Fully Convolutional Mesh Autoencoder using Spatially Varying Kernels

Presenter: Yi Zhou

09/2020
Zhou Yi, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh.

"Fully Convolutional Mesh Autoencoder using Efficient Spatially Varying Kernels."
Accepted by Neurips 2020
How to apply CNN on registered 3D meshes?

Registered Mesh: Mesh with the same number and order of vertices and edges.

http://dfaust.is.tue.mpg.de/
Common Practice: 2D CNN in UV space

Problems:
1. Artifacts along seam lines and from distortion
2. Poor performance when reconstructing global deformation
CNN on 3D Meshes
Difficulties

A Mesh is usually a non-uniform discretization.

Cannot sample uniform kernels on a non-uniform mesh

Shift-invariance Grid

Shift-variance Mesh
Existing Methods

Spectral Method:
(Chebyshev ...)
Lose Fidelity, unstable

Feature-Conditioned Method:
(GAT, MoNet, FeaStNet ...)
Sensitive to big variations.

Special Method:
(Spiral CNN [Neural3DMM 2020])
Ad-hoc, limited to 2-D manifold

(EdgeCNN [MeshCNN 2019])
Slow, limited to 2-D manifold, Lose geometry information.
Mesh CNN with Up and Down-Scaling
Existing Methods

Quadric Mesh Simplification:
[CoMA 2018, Neural3DMM 2019]
Fixed parameters
Overfit to one template mesh

Dynamic Edge Collapsing:
[MeshCNN 2019]
Very slow
limited to 2-D manifold
minimal downscaling size requirements.
Our Method
A continuous kernel can be shared in a continuous space.

A discretized kernel can be sampled from a continuous kernel.

The sampling function needs to be defined per vertex locally.
Insights

Discrete Conv Filter at A Local Patch

- : Weight Basis on an imaginary grid
- : final weights
Our Convolution Operation

Each Convolution Layer has one kernel basis

\[ B = \{B_k\}_{k=1}^{M}, \ B_k \in \mathbb{R}^{I \times O} \]

Each edge \( j \) for a local vertex \( i \) has coefficients

\[ A_{i,j} = \{\alpha_{i,j,k}\}_{k=1}^{M}, \ \alpha \in \mathbb{R} \]

The weight \( W_{i,j} \) on each edge is computed as

\[ W_{i,j} = \sum_{k=1}^{M} \alpha_{i,j,k} B_k \]

The output feature is computed as

\[ y_i = \sum_{x_{i,j} \in \mathcal{N}(i)} W_{i,j} x_{i,j} + b \]

\( B \) and \( A_{i,j} \) are training parameters, shared across the dataset.
Our Pooling Operation

Observation: local density is non-uniform

Solution: Monte Carlo Integration with learned density coefficients

Formulation:

Each local vertex $j$ has a density coefficient

$$
\rho_{i,j}' = \frac{|\rho_{i,j}|}{\sum_{j=1}^{E_i} |\rho_{i,j}|}
$$

The output feature is computed as

$$
y_i = \sum_{j \in \mathcal{N}(i)} \rho_{i,j}' x_{i,j}
$$

$\rho_{i,j}$ are training parameters, shared across the dataset.
Down and Up Scaling Based Only on Graph Connectivity

Down-sampling

Select sampled vertices with stride=1.

Create input/output graphs for Conv/Pool.

Create edges between input and output graphs.

Reverse

Up-sampling

Create edges between input and output graphs.
## Operations Analog to Regular CNN

<table>
<thead>
<tr>
<th>Down-sampling</th>
<th>Up-sampling</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcConv</td>
<td>vcTransposeConv</td>
<td>(Stride, kernel radius, basis size, in_channel, out_channel, dilation)</td>
</tr>
<tr>
<td>vdPool</td>
<td>vdUnpool</td>
<td>(Stride)</td>
</tr>
<tr>
<td>vdDownResidual</td>
<td>vdUpResidual</td>
<td>(In_channel, out_channel)</td>
</tr>
</tbody>
</table>

### Diagrams

**Original and output graph**

- Input graph: $x_i$
- Output graph: $y_i$

**Output graph**

- Original and input graph: $x_i$
- $y_i$, $...$
Residual Block

Input → vcConv/vcTransConv → Elu → Output

vdDownRes/vdUpRes

Linear Transform → vdPool/vdUnpool
Auto-Encoder for DFAUST Dataset

Stride=2, Kernel radius =2
Localized Latent Features

Middle layer graph
Ours

Receptive field

Middle layer graph
Quadric Mesh Simplification

Receptive field
Results
Ours: Lowest Reconstruction Error
Ours: Lowest Reconstruction Error

Compression Rate: 0.3%

Groundtruth  Ours  Neural3DMM

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Localized Latent Space Interpolation

Man A + Man B = New Man A

New Man A’s left leg = Man B’s left leg
Reconstruct both Geometry and Color

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Localized Latent Space Interpolation

- Latent vertices

Source → Target
Localized Latent Space Interpolation

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Efficient for High Resolution Meshes

153,000 Vertices, 24k training meshes, 2k test meshes, compression rate 0.75%.

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Volumetric Mesh (Tetrahedrons)

960 Vertices, 7k training meshes, 562 test meshes, compression rate 1.1%, test error 0.2 mm.
Volumetric Mesh (Tetrahedrons)

Compression Rate: 1.1%

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Non-Manifold Mesh

20,000 Vertices, 10k training meshes, 2k test meshes, compression rate 2%, test error 4.1 cm.

Compression Rate: 2%

Groundtruth

Output

Videos can be watched at https://zhouyisjtu.github.io/project_vcmeshcnn/vcmeshcnn.html
Future Work