

Machine Learning

Classic
Alleviate Limitations

push the limits
new possibilities

quality
expensive training
generalization
large training data
Roles of machine learning
Non-linear Function Approximator

\[ y = f_{\theta}(x) \]
Non-linear Function Approximator

$y$

$x$

$f_\theta$
Need a lot of training data
May suffer from overfitting
Difficult to Generalize
Machine learning as a feature extractor
Feature extractor

\[ y = f_\theta(x) \]
Shape Classification

human

not human
Global Shape Descriptors

entire shape $\rightarrow [a_1, a_2, \cdots, a_n]$
Measure Shape Difference

\[ \| [a_1, a_2, \cdots, a_n] - [b_1, b_2, \cdots, b_n] \| = \]
Measure Shape Difference

\[\| [c_1, c_2, \cdots, c_n] - [b_1, b_2, \cdots, b_n] \| = \]
E.g., Spherical Harmonics

\[ \psi \approx \sum_{i=1}^{k} a_i \phi_i \]

Kazhdan et al. 2003
E.g., Bags of Words

in math science, **matrix decomposition** is a factorization of a matrix into some **canonical form**. Each type of decomposition is used in a particular problem.

in particular matrix used type a some science decomposition form a factorization of is canonical matrix math decomposition is in a each problem into of

in biological science, decomposition is the process of organisms to break down into simpler form of matter. Usually, decomposition occurs after death.

matrix is a **science fiction** movie released in 1999. Matrix refers to a simulated reality created by machines in order to subdue the human population.

---

Bronstein et al. 2011
E.g., Bags of Features

Bronstein et al. 2011
Learned Global Descriptors

conv

#E

#filters
Learned Global Descriptors

- conv
- pool
- conv
- pool
- global pool
Learned Global Descriptors

conv pool conv pool global pool
Learned Global Descriptors
Inputs to the network
Not Orientation Invariant

Frog
Not Orientation Invariant
One Solution: Data Augmentation

Frog
It helps, but ...

longer to train

Adversarial attack
Leverage mesh structure

not invariant quantities

rotation invariant quantities
Orientation Invariant
Orientation Invariant
Push the limits

<table>
<thead>
<tr>
<th>Method</th>
<th>Split 16</th>
<th>Split 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshCNN</td>
<td>98.6%</td>
<td>91.0%</td>
</tr>
<tr>
<td>GWCNN</td>
<td>96.6%</td>
<td>90.3%</td>
</tr>
<tr>
<td>GI</td>
<td>96.6%</td>
<td>88.6%</td>
</tr>
<tr>
<td>SN</td>
<td>48.4%</td>
<td>52.7%</td>
</tr>
<tr>
<td>SG</td>
<td>70.8%</td>
<td>62.6%</td>
</tr>
</tbody>
</table>

Classic method

[Bronstein et al. 2011]
Push the limits

<table>
<thead>
<tr>
<th>Method</th>
<th>Split 16</th>
<th>Split 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshCNN</td>
<td>98.6</td>
<td>91.0%</td>
</tr>
<tr>
<td>GWCNN</td>
<td>96.6%</td>
<td>90.3%</td>
</tr>
<tr>
<td>GI</td>
<td>96.6%</td>
<td>88.6%</td>
</tr>
<tr>
<td>SN</td>
<td>48.4%</td>
<td>52.7%</td>
</tr>
<tr>
<td>SG</td>
<td>70.8%</td>
<td>62.6%</td>
</tr>
</tbody>
</table>

Classic method
[Bronstein et al. 2011]
Shape Segmentation
Local Shape Descriptors

\[ [a_1, a_2, \cdots, a_n] \]

\[ [b_1, b_2, \cdots, b_n] \]

a fixed dimensional vector per element
E.g., Heat Kernel Signature

Sun et al. 2009
Learned Local Descriptors

conv
pool
conv
pool

global pool
Learned Local Descriptors
Learned Local Descriptors
Machine Learning Segmentation
Machine Learning Segmentation
Machine Learning Segmentation

unpooling
Handle Shape Variants

image source: Brandt et al. 2016
Handle Shape Variants

isometry invariant quantities
Handle Shape Variants

isometry invariant quantities
discretizations
Handle Shape Variants

isometry invariant quantities

discretization agnostic convolution
(e.g., Sharp et al. 2021)
ML as Feature Extractors
Applications of learned feature extractors — combined with classic methods —
Example: Shape Matching
Texture Transfer

Schmidt et al. 2019
Point Maps

a point on $\mathcal{M}$ $\rightarrow$ a point on $\mathcal{N}$
Functional Maps

Ovsjanikov et al. 2012
Functional Maps Overview
Functional Maps Overview

\[ \psi^M = \sum_{i=1}^{k} a_i \phi_i^M \]

\[ \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} \]

\[ C \]

functional map
Functional Maps Overview

\[ \psi^M = \sum_{i=1}^{k} a_i \phi_i^M \]

\[
\begin{bmatrix}
  a_1 \\
  \vdots \\
  a_k
\end{bmatrix} = 
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_k
\end{bmatrix} \quad \text{functional map}
\]

\[
\begin{bmatrix}
  a_1 \\
  \vdots \\
  a_k
\end{bmatrix} = C
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_k
\end{bmatrix}
\]
Functional Maps Overview

\[ \psi^M = \sum_{i=1}^{k} a_i \phi^M_i \]

\[ \psi^N = \sum_{i=1}^{k} b_i \phi^N_i \]

\[ \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} \]

functional map
Computing Functional Maps

\[ \psi^M = \sum_{i=1}^{k} a_i \phi_i^M \]

\[ \psi^N = \sum_{i=1}^{k} b_i \phi_i^N \]

\[ \begin{bmatrix}
    a_1 \\
    \vdots \\
    a_k
\end{bmatrix} = \arg \min_C E \]

\[ \begin{bmatrix}
    b_1 \\
    \vdots \\
    b_k
\end{bmatrix} \]
Computing Functional Maps

\[ \psi^M = \sum_{i=1}^{k} a_i \phi^M_i \]

\[ \psi^N = \sum_{i=1}^{k} b_i \phi^N_i \]

\[ \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} \]

\[ \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} \]

\[ \mathbf{C} = \text{arg min}_c \mathcal{J} \]
Deep Functional Maps

\[
\hat{\psi}^M = \sum_{i=1}^{k} a_i \phi_i^M \\
\hat{\psi}^N = \sum_{i=1}^{k} b_i \phi_i^N
\]
Deep Functional Maps

\[ \psi^M \rightarrow \sum_{i=1}^{k} a_i \phi_i^M \]

\[ \hat{\psi}^M \rightarrow \sum_{i=1}^{k} a_i \phi_i^M \]

\[ C = \arg \min_{C} E \]

\[ \psi^N \rightarrow \sum_{i=1}^{k} b_i \phi_i^N \]

\[ \hat{\psi}^N \rightarrow \sum_{i=1}^{k} b_i \phi_i^N \]

Litany et al. 2017
Deep Functional Maps

\[ \psi^M \rightarrow \sum_{i=1}^{k} a_i \phi_i^M \rightarrow C = \arg \min_{C} E \rightarrow \text{convert to soft maps} \rightarrow J \]

\[ \psi^N \rightarrow \sum_{i=1}^{k} b_i \phi_i^N \]

Litany et al. 2017
Push the limits

**Accuracy vs. Geodesic Error**

- **Deep Method**
- **Classic Method**
Requires Supervision

obtaining ground truth is expensive

Litany et al. 2017
arg min \_C \_E(C)
\_E(C) = \alpha_1 E_1(C) + \alpha_2 E_2(C) + \alpha_3 E_3(C) + \alpha_4 E_4(C)
Desired Properties

• Bijectivity

Ovsjanikov et al. 2016, 2017
Desired Properties

• Bijectivity
• Area preservation
• Laplacian commutativity
• Descriptor preservation
• …

Ovsjanikov et al. 2016, 2017
From Supervised to Unsupervised

\[ \psi^M \rightarrow \sum_{i=1}^{k} a_i \phi_i^M \]

\[ C = \arg \min_C E \rightarrow \text{convert to soft maps} \rightarrow J \]

\[ \psi^N \rightarrow \sum_{i=1}^{k} b_i \phi_i^N \]
From Supervised to Unsupervised

\[
\psi^M \rightarrow \sum_{i=1}^{k} a_i \phi_i^M \\
\hat{\psi}^M \rightarrow \text{desired properties}
\]

\[
\psi^N \rightarrow \sum_{i=1}^{k} b_i \phi_i^N \\
\hat{\psi}^N \rightarrow \text{unsupervised loss function}
\]

\[
C = \arg\min_C E \rightarrow E(C)
\]

Roufosse et al. 2019
Comparable Results

![Graph showing accuracy vs. geodesic error for FAUST and Remeshed FAUST datasets.]

- Supervised and unsupervised methods are compared.
- Accuracies are higher for supervised methods in both datasets.

Roufosse et al. 2019
Key Takeaways

- good results, but not perfect
- push the limit
- require supervision
- combine with classic method
- weak/no supervision
- push the limit
- open questions
Example: Shape Deformation

(image source: Jacobson & Sorkine 2011)
Learning Shape Deformation
Learning Shape Deformation
Learning Shape Deformation

Groueix et al. 2018

Yumer & Mitra 2016

Gao et al. 2019

Rakotosaona & Ovsjanikov, 2020
Issues of preserving details

image source: Rakotsaona & Ovsjanikov, 2020
Linear Blend Skinning

\[ \mathbf{v}_i' = \sum_{j=1}^{m} w_{i,j} T_j \mathbf{v}_i \]

- new vertex locations
- input vertex locations
- weight
- transformation of e.g., handles
Linear Blend Skinning

- input
- skeleton
- cage
- point handles

image source: Jacobson et al. 2011
Linear Blend Skinning

input

deformed

image source: Jacobson et al. 2011
Linear Blend Skinning

Image source: Jacobson et al. 2011
Cage-Based Deformation

construct a cage
compute cage weights
deform the model

Neural Cages

Sources → Targets

Yifan et al. 2020
Neural Cages

Yifan et al. 2020
\[ J = w_1 L_{MVC} + w_2 L_{\text{align}} + w_3 L_{\text{shape}} \]

- “nice” cage (e.g., not self-overlap)
- positive cage weights
- align with target
- Chamfer
- preserve input shapes
- e.g., preserve normals
Keypoint Deformer

Jakab et al. 2021
Keypoint Deformer

source shape \( x \) → influence predictor \( \Gamma \) → influence matrix \( W \) → source cage \( c \) → cage-based deformation \( \beta \) → deformed source shape \( x^* \)

target shape \( x' \) → keypoint predictor \( \Phi \) → unsupervised keypoints \( p, p' \) → cage skinning \( \eta \) → deformed cage \( c^* \) → target shape \( x' \)

Jakab et al. 2021
Key Takeaways

Classic
- not global shape aware
- not preserve details
- manual efforts

Deep
- global shape aware
- preserve details
- less manual efforts
- lower quality

Classic + Deep
- global shape aware
- preserve details
- less manual efforts
- lower quality
Key Takeaways

Classic
- not global shape aware
- less manual efforts
- lower quality

Deep
- global shape aware
- preserve details
- lower quality

Classic + Deep
- global shape aware
- preserve details
- less manual efforts
- lower quality
Machine learning as geometric prior
Surface Reconstruction
An ill-posed problem
A Smooth Solution
Classic Smoothness Prior

Kazhdan et al. 2006
Smoothness is not always good

image source: Hanocka et al. 2020
Inductive Bias as Prior

linear
Deep Network Biases

network 1

network 2

network 3
New Possibilities : Self-Prior

Hanocka et al. 2020
Results of Self-Prior

input

Smoothness
[Khazhdan et al. 2006]

Self-prior
[Hanocka et al. 2020]
Point2Mesh Overview

Hanocka et al. 2020
Point2Mesh Overview

U-Net
[Ronneberger et al. 2015]

conv. pool → conv. unpool

per edge

avg

per vertex

Hanocka et al. 2020
Point2Mesh Overview

U-Net
[Ronneberger et al. 2015]

per edge
avg

per vertex

Hanocka et al. 2020
CNN filters are shared
Point2Mesh Overview

Hanocka et al. 2020
Loss Function: Chamfer Distance

Barrow et. al. 1977, Fan et. al. 2017
Loss Function: Chamfer Distance

\[ J(X, Y) = \sum_{x} (x - X_{\text{min}})^2 + (x - Y_{\text{min}})^2 \]

Barrow et. al. 1977, Fan et. al. 2017
Loss Function: Chamfer Distance

\[ \mathcal{J}(X, Y) = \|x - y\|^2 \]

Barrow et. al. 1977, Fan et. al. 2017
Loss Function: Chamfer Distance

\[ \mathcal{J}(X, Y) = \min_{y \in Y} \|x - y\|^2 \]

Barrow et. al. 1977, Fan et. al. 2017
Loss Function: Chamfer Distance

\[ \mathcal{J}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|^2 \]

Barrow et. al. 1977, Fan et. al. 2017
Loss Function: Chamfer Distance

\[ J(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|^2 + \sum_{y \in Y} \min_{x \in X} \|y - x\|^2 \]

bidirectional Chamfer

Barrow et. al. 1977, Fan et. al. 2017
Optimization

Hanocka et al. 2020
Results

Hanocka et al. 2020
Denoising

smoothness prior  deep geometric prior  self-prior

Hanocka et al. 2020
Early-stop as a regularization

train for a while

train until converge
Learning from external data
Learning from internal data
Example: Mesh Upsampling
Image Upsampling
Image Upsampling

- assume image size 64
- assume grid structure
- assume top-left to bottom-right
- assume upright position

- assume image size 128
- assume grid structure
- assume top-left to bottom-right
- assume upright position
Draw inspiration from images

- assume image size 64
- assume grid structure
- assume top-left to bottom-right
- assume upright position

- assume image size 128
- assume grid structure
- assume top-left to bottom-right
- assume upright position
Draw inspiration from images

- assume 269 vertices
- assume a specific connectivity
- assume a specific ordering
- assume a specific orientation

- assume 1070 vertices
- assume a specific connectivity
- assume a specific ordering
- assume a specific orientation
Classic Subdivision

\[ \sum w_i v_i \]
Where we should put the network?
Discrete Variables

image source: Zorin 2005
Where we should put the network?
From features to locations

\[ y = f_\theta(x) \]
From features to locations

difficult to define

\[ y = f_\theta(x) \]
Still an ongoing research

Catmull-Clark 1978

Loop 1987

Vaxman et al. 2018

Preiner et al. 2019
Where we should put the network?
Where we should put the network?
Neural Subdivision Training

input \rightarrow J(\cdot, \cdot) \rightarrow output, ground truth
Neural Subdivision Training

input

$J()$

output

ground truth
Training Data
Neural Subdivision Training

input

output

ground truth
Chamfer Distance

Barrow et. al. 1977, Fan et. al. 2017
Chamfer Distance

Barrow et. al. 1977, Fan et. al. 2017
Leverage the Structure

output  subdivision  input  decimation  ground truth
Bijective Mapping

\[ \|v - f(v)\|^2 \]
Neural Subdivision Training

input

output

ground truth
Subdivision Network

half-flap

displacement
Subdivision Network
Key Takeaways

- assume 1070 vertices
- assume a specific connectivity
- assume a specific ordering
- assume a specific orientation
Alleviate limitations

- quality
- expensive training
- generalization
- large training data
- push the limits
- new possibilities
Alleviate limitations

quality
expensive training
generalization
large training data
push the limits
new possibilities

single shape training
Alleviate limitations

- quality
- expensive training
- generalization
- large training data
- push the limits
- single shape training
- new possibilities
Alleviate limitations

- quality
- expensive training
- generalization
- large training data

push the limits
new possibilities

single shape training
Alleviate limitations

- quality
- expensive training
- generalization
- large training data
- push the limits
- new possibilities
- single shape training
Alleviate limitations

- quality
- expensive training
- generalization
- large training data

push the limits
new possibilities

generalization
large training data

single shape training
Enjoy Advantages

- quality
- expensive training
- generalization
- large training data
- push the limits
- new possibilities

Dyn et al. 1990
Loop 1987
learned subdivision
Enjoy Advantages

- quality
- expensive training
- generalization
- large training data
- push the limits
- new possibilities
Ongoing Research

black-box

convergence
Deep Geometric Texture Synthesis
Geometric Textures
Objective: Generative Texture Model
External Dataset

Difficult to obtain collections with same geometric textures

Inconsistencies make learning difficult
Learn from a Single Shape

Patches are training data

Textures contain self-repetitions
Learn from a Single Shape

Patches are training data

Textures contain self-repetitions

Local patches are similar for different shapes!

Exemplar

New Object

Synthesized Textures
Train on local patches

Untextured

Generative Network

Exemplar
Generalization from local patches

New Object → Generative Network → Synthesized
Progressive Texture Synthesis
Multi-Scale Textures
Texture Interpolation
Summary
Connectivity based Mesh Learning

- Sharp features
- Fix non-manifold
- Hole filling
Mesh Generative Models

- structural learning
- structural shape synthesis
- mesh boolean
Shape Perception

input

style

output
Futuristic 3D Modeling Tools

Park et al. 2019
Draw Inspiration from Classic Methods