Geometric Deep Learning on 3D Meshes
an overview

Rasmus R. Paulsen
DTU Compute

Based on contributions from (among others):

Kristine Aavild Juhl
Christian Keilstrup Ingwersen
Patrick Møller Jensen
Mathias Micheelsen Lowes
Bjørn Marius Schreblowski Hansen
Anders Bjorholm Dahl
Vedrana Andersen Dahl
Who is this aimed at?

- The ideal audience
  - Limited practical experience with geometrical deep learning
  - Has a good understanding of basic convolutional neural networks
    - Has seen the U-net before
  - Might come in a situation where your data is actually 3D meshes or have been *magicked* into 3D meshes
  - Would like to do surface based classification or labelling / segmentation
  - Lacks a good starting point
    - Which approach is good for my data
What is your experience with geometric deep learning?

- This is the first time I hear about it
- I have superficial knowledge about the field
- I have read several articles about it
- I have tested an existing framework
- I have adapted an existing framework to my own data
- I have coded my own framework
What’s in it for me?

- You will (hopefully) get an overview of different approaches to work with 3D meshes
- Some understanding of the strengths and weaknesses of the different methods
  - How invariant the methods are to geometric transformation (translation, rotations etc)
  - How large meshes can they process?
  - What are the restriction with regards to geometry/topology
  - How do they handle noise?
What is my interest in the field?

I am here for the ECTS, the social network and of general interest

I am working with data that might benefit from geometric deep learning

I have a theoretical interest in the field and would like to advance the theory in the field

Something else
Surfaces – where do they come from?

Direct surface scanning using a Canfield Vectra facial scanner.
Object scanners

An ear impression scanned by a 3Shape scanner.
Probably one of the most scanned anatomies in the world

The founder of DTU – H. C. Ørsted
Scanned by Dolores Messer with a custom built structured light scanner at DTU Compute

Eiríksson et al. "Precision and accuracy parameters in structured light 3-D scanning." International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 5 (2016)
Iso-surfaces or pixel-wise classifications
CAD Models

Important properties of meshes

- Rotational aspects (geometric invariances)
- Size (number of vertices and faces)
- Topology and if it is "manifold"
- Mesh sampling and noise properties
Translation and rotational aspects

- Does it make sense to have a "canonical orientation" of your objects?
- Does the method require that the objects are pre-oriented?
- Translation is often fixed by aligning center-of-mass
  - Not a universal solution
# Mesh sizes

<table>
<thead>
<tr>
<th>Type</th>
<th>Vertices</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapenet model (CAD)</td>
<td>Hundreds (guess)</td>
<td>Hundreds (guess)</td>
</tr>
<tr>
<td>Facial scan with accuracy ~0.5 mm</td>
<td>110.000</td>
<td>35.440</td>
</tr>
<tr>
<td>Left atrium from CT scan (voxel size 0.50 mm^3) (iso-surface)</td>
<td>35.000</td>
<td>65.000</td>
</tr>
<tr>
<td>Scanned H. C. Ørsted (accuracy 150 mikrometer)</td>
<td>1.375.930</td>
<td>2.751.840</td>
</tr>
<tr>
<td>Full head model with accuracy ~1 mm</td>
<td>450.000</td>
<td>830.000</td>
</tr>
<tr>
<td>FAUST human body (processed)</td>
<td>6.890</td>
<td></td>
</tr>
</tbody>
</table>
What are the topological equivalences of the three meshes?

Sphere, Sphere, Plane
Sphere, Plane, Plane
Sphere, Plane, Tube
Plane, Sphere, Tube
Sphere, Sphere, Sphere
What are the topological equivalences of the three meshes?

- Sphere, Sphere, Plane
- Sphere, Plane, Plane
- Sphere, Plane, Tube
- Plane, Sphere, Tube
- Sphere, Sphere, Sphere
What are the topological equivalences of the three meshes?

- Sphere, Sphere, Plane
- Sphere, Plane, Plane
- Sphere, Plane, Tube
- Plane, Sphere, Tube
- Sphere, Sphere, Sphere
Mesh topology

- Topologically equivalent to a sphere, plane, tube, donut?
- or something far far beyond?
- Is it "manifold"?

Mesh sampling and noise?

- Are the vertices sampled equally over the underlying surface?
- Are the faces/triangles well shaped?
  - Classical *marching cubes* makes notoriously bad aspect ratio triangles
- What is the nature of the sampling noise?
  - Outliers, Gaussian or something else?
A mesh biopsy

- Raw facial scan from BU-3DFE – a reference dataset
- “Mesh in the wild”
  - representative for current facial scanners
- 106,320 vertices and 35,440 faces

"A 3D Facial Expression Database For Facial Behavior Research" by Lijun Yin; Xiaozhou Wei; Yi Sun; Jun Wang; Matthew J. Rosato, 7th International Conference on Automatic Face and Gesture Recognition, 10-12 April 2006 P:211 - 216
A mesh biopsy

- Looks topologically to be a plane
  - but it is not
- Flipped triangles
- Non-manifold parts
- Complex noise issues
- A face has a canonical orientation
  - But facial scanners have many different coordinate systems
Do topology and artefacts matter?

- Quite a lot actually
  - A lot of the current methods have severe restrictions on topology and if the surfaces are manifold

- A crude comparison
  - Imagine your 2D CNN would crash and burn because of one single bad pixel due to a dead CCD cell

- A typical solution – preprocess the mesh so it is nice and clean
  - Often needs a specific solution for each dataset
  - Large risk of removing / smoothing out important information
My experience with the U-net

Never heard of it

I have superficial knowledge of the U-net

I have read several papers where the U-net is used

I have tried a pre-made U-net

I have coded my own custom version of the U-net
CNN recap – the U-net

Convolution – a conceptual heads-up

Your data – an image, a mesh, a graph or something more exotic

A kernel – containing (learnable) weights

For each “node” in your data you have values

For each “node” in your data you have a neighborhood that should be “covered” by the kernel

But first something completely different!
Approaches covered in the following

- Multi-view rendering approaches
- Volumetric approaches
- Methods that define convolutions on meshes
- Methods based on implicit representations of meshes.
  - For example implicit functions on grids and signed/un-signed distance fields
- Hybrid methods based on mesh operations for convolutions and pooling

Disclaimer: It will mostly be a conceptual overview
I am certainly not a specialist on all approaches.
Multi-view Convolutional Neural Networks for 3D Shape Recognition

Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller
University of Massachusetts, Amherst
{hsu, smaji, kalo, elm}@cs.umass.edu

1900 google scholar citations per August 2021

Multi-view convolutional neural networks for 3d shape recognition

- Object classification based on 3D shapes
- Rendering pipeline
- Standard 2D CNN to do the classification
Multi-view convolutional neural networks for 3d shape recognition – rendering setup

- 12 positions with rotations around the z-axis
- 80 views
  - 20 vertices of an icosahedron enclosing the shape
  - 4 rotations around camera axes
Multi-view convolutional neural networks for 3D shape recognition – network

CNN1 is pre-trained on ImageNet

2D rendered images → our multi-view CNN architecture
Multi-view convolutional neural networks for 3d shape recognition – results

Princeton ModelNet
- 128K 3D CAD models
- 662 categories

ModelNet40
- 12K models
- 40 categories

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Config.</th>
<th>Test Config.</th>
<th>Classification (Accuracy)</th>
<th>Retrieval (mAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre-train</td>
<td>Fine-tune</td>
<td>#Views</td>
<td>#Views</td>
</tr>
<tr>
<td>(1) SPH [16]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(2) LFD [5]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(3) 3D ShapeNets [37]</td>
<td>ModelNet40</td>
<td>ModelNet40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(4) FV</td>
<td>-</td>
<td>ModelNet40</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>(5) FV, 12×</td>
<td>-</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(6) CNN</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>(7) CNN, f.t.</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>(8) CNN, 12×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>(9) CNN, f.t.12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(10) MVCNN, 12×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>(11) MVCNN, f.t., 12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(12) MVCNN, f.t.+metric, 12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(13) MVCNN, 80×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>(14) MVCNN, f.t., 80×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>(15) MVCNN, f.t.+metric, 80×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

* f.t.=fine-tuning, metric=low-rank Mahalanobis metric learning
Multi-view convolutional neural networks for 3d shape recognition – some observations

- If you can render your object – you can classify it
  - Robust to topology variations, large mesh sizes, noise

- Pre-aligning an object to a canonical orientation is ill-posed
  - the view sequence is somewhat arbitrary
  - Only partially rotationally invariant
3D landmark prediction

- Given a set of rendered faces
- 2D landmark positions are estimated
- A predicted landmark in 2D corresponds to a line in space
What can RANSAC do for me here?

- Sample random positions in space for view directions
- Render coherent images of skin
- Robustly estimate a line crossing avoiding outlier influence
- Effectively computing intersection between rays and a triangulated surface
What can RANSAC do for me here?

- Sample random positions in space for view directions
- Render coherent images of skin
- Robustly estimate a line crossing avoiding outlier influence
- Effectively computing intersection between rays and a triangulated surface
What can RANSAC do for me here?

- Sample random positions in space for view directions
- Render coherent images of skin
- Robustly estimate a line crossing avoiding outlier influence
- Effectively computing intersection between rays and a triangulated surface

When poll is active, respond at pollev.com/rasmuspaulse538
Least squares and RANSAC
Using trained network on MR data

Trained on the z-buffer / distance map

http://shapeml.compute.dtu.dk/

Works with significant amount of surface noise
Volumetric CNN for object classification - occupancy representation

30 x 30 x 30 occupancy grid


Volumetric CNN for object classification
Volumetric CNN for object classification – some observations

- If you can turn your object solid – you can classify it
  - Can only handle closed surfaces

- Pre-aligning an object to a canonical orientation is ill-posed
  - Only partially rotationally invariant

- Massive loss of resolution when using this volumetric representation
Extrinsic vs. intrinsic

Convolution – a conceptual heads-up

Your data – an image, a mesh, a graph or something more exotic

A kernel – containing (learnable) weights

For each "node" in your data you have values

For each "node" in your data you have a neighborhood that should be "covered" by the kernel
How many edge neighbours does an edge have in its 1-ring neighborhood?

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
How many edge neighbours does an edge have in its 1-ring neighborhood?
How many edge neighbours does an edge have in its 1-ring neighbourhood?
Hybrid methods based on mesh operations for convolutions and pooling

- MeshCNN used for semantic segmentation of 3D objects.
- The labelling is done per edge
- To the left the result of the segmentation
- Second, third and fourth row show simplified/reduced/pooled meshes


[https://ranahanocka.github.io/MeshCNN/](https://ranahanocka.github.io/MeshCNN/)
MeshCNN – node (edge) data (features)

- **Five features per edge:**
  - The dihedral angle
  - The two inner angles
  - The two edge-length ratios

- **Neighborhood of edge** $e$

- **Invariant to translation, scaling and rotation**

\[
(e_1, e_2, e_3, e_4) = (a + c, b + d, |a - c|, |b - d|)
\]
MeshCNN – convolutions

- Symmetric features on 1-ring neighbors
- Normal features for edge itself, $e_0$
- 1 x 5 standard 2D convolutions

$$(e_1, e_2, e_3, e_4) = (a + c, b + d, |a - c|, |b - d|)$$
MeshCNN – pooling / unpooling

- The edge with the feature vector of lowest magnitude is collapsed – similar to standard mesh decimation
- Five edges $\rightarrow$ Two edges
- Bookkeeping matrix $G$ (size $\#\text{edge} \times \#\text{edge}$)
MeshCNN – network architectures

<table>
<thead>
<tr>
<th>Segmentation (Down)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResConv ( f_{in} \times 32 )</td>
</tr>
<tr>
<td>MeshPool → 1800</td>
</tr>
<tr>
<td>ResConv 32 × 64</td>
</tr>
<tr>
<td>MeshPool → 1350</td>
</tr>
<tr>
<td>ResConv 64 × 128</td>
</tr>
<tr>
<td>MeshPool → 600</td>
</tr>
<tr>
<td>ResConv 128 × 256</td>
</tr>
</tbody>
</table>

Symmetric up- and down path
MeshCNN with U-net architecture
Based on BSc work of Bjørn Marius Schreblowski Hansen & Mathias Micheelsen Lowes
MeshCNN - results

Classification SHREC

<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>

Human Body Segmentation

<table>
<thead>
<tr>
<th>Method</th>
<th># Features</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeshCNN</td>
<td>5</td>
<td>92.30%</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>

[Ezuz et al. 2017] [2018]

https://ranahanocka.github.io/MeshCNN/

MeshCNN - observations

- Achieved impressive segmentation results on standard datasets
- Invariant to rotation, scaling and translation
- Limited to small meshes with a few hundred edges
  - Due to $N^2$ memory footprints (in matrix $G$)
- Vulnerable to mesh topology and surfaces being manifold
  - Can create non-manifold surfaces during pooling
Sparse MeshCNN with attention - paper in review

Based on BSc work of Bjørn Marius Schreblowski Hansen & Mathias Micheelsen Lowes

Prediction of intersection between the left atrium and the left atrial appendage in the human heart. For simulation of surgical device insertion.
Sparse MeshCNN

- In MeshCNN
  - The matrix $G$ is of size $n_e^2$
  - Scales quadratically with mesh size

- In Sparse MeshCNN
  - The matrix $G$ is sparse
  - Can operate on larger meshes
Methods based on convolutions on meshes
Convolutional Layers

MessagePassing

Base class for creating message passing layers of the form

GCNConv

The graph convolutional operator from the "Semi-supervised Classification with Graph Convolutional Networks" paper

ChebConv

The Chebyshev spectral graph convolutional operator from the "Convolutional Neural Networks on Graphs With Fast Localized Spectral Filtering" paper

GrahSAGEconv

The GraphSAGE operator from the "Inductive Representation Learning on Large Graphs" paper

GraphConv

The graph neural network operator from the "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks" paper

GravNetConv

The GravNet operator from the "Learning Representations of Irregular Particle-detector Geometry with Distance-weighted Graph Networks" paper, where the graph is dynamically constructed using nearest neighbors.

GatedGraphConv

The gated graph convolution operator from the "Gated Graph Sequence Neural Networks" paper

ResGatedGraphConv

The residual gated graph convolutional operator from the "Residual Gated Graph ConvNets" paper

GATConv

The graph attentional operator from the "Graph Attention Networks" paper

GATv2Conv

The GATv2 operator from the "How Attentive are Graph Attention Networks?" paper, which fixes the static attention problem of the standard GAT layer, since the linear layers in the standard GAT are applied right after each other, the ranking of attended nodes is unconditioned on the query node.

TransformerConv

The graph transformer operator from the "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification" paper

GAINConv

The graph attentional propagation layer from the "Attention-based Graph Neural Network for Semi-Supervised Learning" paper

TANConv

The topology adaptive graph convolutional networks operator from the "Topology Adaptive Graph Convolutional Networks" paper

GINConv

The graph isomorphism operator from the "How Powerful are Graph Neural Networks?" paper

GINConv

The modified GINConv operator from the "Strategies for Pre-training Graph Neural Networks" paper

ARMAConv

The ARMA graph convolutional operator from the "Graph Neural Networks with Convolutional ARMA Filters" paper

SimpConv

The simple graph convolutional operator from the "Simplifying Graph Convolutional Networks" paper

APPNP

The approximate personalized propagation of neural predictions layer from the "Predict then Propagate: Graph Neural Networks meet Personalized PageRank" paper
### Pooling Layers

<table>
<thead>
<tr>
<th>Pooling Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopPooling</td>
<td>The self-attention pooling operator from the “Self-Attention Graph Pooling” and “Understanding Attention and Generalization in Graph Neural Networks” papers.</td>
</tr>
<tr>
<td>SNAPooling</td>
<td>The edge pooling operator from the “Towards Graph Pooling by Edge Contraction” and “Edge Contraction Pooling for Graph Neural Networks” papers.</td>
</tr>
<tr>
<td>PIPApooling</td>
<td>The path integral based pooling operator from the “Path Integral Based Convolution and Pooling for Graph Neural Networks” paper.</td>
</tr>
<tr>
<td>MemPooling</td>
<td>Memory based pooling layer from “Memory-Based Graph Networks” paper, which learns a coarsened graph representation based on soft cluster assignments</td>
</tr>
<tr>
<td>max_pool</td>
<td>Pools and coarsens a graph given by the torch_geometric.data.Data object according to the clustering defined in <code>cluster</code>.</td>
</tr>
<tr>
<td>avg_pool</td>
<td>Pools and coarsens a graph given by the torch_geometric.data.Data object according to the clustering defined in <code>cluster</code>.</td>
</tr>
<tr>
<td>max_pool_x</td>
<td>Max Pools node features according to the clustering defined in <code>cluster</code>.</td>
</tr>
<tr>
<td>max_pool_neighbor_x</td>
<td>Max pools neighboring node features, where each feature in <code>data.x</code> is replaced by the feature value with the maximum value from the central node and its neighbors.</td>
</tr>
<tr>
<td>avg_pool_x</td>
<td>Average pools node features according to the clustering defined in <code>cluster</code>.</td>
</tr>
<tr>
<td>avg_pool_neighbor_x</td>
<td>Average pools neighboring node features, where each feature in <code>data.x</code> is replaced by the average feature values from the central node and its neighbors.</td>
</tr>
<tr>
<td>greedy_voxel</td>
<td>A greedy clustering algorithm from the “Weighted Graph Cuts without Eigenvectors: A Multilevel Approach” paper of picking an unmarked vertex and matching it with one of its unmarked neighbors (that maximizes its edge weight).</td>
</tr>
<tr>
<td>voxel_grid</td>
<td>Voxel grid pooling from the, e.g., “Dynamic Edge-Conditioned Filters in Convolutional Networks on Graphs” paper, which overlays a regular grid of user-defined size over a point cloud and clusters all points within the same voxel.</td>
</tr>
<tr>
<td>fps</td>
<td>A sampling algorithm from the “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space” paper, which iteratively samples the most distant point with regard to the rest points.</td>
</tr>
<tr>
<td>knn</td>
<td>Finds for each element in <code>y</code> the k nearest points in <code>x</code>.</td>
</tr>
<tr>
<td>knn_graph</td>
<td>Computes graph edges to the nearest k points.</td>
</tr>
<tr>
<td>radius</td>
<td>Finds for each element in <code>y</code> all points in <code>x</code> within distance <code>r</code>.</td>
</tr>
<tr>
<td>radius_graph</td>
<td>Computes graph edges to all points within a given distance.</td>
</tr>
<tr>
<td>nearest</td>
<td>Clusters points in <code>x</code> together which are nearest to a given query point in <code>y</code>.</td>
</tr>
</tbody>
</table>
Convolutions on meshes

Convolution – a conceptual heads-up

Your data – an image, a mesh, a graph or something more exotic

A kernel – containing (learnable) weights

For each “node” in your data you have values

For each “node” in your data you have a neighborhood that should be “covered” by the kernel
Convolutions on meshes

- Main differences between approaches
  - How is a node neighborhood defined / computed
  - What values are used per node
  - How are the weights in the convolutions defined

- How are we dealing with kernel rotational invariance?

One example - MoNet

MoNet – vertex features

- Vertex features should represent local geometry
- Local shape signature
  - Histogram of local normal vectors
  - 544 dimensional vector (per vertex)


MoNet – vertex features.
Bam! Back to classical shape matching

The local shape descriptor used in MoNet is similar to 3D extensions of shape contexts – and comes with its own choices, strengths and weaknesses.

Local reference frame

- The local reference frame is the per-vertex coordinate system.
- Determines the orientation of SHOT feature extractor.
- Might determine the orientation of the local convolution patch.
  - Unless convolution is taken as the maximum over all rotations (around the normal) of the patch.
Inconsistent local reference frame

- Imagine that you had no general orientation of your 2D image
- For each pixel, your convolution kernel has an arbitrary orientation
- That is the default situation with 3D meshes
Inconsistent local reference frame

- One approach:
  - Compute local reference frame (coordinate system) following the gradient in the image
  - Convolutions would be following gradients – maybe good – maybe bad

- Another approach: Rotate kernel and take the maximum output...very expensive
Computing a Local reference frame - using a 3D equivalent of gradients/curvature

- Sample points in a local neighborhood
- Do eigenvector decomposition
- 3 Eigenvectors
  - One is the normal (smallest eigenvalue)
  - Two follow the surface
- Normally inconsistent and ambiguous
- Reference below claims to have solved it
- Used in MoNet (as far as I understand the paper)

MoNet and similar methods – observations

- Choice of local features to represent geometry
- Are they dependent on a consistent local reference frame?
- Topological constraints?

A note on spectral methods

- There is a strong relation between Fourier analysis and convolutions for 1D and 2D signals
- This can be replicated on 3D meshes and is also related to the mesh Laplacian
- Quite a lot of spectral methods have been published
- It seems that they are losing popularity and they are beyond the scope and time of this presentation
- Some comments can be found in
Deep learning with implicit functions
The signed distance function

- Voxel grid – each voxel contains a scalar value
- Carries information about the shape in the entire field
DeepSDF

\[ f_\theta(x) \approx SDF(x), \forall x \in \Omega. \]

(b) Coded Shape DeepSDF

DeepSDF – single shape representation

\[ f(p) \leftrightarrow s, \quad p \in \mathbb{R}^3, s \in \mathbb{R} \]

Distance from \((x,y,z)\) to surface

\[ p = (x,y,z) \]
DeepSDF – multiple shape representation

Multiple shape representation

\[ f(z, p) \mapsto s, \quad p \in \mathbb{R}^3, s \in \mathbb{R} \]

Decoder

Distance from (x,y,z) to surface

p = (x,y,z)
DeepSDF - Training

• Training:

\[ X_t = \{ (p_j, s_j) : \quad s_j = DF^t(p_j) \} \]

\[
\arg\min_{\theta, \{z_i\}_{i=1}^N} \sum_{i=1}^N \left( \sum_{j=1}^K \mathcal{L}(f_\theta(z_i, p_j), s_j) + \frac{1}{\sigma^2} \|z_i\|^2 \right)
\]

Regularization

Clamped L1-distance

1. \( p = (x, y, z) \)
2. \( p = (x, y, z) \)

Parks et al., "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," CVPR2019
DeepSDF - results

Results:

- Reconstructing known shapes
- Reconstructing unknown shapes
- Shape completion
- Latent space shape interpolation

Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.
DeepSDF - observations

- You should be able compute a signed distance to your mesh
  - Needs closed surfaces

- Not rotational invariant – unless you do heavy data augmentation

- Can do shape classification, shape synthesis and shape completion
  - Has a very usable latent space
Other implicit approaches

Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion

Julian Chibane\textsuperscript{1,2} \quad Thiemo Alldieck\textsuperscript{1,3} \quad Gerard Pons-Moll\textsuperscript{1}

\textsuperscript{1}Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
\textsuperscript{2}University of Würzburg, Germany
\textsuperscript{3}Computer Graphics Lab, TU Braunschweig, Germany

{jchibane, gpons}@mpi-inf.mpg.de, alldieck@cpg.cs.tu-bs.de

Adversarial Generation of Continuous Implicit Shape Representations

Marian Kleinberg, Matthias Fey and Frank Weichert
TU Dortmund University, Germany

SAL: Sign Agnostic Learning of Shapes from Raw Data

Matan Atzmon and Yaron Lipman
Weizmann Institute of Science
Rehovot, Israel
Unsigned distance fields

Implicit Neural Distance Representation for Unsupervised and Supervised Classification of Complex Anatomies. Kristine Aavild Juhl et al. MICCAI 2021
Unsigned distance fields

- Can handle arbitrary topologies
- Meshing an unsigned distance field is very tricky

Implicit Neural Distance Representation for Unsupervised and Supervised Classification of Complex Anatomies. Kristine Aavild Juhl et al. MICCAI 2021
Shape classification using unsigned distance fields

**Fig. 3.** Visualization of the latent space projected to 2D using principle component analysis (PCA) for the 128 dimensional latent space of the three datasets (left) and the ESOF-faces only (right). Opaque: training set, Solid: test set.

Implicit Neural Distance Representation for Unsupervised and Supervised Classification of Complex Anatomies. **Kristine Aavild Juhl et al.** **MICCAI 2021**
That's is – the tour is over!
If you have data and need a way forward

- What is the nature of my data?
  - number of samples, number of vertices, topology, cleanness, canonical orientations?

- What is my goal?
  - segmentation, classification, shape correspondence, shape completion?

- What approach fits my data and goals?
  - can it handle your data (size is a main issue)
  - can it be adapted to solve your task?
  - are there any code available
  - what are the hardware requirements (mainly GPU memory size)
  - what are the software / operating system requirements?
What if I need a more theoretical research direction?

- Find your own niche that you want to explore
- Locate an unsolved problem
- In GDL there are lots of problems
  - but also a large number of people looking at them.
- You should have a competitive advantage
  - new idea, alternative approaches

- A new mesh convolution operator will probably have limited novelty