DualConvMesh-Net:
Joint Geodesic and Euclidean Convolutions on 3D Meshes

Jonas Schult*, Francis Engelmann*, Theodora Kontogianni, Bastian Leibe
RWTH Aachen University
{schult, engelmann, kontogianni, leibe}@vision.rwth-aachen.de

Abstract

We propose DualConvMesh-Nets (DCM-Net) a family of deep hierarchical convolutional networks over 3D geometric data that combines two types of convolutions. The first type, geodesic convolutions, defines the kernel weights over mesh surfaces or graphs. That is, the convolutional kernel weights are mapped to the local surface of a given mesh. The second type, Euclidean convolutions, is independent of any underlying mesh structure. The convolutional kernel is applied on a neighborhood obtained from a local affinity representation based on the Euclidean distance between 3D points. Intuitively, geodesic convolutions can easily separate objects that are spatially close but have disconnected surfaces, while Euclidean convolutions can represent interactions between nearby objects better, as they are oblivious to object surfaces. To realize a multi-resolution architecture, we borrow well-established mesh simplification methods from the geometry processing domain and adapt them to define mesh-preserving pooling and unpooling operations. We experimentally show that combining both types of convolutions in our architecture leads to significant performance gains for 3D semantic segmentation, and we report competitive results on three scene segmentation benchmarks. Our models and code are publicly available[1].

1. Introduction

Geometric deep learning [3, 4, 12, 26, 35, 43] aims at transferring the successes of CNNs from regular, discrete domains, *e.g.*, 1D audio, 2D images or 3D voxel grids, onto irregular data representations such as graphs, point clouds or 3D meshes. Currently, geometric deep learning is divided into two main areas relying on different data representations: *3D scene understanding* and *3D shape analysis*.

The former looks at tasks such as semantic segmentation [9, 17, 21, 40, 45, 47, 62], instance segmentation [14, 15, 19, 28, 59, 60] and part segmentation [21, 45, 53, 62]. Here, the focus lies primarily on processing point cloud
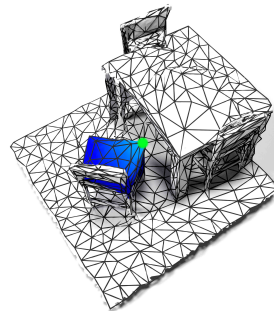
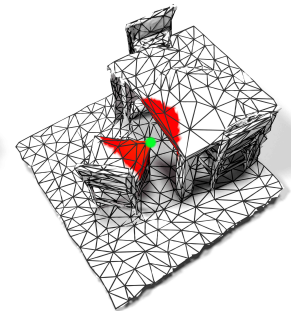*Geodesic Neighborhood*   *Euclidean Neighborhood*

Figure 1: **Comparison of geodesic (left) and Euclidean neighborhoods (right).** Our DCM-Net combines geodesic and Euclidean convolutions. Geodesic convolutions follow the surface of individual objects, which can be beneficial to learn specific object shapes. Euclidean convolutions can bridge over small gaps, which can encourage the flow of relevant context information between spatially nearby but geodesically distant objects, while being able to connect disconnected parts due to scanning artifacts. The color gradient shows the geodesic and Euclidean distances between the ● center point and its neighbors. The scan section is taken from the ScanNet dataset [8].

data. One choice is to project raw point clouds into a discrete 3D grid representation, which enables standard 3D CNNs to be applied, *i.e.*, by sliding kernels over neighboring voxels [10, 42, 55, 63, 64]. Alternative approaches operate directly on raw point clouds [2, 29, 39, 45, 46, 53, 65]. In this case, the challenge consists in defining convolutional operators over point sets. Commonly, the convolutional kernels are applied to local point neighborhoods obtained from spherical or k-nn neighborhoods defined over the Euclidean distance between pairs of points. We refer to these convolutions as *Euclidean convolutions* (see Figure 1, right). Consequently, regardless of point cloud or voxel representations, these convolutions are agnostic to the surface information and, therefore, sensitive to surface deformations.

Unlike 3D scene understanding, *3D shape analysis* is concerned with tasks such as shape correspondence [3], shape descriptors [33], and shape retrieval [43]. As opposed to the methods mentioned earlier, shape analysis fo-

---

cuses on the surface information encoded in meshes or graphs. Here, convolutional kernels are defined over local patches or neighborhoods on the surface of a mesh or graph. These neighborhoods are localized by the *geodesic* distance between nodes on the surface mesh, *i.e.*, points that are reachable by one edge connection along the surface mesh. We therefore refer to them as *geodesic convolutions* (see Figure 1, left). A notable property of geodesic convolutions is their invariance to surface deformations, which is generally desired in tasks such as shape correspondence.

In this work, we investigate the role of *geodesic* and *Euclidean* convolutions in the task on 3D semantic segmentation of 3D meshes. So far, few approaches have made use of explicit surface information and geodesic convolutions for semantic scene segmentation [24, 30, 54], whereas Euclidean convolutions are very popular in the field, *e.g.*, [2, 7, 29, 56, 65]. As visualized in Figure 1, both approaches have their own characteristics. While geodesic convolutions follow the surface to learn specific object shapes, Euclidean convolutions encourage the feature propagation over geodesically remote areas to accumulate contextual information. It is therefore natural to ask how these advantages can be combined in a common architecture. This is the question we address in this work.

We propose a novel deep hierarchical architecture, *DualConvMesh-Nets*, that starts from a mesh representation and combines both types of convolutions. In order to design such a hierarchical architecture that is capable of learning useful Euclidean and geodesic features at different scales, it is critical to define a mesh pooling algorithm which maintains a meaningful mesh structure throughout all mesh levels. We therefore adapt *vertex clustering (VC)* [50] and *Quadric Error Metrics (QEM)* [20], two well-established mesh simplification approaches from the geometry processing domain, in order to define meaningful pooling and unpooling operations on meshes. We introduce *Pooling Trace Maps* as an efficient way to keep track of vertex connectivity for pooling and unpooling. As a practical way of reducing the dependency to local vertex densities, we propose *Random Edge Sampling (RES)* for radius neighborhoods [27] .

Our proposed DCM-Net architecture achieves competitive results on the popular ScanNet v2 benchmark [8], as well as on Stanford 3D Indoor Scenes dataset [1]. For graph convolutional approaches, we define a new state-of-the-art on both datasets. Furthermore, we achieve state-of-the-art performance on the recent Matterport3D [5] benchmark.

In summary, the main contributions of this paper are: ① We propose a novel family of deep convolutional networks, *DCM-Nets*, that operate in both the Euclidean and geodesic space. ② We adapt two theoretically well-founded mesh simplification algorithms as means of pooling and unpooling in order to create multi-scale architectures on meshes, and we experimentally compare their per-

formance. ③ We introduce a novel sampling method on graph neighborhoods, *Random Edge Sampling*, which allows us to train networks with smaller sample sizes while evaluating them with better approximations. ④ We present a thorough ablation study, which empirically proves that combining Euclidean and geodesic convolutions provides a consistent benefit using radius neighborhoods, regardless of the pooling method used in the architecture.

2. Related work

Convolutions on point clouds. A simple way of handling point clouds is to transform them into a voxel grid representation that enables standard CNNs to be applied [8, 10, 42, 63, 64]. By construction, such approaches are limited to applying convolutional kernels on voxel neighborhoods, as it is not trivial to define geodesic neighborhoods on regular grids. Even recent methods focusing on efficient sparse voxel convolutions [7, 21] have similar limitations. Numerous other approaches operate directly on raw point clouds using convolutional kernels that are applied to the local neighborhoods of points obtained using k-nn or spherical neighborhoods [2, 38, 39, 46, 58]. Alternative methods define the position of the kernel weights explicitly in the Euclidean space relative to point positions [2, 29, 56, 65]. In both cases, the convolutional kernels are defined over the Euclidean space and are independent of the actual underlying object surface. In contrast, we additionally consider surface information using geodesic convolutions in combination with the Euclidean convolutions.

Convolutions on meshes and graphs. Spectral filtering methods build on eigenvalue decomposition of the graph Laplacian [12, 26, 35, 51]. While they work well on clean synthetic data, they are sensitive to reconstruction noise and do not generalize well across different graph structures. Local filtering methods, such as geodesic CNN [41], anisotropic CNN [3] or the work of Monti *et al.* [43] rely on handcrafted local coordinate systems defined over local patches on mesh surfaces. Verma *et al.* [57] replace these hand-designed pseudo-coordinates with a learned mapping between filter weights and graph patches. TextureNet [30] applies traditional CNNs to high resolution textures originating from geodesic mesh surfaces. Tangent-Convolutions [54] implicitly use surface information from estimated point normals by projecting point features on a local tangent plane and apply 2D CNNs. Whereas all previously mentioned methods perform convolutions on vertices, MeshCNN [23] defines them over the edges of a mesh.

In summary, these methods use surface information from a mesh or graph to run geodesic convolutions. Similarly, we consider geodesic convolutions as graph convolutions defined over the mesh and take special provisions to enable pooling operations such that all simplifications of the original mesh still contain meaningful geodesic information.
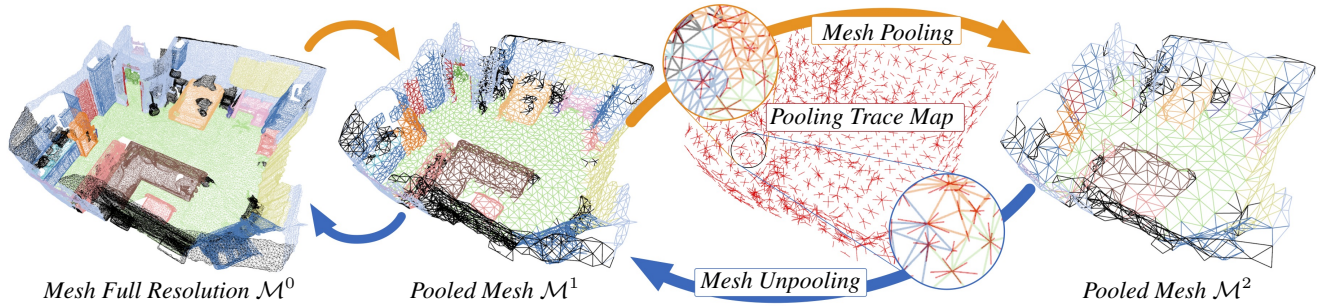
Figure 2: **Pooling on meshes.** To perform geodesic convolutions on multiresolution representations, the geodesic mesh neighborhood needs to be preserved throughout the pooling operations. We leverage vertex clustering and Quadric Error Metrics which both preserve meaningful geodesic neighborhoods in all mesh levels. These pooling operations rely on a *pooling trace map* (shown in red) that keeps track of vertex connectivity and is used for (un)pooling between adjacent mesh levels $\mathcal{M}^\ell$ with simple look-up operations.

Pooling operations on point clouds and meshes. Hierarchical networks operate on multiple resolution levels of a 3D model (see Figure 2), resulting in an increased receptive field of convolutions and robustness to small transformations. To obtain fine-to-coarse representations, different pooling operations exist. An important property of a pooling operation is whether it preserves the geometric and geodesic affinity information. On point clouds, random sampling of points or Farthest Point Sampling (FPS) are popular and effective approaches [39, 46, 62]. They work well on point clouds; however, when applied to mesh vertices the interconnectivity of vertices is lost. Hanocka *et al*. [23] perform mesh pooling by *learning* which edges to collapse. Unlike previous works, Tatarchenko *et al*. [54] propose to pool on a regular 3D grid. On meshes, Defferrard *et al*. [12] and Verma *et al*. [57] use the Graclus algorithm [13], while Ranjan *et al*. [48] and Pan *et al*. [24] rely on the mesh simplifying Quadric Error Metrics [20]. These mesh simplification approaches aim at reducing the number of vertices while introducing minimal geometric distortion by collapsing vertex pairs along the way. However, this can lead to high-frequency signals in noisy areas.

In this work, we leverage two theoretically well-founded methods from the geometry processing domain: *vertex clustering (VC)* [50] and *Quadric Error Metrics (QEM)* [20]. In order to allow multiresolution processing, we introduce *Pooling Trace Maps* (see Figure 2) to ensure well-defined pooling and unpooling operations on meshes.

Sampling neighborhoods. Hermosilla *et al*. [27] and Thomas *et al*. [56] argue that k-nn graph approaches suffer from non-uniform point densities in point clouds. Thus, they propose to use radius graphs to define the notion of neighborhoods for vertices in the Euclidean space. However, very densely populated regions can lead to arbitrarily large neighborhoods, which introduces a computational burden for the algorithm. Sampling the neighborhood space becomes inevitable. Therefore, Hermosilla *et al*. [27] use Poisson Disk Sampling, which preserves the relative density distribution of the point cloud but restricts the maximal

density per cubic unit by the radius of the non-overlapping poisson disks. Then, the Kepler Conjecture [22] gives an upper bound for the neighborhood size. Thomas *et al*. [56] control the density of the point cloud by low-pass filtering it via grid subsampling. In concurrent work, Lei *et al*. [36] randomly sub-sample the neighborhood to obtain at most K samples for approximating the neighborhood set. In this work, we propose *Random Edge Sampling* which is similar in spirit to [36] but has a special appeal in its probabilistic interpretation of reducing the neighborhood size.

3. Method

We propose a novel family of deep hierarchical network architectures. DCM-Nets combine the previously mentioned benefits of geodesic graph convolutions on 3D surface meshes and Euclidean graph convolutions on 3D vertices in the spatial domain. An important feature of our proposed architecture family is its modularity, which allows us to measure the effects of all components individually. To apply geodesic graph convolutions on multiple mesh levels, we describe the necessary mesh-centric pooling operations, *i.e*., our extensions to vertex clustering and Quadric Error Metrics. The input to our method is a 3D mesh with vertex features, *e.g*., color and normals, and the outputs are learned features for each vertex of the input mesh, which are used for dense prediction tasks such as semantic segmentation.

Network architecture. Inspired by U-Net [49], our model is defined as an encoder-decoder architecture, where the encoder is symmetric to the decoder, including skip-connections between both. Our deep hierarchical architecture is depicted in Figure 3. At each mesh level $\mathcal{M}^\ell$, multiple *dual convolutions* are applied. Dual convolutions perform geodesic and Euclidean convolutions in parallel, and subsequently concatenate the resulting feature maps. As suggested by He *et al*. [25], we add residual connections such that gradients can by-pass the convolutions for better convergence. For pooling, we leverage vertex clustering and Quadric Error Metrics.
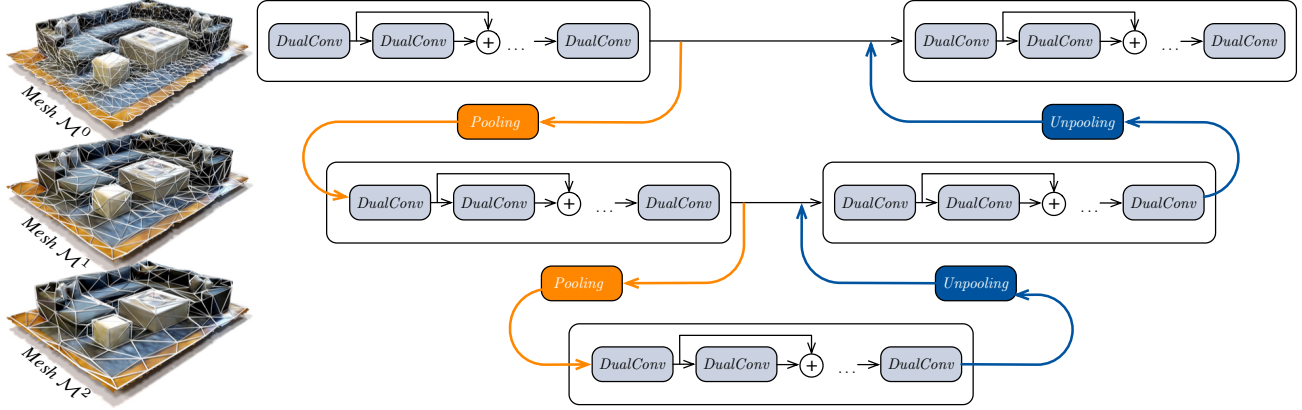
Figure 3: **Our deep hierarchical architecture** comprises several dual convolutions by-passed by skip connections in each mesh level and performs (un)pooling with pooling trace maps generated from mesh simplification algorithms.

Section 4.1 will present an ablation study, in which we disable each convolution type individually in order to measure its impact. We refer to an instantiation of our network that only operates in a single space as **Single**ConvMesh-Net (SCM-Net), whereas our full model operating in both spaces simultaneously is referred to as **Dual**ConvMesh-Net (DCM-Net). Note that SCM-Nets are a subset of the family of DCM-Nets since they equal DCM-Net if the number of filters is set to 0 everywhere for one of the spaces. An in-depth description including filter sizes and activation functions of both the SCM-Net and DCM-Net architecture is given in the supplementary material.

Euclidean and geodesic graph convolutions. We perform graph convolutions on the graph $\mathcal{G}^\ell = \left(\mathcal{V}^\ell, E^\ell \right)$ induced by the underlying mesh $\mathcal{M}^\ell$ in hierarchy level ℓ. The vertices of level ℓ are embedded in Euclidean 3D space, *i.e.*, $\mathcal{V}^\ell = \mathbb{R}^3$. The edge set $E^\ell = E_g^\ell \cup E_e^\ell$ is the union of the geodesic edge set E_g^ℓ, induced by the faces of $\mathcal{M}^\ell$, and the Euclidean edge set E_e^ℓ, obtained from the k-nn or radius graph neighborhood of each vertex $\mathbf{v}_i^\ell \in V^\ell$. Note that we neglect the level superscript ℓ when it eases readability. We implement convolutional layers over point features $\mathbf{x}_i$ associated with vertex $\mathbf{v}_i$ as for example in *EdgeConv* [61] or *FeaStNet* [57]. Specifically, the output feature $\mathbf{y}_i \in \mathbb{R}^E$ of vertex $\mathbf{v}_i$ with input feature $\mathbf{x}_i \in \mathbb{R}^F$ is computed as

$$\mathbf{y}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \varphi \left([\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i]; \theta \right) \qquad (1)$$

where $\mathcal{N}_i$ is the geodesic/Euclidean neighborhood of the vertex $\mathbf{v}_i$, $|\mathcal{N}_i|$ its cardinality, φ is a nonlinear function implemented as an MLP with trainable parameters θ and $[\cdot, \cdot]$ is the concatenation. Note that the number of kernel parameters θ is independent of the kernel size induced by the neighborhood $\mathcal{N}_i$. This is in contrast to 2D CNNs, where the number of parameters increases with the kernel size. By normalizing with $|\mathcal{N}_i|$, the convolutional layer is robust to variations in the number of neighbors.

On the very first convolution in the network, we define a translation invariant version of our convolutional layer which relies only on edge information. Specifically, we apply $\varphi(\cdot)$ to $\mathbf{x}_j - \mathbf{x}_i$ and do not concatenate the initial features containing absolute positions, (*c.f.* Equation 1). This makes it possible to train on scene crops, but evaluate on full rooms, which leads to broader context information for each vertex and decreases runtime during evaluation.

In contrast to DGCNN [61], we do not recalculate the neighborhoods in the learned feature space but we reuse the initial neighborhoods in the Euclidean and geodesic spaces. Skipping this dynamic recalculation of neighbors allows us to create deeper graph convolutional networks while enabling faster and more memory-efficient computations.

Alternative convolutional layers defined over relative vertex *positions* may also be used, such as *PointConv* [62] or *DPCC* [58]. However, in this work we focus on the neighborhood $\mathcal{N}_i$ which differentiates *geodesic* convolutions from *Euclidean* ones (see Figure 1): *Geodesic graph convolutions* define the geodesic neighborhood $\mathcal{N}_i^G$ of a vertex $\mathbf{v}_i$ as the 1-hop neighborhood, *i.e.*, all points that are reachable from the center vertex by one edge connection along the surface mesh. As such, the geodesic neighborhood $\mathcal{N}_i^G$ contains only points in the localized geodesic proximity of vertex $\mathbf{v}_i$. *Euclidean graph convolutions* rely on the Euclidean neighborhood $\mathcal{N}_i^E$ of a vertex $\mathbf{v}_i$ that is only constrained by the Euclidean distance. In this work, we obtain the Euclidean neighborhood $\mathcal{N}_i^E$ using a k-nn or radius graph. We compare both approaches in Section 4.1.

Random Edge Sampling (RES). Hermosilla *et al.* [27] argue that radius neighborhoods increase the robustness to non-uniformly sampled point clouds in contrast to k-nn ones. Since the simplification with QEM does not guarantee uniformly sampled mesh simplification, we rely on radius neighborhoods. However, radius neighborhoods may lead to arbitrarily many neighbors. We thus resort to sampling methods for reducing the computational load.

Motivated by Dropout[52], we define a novel sampling method on graph neighborhoods, called *random edge sampling (RES)*. RES randomly samples edges from the Euclidean edge set on all mesh levels. We define a function $D : \mathcal{N}_i \rightarrow [0,1]$ which maps the vertex neighborhood $\mathcal{N}_i$ of a given vertex $\mathbf{v}_i$ to its corresponding sampling probability, which is subsequently applied to all edges between vertices $\mathbf{v}_i$ and $\mathbf{v}_j \in \mathcal{N}_i$. D is defined as follows:

$$D(\mathcal{N}_i) = \begin{cases} 1 & \text{if } |\mathcal{N}_i| \leq T \\ (|\mathcal{N}_i| - (T-1))^{-\text{ld}(T+1)^{-1}} & \text{if } |\mathcal{N}_i| > T \end{cases} \quad (2)$$

Only edges $(\mathbf{v}_i, \mathbf{v}_j)$ connecting vertices $\mathbf{v}_j$ of the vertex neighborhood $\mathcal{N}_i$ whose size exceeds the threshold T are subject to sampling. We argue that the approximation with a neighborhood of small size is already limited and therefore, the neighborhood should not be further decimated. We visualize $D(\mathcal{N}_i)$ in Figure 4. Varying the threshold T equals to varying the expected number of vertices we draw from the vertex neighborhood distribution. By doing so, we introduce a larger variety to the training data and thus increase the generalization capability of our approach, while simultaneously reducing the computational load. We experience that decreasing the threshold for training still leads to a good approximation of the neighborhood.

Mesh simplification as a means of pooling. We interpret pooling operations as generating a hierarchy of mesh levels $(\mathcal{M}^0, ..., \mathcal{M}^\ell, ..., \mathcal{M}^{\mathcal{L}})$ of increasing simplicity interlinked by *pooling trace maps* $(\mathcal{T}^0, ..., \mathcal{T}^\ell, ..., \mathcal{T}^{\mathcal{L}-1})$ (see Figure 2). $\mathcal{M}^0$ is the mesh at its original resolution and $\mathcal{M}^{\mathcal{L}}$ is the coarsest representation after the final simplification operation. A pooling trace map $\mathcal{T}^\ell$ maps the elements of a vertex partition $\{\mathbf{v}_i^\ell\} \subset \mathcal{V}^\ell$ bijectively to a single representative vertex $\mathbf{v}^{\ell+1} \in \mathcal{V}^{\ell+1}$ in the next mesh level $\ell + 1$. Vertices of the mesh level $\mathcal{M}^{\ell+1}$ are interconnected by the edge set $E_g^{\ell+1}$ obtained by the mesh simplification algorithm. Similar to [24], on the features of $\{\mathbf{v}_i^\ell\}$, we propose to apply permutation invariant aggregation functions, *e.g.*, $\text{sum}(\cdot)$, $\text{max}(\cdot)$ or $\text{mean}(\cdot)$. To obtain pooled features for $\mathbf{v}^{\ell+1}$, we use *mean* aggregation for our experiments, in accordance to the definition of graph convolutions (Eq. 1).

As two well-approved methods from the geometry processing domain, we extend Vertex Clustering (VC) [50] and Quadric Error Metrics (QEM) [20] with pooling trace maps to achieve (un)pooling through simple look-up operations.

We modify the VC approach as follows: We place a 3D uniform grid with cubical cells of a fixed side length s over the input graph and group all vertices that fall into the same cell. We define $\mathbf{v}^{\ell+1}$ as the centroid $\mathbf{v}^{\ell+1} = |\{\mathbf{v}_i^\ell\}|^{-1} \sum \mathbf{v}_i^\ell$. Moreover, we store the mapping between the representative vertex $\mathbf{v}^{\ell+1}$ and its corresponding vertices $\{\mathbf{v}_i^\ell\}$ in the pooling trace map. A similar approach is followed in [54]. However, in order to perform geodesic convolutions on pooled graphs, the surface information, *i.e.*,
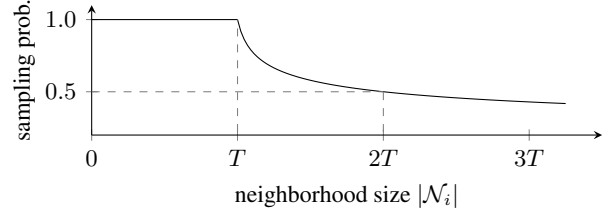

Figure 4: **Sampling probabilities for Random Edge Sampling**. Using the function $D(\mathcal{N}_i)$, RES only samples edges interconnecting the vertex $\mathbf{v}_i$ with neighboring vertices $\mathbf{v}_j \in \mathcal{N}_i$ if the neighborhood set $\mathcal{N}_i$ does not exceed the threshold T.

the *edges*, needs to be preserved as well. To achieve this, we first delete all edges between vertices that fall into the same cell, then we connect the representative vertices of those cells that were previously connected with at least one edge. Although the cell size s performs a low-pass filtering of the mesh vertex density and furthermore limits the introduced geometric error, this method is sensitive to the exact placement and orientation of the grid.

Alternatively, we consider QEM[20]. In contrast to VC, this approach incrementally contracts vertex pairs $(\mathbf{v}_1, \mathbf{v}_2)$ to a new representative $\bar{\mathbf{v}}$ according to an approximate error of the geometric distortion this contraction introduces. We keep track of these contractions and thus are able to generate pooling trace maps. Since QEM performs vertex contraction, we may contract vertices which are not adjacent in the mesh. This compensates for small scanning artifacts.

The same trace maps are used for unpooling a mesh from $\mathcal{M}^{\ell+1}$ to $\mathcal{M}^\ell$ by copying the features of $\mathbf{v}^{\ell+1}$ to its corresponding vertex $\mathbf{v}_i^\ell$. As VC aims for uniform vertex density and QEM for minimal geometric distortion, we compare both approaches in our ablation study in Section 4.1.

4. Experiments

We evaluate our method on three large scale 3D scene segmentation datasets, which contain meshed point clouds of various indoor scenes.

Stanford Large-Scale 3D Indoor Spaces (S3DIS)[1] contains dense 3D point clouds from 6 large-scale indoor areas, consisting of 271 rooms from 3 different buildings. The points are annotated with 13 semantic classes. It also includes 3D meshes, which are not semantically annotated. On average, each mesh contains $2 \cdot 10^5$ triangular faces [1]. As the resolution of these meshes is low compared to ScanNet v2 or Matterport3D, we oversample all faces and interpolate the color and ground truth information from the semantically annotated points. Our final predictions are then interpolated to the original point cloud to generate comparable results on the benchmark. We follow the common train/test split [1, 58, 55] and train on all areas except Area 5, which we keep for testing. we provide cross-validation mean IoU scores on all areas in the supplementary material.

Method	mIoU		Convolutional Category
	ScanNet	S3DIS	
PointNet [45]	-	41.1	Permutation Invariant Networks
PointNet++ [46]	33.9	-	
FCPN [11]	44.7	-	
3DMV [9]	48.3	-	2D-3D
JPBNet [6]	63.4	-	
MVPNet [31]	64.1	62.4	
TangentConv [54]	43.8	52.6	SurfaceConv
SurfaceConvPF* [24]	44.2	-	
TextureNet [30]	56.6	-	
PointCNN [39]	45.8	57.3	PointConv
ParamConv [58]	-	58.3	
DPC [16]	59.2	61.3	
MCCN [27]	63.3	-	
PointConv [62]	66.6	-	
KPConv [56]	68.4	**67.1**	
SparseConvNet [21]	72.5	-	Voxelized SparseConv
MinkowskiNet [7]	**73.4**	65.3	
DeepGCN [37]	-	52.5	GraphConv
SPGraph [40]	-	58.0	
SPH3D-GCN* [36]	61.0	59.5	
HPEIN [32]	61.8	61.9	
DCM-Net (**Ours**)	**65.8**	**64.0**	

Table 1: **Comparison to state-of-the-art.** Semantic segmentation mIoU scores on the offical ScanNet benchmark [8] and S3DIS Area-5 [1]. We outperform other graph convolutional approaches on all benchmarks. * indicates concurrent work. Full network definitions in the supplementary. ScanNet benchmark was accessed on 11/15/2019. S3DIS results as reported in original publications.

ScanNet v2 Benchmark [8]. We furthermore evaluate our architecture on the ScanNet v2 benchmark dataset. ScanNet contains 3D meshed point clouds of a wide variety of indoor scenes with reconstructed surfaces, textured meshes, and semantic ground truth annotations. The dataset contains 20 valid semantic classes. We perform all our experiments using the public training, validation, and test split of 1201, 312 and 100 scans, respectively. To validate our proposed components, the ablation study is conducted on the ScanNet validation set, where we report mean IoU scores.

Matterport3D [5]. Similar to ScanNet v2, Matterport3D contains meshed reconstructions of 90 building-scale RGB-D scans. We use the same evaluation protocol as introduced in 3DMV [9] and TextureNet [30] and report mean class accuracy scores on 21 classes on the test set.

Implementation and Training Details. We use VC and QEM to precompute the hierarchical mesh levels $\mathcal{M}^\ell$ interlinked with pooling trace maps $\mathcal{T}^\ell$. For VC, we set the cubical cell lengths to $4\,\mathrm{cm}$, $8\,\mathrm{cm}$, $16\,\mathrm{cm}$, and $32\,\mathrm{cm}$, respectively, for each mesh level. We experience that directly applying QEM on the full-resolution mesh results in high-frequency signals in noisy areas. Before applying QEM, we therefore first apply VC on the original mesh with a cubical

Input Mesh   **Ground Truth**   **Prediction**

● unlabeled ● wall ● floor ● cabinet ● bed ● chair ● sofa ● table ● door ● picture ● counter ● desk ● curtain ● fridge ● shower curtain ● toilet ● sink ● bathtub ● otherfurn
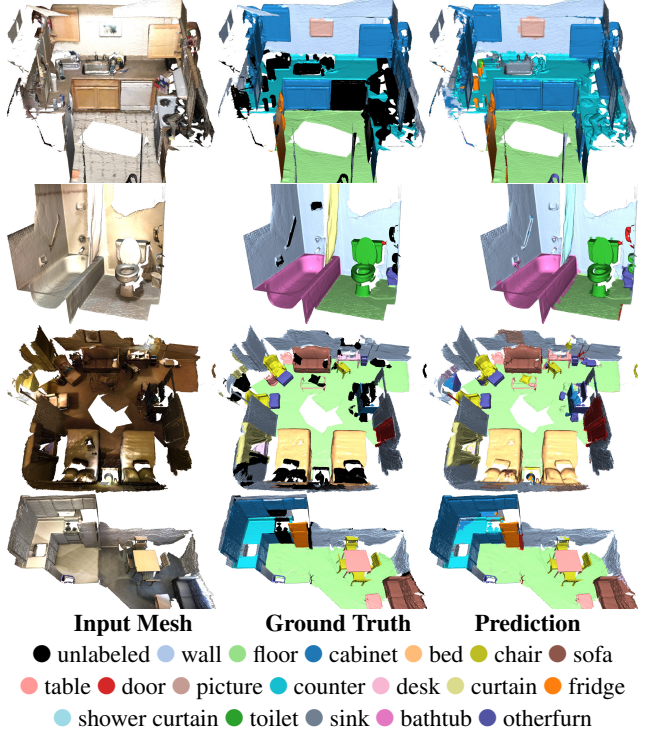
Figure 5: **Results on ScanNet v2 validation [8].** Our method correctly predicts challenging classes such as images and curtains, while maintaining clear boundaries. In the second row, our method correctly predicts shower ● curtain even though the ground truth is falsely labeled as regular curtain ●. Similarly in row three the partial ground truth label of the bottom right corner is properly predicted fully. There are some reasonable mistakes like the desk ● in row three labeled as table ●.

cell length of $4\,\mathrm{cm}$. For each mesh level, QEM simplifies the mesh until the vertex number is reduced to 30% of its preceding mesh level. As input features, we use the position, color, and normal of each vertex in the mesh. At each mesh level, we perform three dual convolutions (see Figure 3). We train the network end-to-end by minimizing the cross entropy loss using the Adam optimizer [34] with an initial learning rate of 10^{-3} and exponential learning rate decay of 0.5 after every 40 epochs and a batch size of 4.

It is common practice among recent approaches to discard training samples of low quality. Methods only differ in the used criteria: Qi *et al*. [45] reject training examples if the number of points in a training crop falls below a certain threshold. Analogously to our method, [46] rejects crops when the number of unlabeled points exceeds a threshold of 70%. We reject training crops, which have more than 80% unlabeled vertices, which corresponds to 0.8% of the $18,530$ cropped training samples of the ScanNet v2 train set. We do not apply this filtering during inference.

We conduct our experiments with random edge sampling with threshold $T = 15$ while training and $T = 25$ while testing, as we observed that a lower threshold for train-

Method	mAcc	wall	floor	cab	bed	chair	sofa	table	door	wind	shf	pic	cntr	desk	curt	ceil	fridg	show	toil	sink	bath	other
PointNet++ [46]	43.8	80.1	81.3	34.1	71.8	59.7	63.5	**58.1**	49.6	28.7	1.1	34.3	10.1	0.0	68.8	79.3	0.0	29.0	70.4	29.4	62.1	8.5
SplatNet [53]	26.7	**90.8**	**95.7**	30.3	19.9	**77.6**	36.9	19.8	33.6	15.8	15.7	0.0	0.0	0.0	12.3	75.7	0.0	0.0	10.6	4.1	20.3	1.7
TangentConv [54]	46.8	56.0	87.7	41.5	73.6	60.7	69.3	38.1	55.0	30.7	33.9	50.6	38.5	19.7	48.0	45.1	22.6	35.9	50.7	49.3	56.4	16.6
3DMV [9]	56.1	79.6	95.5	59.7	82.3	70.5	73.3	48.5	64.3	55.7	8.3	55.4	34.8	2.4	**80.1**	**94.8**	4.7	54.0	71.1	47.5	76.7	19.9
TextureNet [30]	63.0	63.6	91.3	47.6	82.4	66.5	64.5	45.5	69.4	60.9	30.5	**77.0**	**42.3**	44.3	75.2	92.3	49.1	**66.0**	80.1	**60.6**	86.4	27.5
DCM-Net (**Ours**)	**66.2**	78.4	93.6	**64.5**	**89.5**	70.0	**85.3**	46.1	**81.3**	**63.4**	**43.7**	73.2	39.9	**47.9**	60.3	89.3	**65.8**	43.7	**86.0**	49.6	**87.5**	**31.1**

Table 2: **Mean class accuracy scores on Matterport3D Test [5].** We outperform other approaches in 11 out of 21 classes. We use the same network definition as for the ScanNet v2 benchmark. Scores from [30].

ing reduces the computational load of the algorithm while learning useful features. Since we use a random sampling method for neighborhoods, the predictions vary in each run. We therefore run each evaluation 10 times and provide mean and standard deviations in our ablation study. Our models are implemented in PyTorch (Geometric) [18, 44] and trained on a Tesla V100 16GB.

Data Augmentation. From each mesh in ScanNet v2, we obtain $3\,\text{m} \times 3\,\text{m}$ crops with a stride of $1.5\,\text{m}$ from the ground plane. Since S3DIS and Matterport3D provide denser meshes, we reduce the crop size to $2\,\text{m} \times 2\,\text{m}$ with a stride of $1\,\text{m}$. Each cropped mesh is transformed by a random affine transformation, colors and positions are normalized to the range $[0, 1]$. Despite training on cropped meshes, we can perform inference on full meshes as the model is invariant to absolute vertex positions.

Results. Table 1 shows the performance of our approach compared to recent competing approaches on the ScanNet benchmark test dataset as well as S3DIS Area 5, grouped by the approaches' inherent categories. We are able to report state-of-the-art results for graph convolutional approaches by a significant margin of $4\,\%$ mIoU for the ScanNet benchmark, as well as $2.1\,\%$ mIoU for S3DIS Area 5. Only 4 approaches report better results on ScanNet. SparseConvNet [21] and MinkowskiNet [7] use Voxelized Sparse Convolutions, which currently perform best on ScanNet, but which are inherently limited for other tasks in that they cannot make use of detailed surface information. We also evaluated our algorithm on the novel Matterport3D dataset [5] and report overall state-of-the-art results on that benchmark in Table 2. Figure 5 shows our qualitative results on the ScanNet validation set. In the supplementary material, we provide our results on S3DIS in the k-fold test setting, as well as detailed descriptions of our models.

4.1. Ablation study.

We conduct an ablation study to support our claims that ① a mesh-centric pooling method in the shape of Quadric Error Metrics, and ② the combination of geodesic and Euclidean graph convolutions, and ③ random edge sampling for effectively sampling the neighborhood space independently contribute to an overall improved performance. In
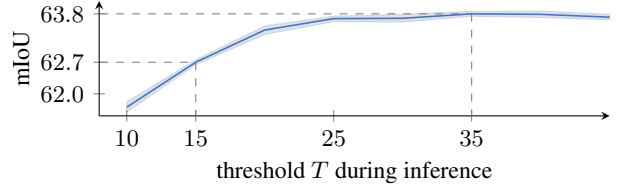

Figure 6: **Varying the threshold T during inference**. We observe that a smaller number of samples during test is sufficient for learning useful neighborhood features ($T = 15$). During test, we gain $1.1\,\%$ mIoU by increasing the threshold to $T = 35$. (Experiments conducted on S3DIS Area 5 with 10 runs for each threshold).

pool	arch	neighb	mIoU ($\pm$ stdev)	Δ
VC	Single	geo	57.1	-0.3
QEM	Single	geo	56.8	
VC	Single	knn	60.1	$+0.8$
QEM	Single	knn	60.9	
VC	Single	rad	61.9 (± 0.20)	$+2.0$
FPS	Single	rad	63.5 (± 0.13)	$+0.4$
QEM	Single	rad	63.9 (± 0.20)	
VC	Dual	knn/geo	59.7	$+3.2$
QEM	Dual	knn/geo	62.9	
VC	Dual	rad/geo	62.8 (± 0.12)	$+4.5$
QEM	Dual	rad/geo	67.3 (± 0.22)	

Table 3: **Comparison of pooling methods.** We compare Vertex Clustering (VC), Farthest Point Sampling (FPS), and Quadric Error Metrics (QEM) as pooling methods.

our study, we compare vertex clustering (VC), Quadric Error Metrics (QEM) and Farthest Point Sampling (FPS) as means of pooling in our DCM-Net architecture. We conduct experiments with the DCM-Net and SCM-Net instantiations of our architecture with different notions of neighborhoods for the Euclidean (k-nn and radius (rad)) as well as the geodesic domain (geo). For each EdgeConv in the SCM-Net architecture, we set the hidden feature size to 128 and the output size to 64. To enable a fair comparison, we halve the hidden and output feature size of DCM-Net architectures, such that the total number of feature channels is equal between the two versions. Note that this results in more than $15\,\%$ less parameters for DCM-Net architectures while performing better.

pool	arch	neighb	mIoU (± stdev)	Δ
VC	Single	geo	57.1	+2.6
VC	Single	knn	60.1	−0.4
VC	Dual	knn/geo	59.7	
QEM	Single	geo	56.8	+6.1
QEM	Single	knn	60.9	+2.0
QEM	Dual	knn/geo	62.9	
VC	Single	geo	57.1	+5.7
VC	Single	rad	61.9 (± 0.20)	+0.9
VC	Dual	rad/geo	62.8 (± 0.12)	
QEM	Single	geo	56.8	+10.5
QEM	Single	rad	63.9 (± 0.20)	+3.4
QEM	Dual	rad/geo	67.3 (± 0.22)	

Table 4: **Combining geodesic and Euclidean convolutions** in our DCM-Net brings significant performance improvements, especially compared to solely geodesic convolutions.

pool	arch	neighb	mIoU (± stdev)	Δ
QEM	Dual	geo/geo	56.3	+11.0
QEM	Dual	rad/rad	62.6 (± 0.21)	+4.7
QEM	Dual	rad/geo	67.3 (± 0.22)	
QEM	Single	rad	63.9 (± 0.20)	−1.3
QEM	Dual	rad/rad	62.6 (± 0.26)	
QEM	Single	geo	56.8	−0.5
QEM	Dual	geo/geo	56.3	

Table 5: **Architectural influence.** For the DCM-Net, we see improvements when using geodesic and Euclidean neighborhoods in parallel, in contrast to only using the same neighborhood notion.

Varying the expected sample size during test. In Equation 2, we introduce RES for reducing the expected size of the neighborhood set and therefore reducing the computational load. In Figure 6, we show the relationship between training a network with a relatively small sampled neighborhood and evaluating the algorithm with other set sizes. We experience that a small neighborhood size, *e.g.*, $T = 15$, during training is still sufficient to learn useful features. During test, we obtain better approximations of the neighborhood with larger thresholds, *e.g.*, $T = 35$, and report significantly better segmentation performances of +1 % mIoU. By decoupling the neighborhood size of the train and test times, we can adapt the expected size of the neighborhood to the computational resources given in the respective setting.

Comparison of pooling methods. In Section 3, we motivate to adapt the mesh simplification algorithms Vertex Clustering and Quadric Error Metrics as means of pooling using pooling trace maps. In Table 3, we evaluate the influence of different pooling methods in our architecture. As an additional experiment, we perform pooling using Farthest Point Sampling [46] on the underlying point cloud since it neglects the mesh structure. Therefore, we can only perform Euclidean graph convolutions in this setting. QEM performs significantly better than other pooling methods when using radius neighborhoods or the DCM-Net, while being on par with VC when considering k-nn or geodesic neighborhoods for SCM-Nets. We assume that the interplay between radius neighborhoods and QEM leads to this result. In contrast to VC, QEM does not aim for uniform vertex density. k-nn neighborhoods are sensible to varying vertex densities, as their spatial size is not limited [27].

Comparison of geodesic and Euclidean convolutions. In Section 3, we prompt the question whether a combination of geodesic and Euclidean convolutions leads to performance gains. In Table 4, we compare models using only geodesic convolutions, only Euclidean convolutions, and both combined in dual convolution modules, while keeping the pooling method fixed. We experience a clear trend that geodesic SCM-Net architectures fall behind their Euclidean counterparts, whereas the effect for radius neighborhoods is stronger than for k-nn ones. While the DCM-Net combining VC with k-nn falls behind its SCM-Net counterpart, the combination of geodesic and Euclidean neighborhoods in a DCM-Net architecture in all other settings outperforms the corresponding SCM-Net architectures. To evaluate the confounding factor of storage limitation for SCM-Nets with radius neighborhoods and QEM pooling, we test our model on a Titan RTX with 24 GB. We experience a performance of 65.9 %, *i.e.*, +2 % to the model trained on a 16 GB V100. To additionally prove that these performance gains do not just originate from the change to a DCM-Net architecture, we conduct further experiments in Table 5. Introducing our DCM-Net architecture leads to worse results in direct comparison with the SCM-Net architecture when using the same notion of neighborhood twice. We thus conclude that the improvements brought by the combination of neighborhoods is based on the design decision of combining geodesic and Euclidean neighborhoods and is not just due to architectural artifacts.

5. Conclusion

In this paper, we have motivated a mesh-centric view on 3D scene segmentation and we have proposed DCM-Nets to take advantage of the geometric surface information available in meshes. We hope that our work encourages fellow researchers to perform convolutions in both the geodesic and Euclidean domain, as we have empirically shown that this combination brings significant improvements independent to the architecture used. Future work might include incorporating geodesic convolutions for better separating instances in the task of 3D instance segmentation, as well as extending our work for leveraging point convolutions.

References

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5, 6

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point Convolutional Neural Networks by Extension Operators. *ACM Transactions on Graphics (TOG)*, 2018. 1, 2

[3] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2016. 1, 2

[4] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 2017. 1

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgbd data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2, 6, 7

[6] Hungyueh Chiang, Yenliang Lin, Yuehcheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. *International Conference on 3D Vision (3DV)*, 2019. 6

[7] Christopher Choy, Jun Young Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 6

[9] Angela Dai and Matthias Nießner. 3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation. In *IEEE European Conference on Computer Vision (ECCV)*, 2018. 1, 6, 7

[10] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[11] Rethage Dario, Wald Johanna, Sturm Juergen, Navab Nassir, and Tombari Federico. Fully-Convolutional Point Networks for Large-Scale Point Clouds. In *IEEE European Conference on Computer Vision (ECCV)*, 2018. 6

[12] M Defferrard, X Bresson, and P Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Neural Information Processing Systems (NIPS)*, 2016. 1, 2, 3

[13] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. Weighted Graph Cuts without Eigenvectors: A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence, (PAMI)*, 2007. 3

[14] Cathrin Elich, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. 3D Birds-Eye-View Instance Segmentation. In *German Conference on Pattern Recognition (GCPR)*, 2019. 1

[15] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi Proposal Aggregation for 3D Semantic Instance Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[16] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated Point Convolutions: On the Receptive Field Size of Point Convolutions on 3D Point Clouds. In *International Conference on Robotics and Automation (ICRA)*, 2020. 6

[17] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. In *IEEE European Conference on Computer Vision (ECCV'W)*, 2018. 1

[18] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 7

[19] Narita Gaku, Seno Takashi, Ishikawa Tomoya, and Kaji Yohsuke. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. 1

[20] Michael Garland and Paul S. Heckbert. Surface simplification using Quadric Error Metrics. In *Computer Graphics and Interactive Techniques*, 1997. 2, 3, 5

[21] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6, 7

[22] Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the kepler conjecture. In *Forum of Mathematics, Pi*, 2017. 3

[23] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (TOG)*, 2019. 2, 3

[24] Pan Hao, Liu Shilin, Liu Yang, and Tong Xin. Convolutional Neural Networks on 3D Surfaces Using Parallel Frames. *arXiv preprint arXiv:1808.04952*, 2018. 2, 3, 5, 6

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[26] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv preprint arXiv:1506.05163*, 2015. 1, 2

[27] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Alvar Vinacua, and Timo Ropinski. Monte Carlo Convolution for

Learning on Non-Uniformly Sampled Point Clouds. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018)*, 2018. 2, 3, 4, 6, 8

[28] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[29] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[30] Jingwei Huang, Haotian Zhang, Li Yi, Thomas A. Funkhouser, Matthias Nießner, and Leonidas J. Guibas. TextureNet: Consistent Local Parametrizations for Learning from High-Resolution Signals on Meshes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[31] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view pointnet for 3d scene understanding. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019. 6

[32] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 6

[33] Jin Xie, Yi Fang, Fan Zhu, and Edward Wong. Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[34] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations, (ICLR)*, 2015. 6

[35] Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations, (ICLR)*, 2017. 1, 2

[36] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical Kernel for Efficient Graph Convolution on 3D Point Clouds. *arXiv preprint arXiv:1909.09287*, 2019. 3, 6

[37] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 6

[38] Jiaxin Li, Ben M. Chen, and Gim H. Lee. So-Net: Self-Organizing Network for Point Cloud Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[39] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution On X-Transformed Points. In *Neural Information Processing Systems (NIPS)*, 2018. 1, 2, 3, 6

[40] Landrieu Loic and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 6

[41] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *IEEE International Conference on Computer Vision Workshops (ICCV'W)*, 2015. 2

[42] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 1, 2

[43] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[44] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Neural Information Processing Systems Workshop (NIPSW)*, 2017. 7

[45] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 6

[46] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Neural Information Processing Systems (NIPS)*, 2017. 1, 2, 3, 6, 7, 8

[47] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3D Graph Neural Networks for RGBD Semantic Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[48] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D Faces using Convolutional Mesh Autoencoders. In *IEEE European Conference on Computer Vision (ECCV)*, 2018. 3

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 3

[50] Jarek Rossignac and Paul Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in Computer Graphics*, 1993. 2, 3, 5

[51] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[52] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research (JMLR)*, 2014. 5

[53] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 7

[54] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent Convolutions for Dense Prediction in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3, 5, 6, 7

[55] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *International Conference on 3D Vision (3DV)*, 2017. 1, 5

[56] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 3, 6

[57] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 3, 4

[58] S. Wang, S. Suo, W.C. Ma, A. Pokrovsky, and R. Urtasun. Deep Parametric Continuous Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4, 5, 6

[59] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[60] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively Segmenting Instances and Semantics in Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[61] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019. 4

[62] Wenxuan Wu, Zhongang Qi, and Fuxin Li. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 4, 6

[63] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2

[64] Roynard X., Deschaud J.-E., and Goulette F. Classification of Point Cloud Scenes with Multiscale Voxel Deep Network. *arXiv preprint arXiv:1804.03583*, 2018. 1, 2

[65] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In *IEEE European Conference on Computer Vision (ECCV)*, 2018. 1, 2