MeshCNN Fundamentals: Geometric Learning through a Reconstructable Representation

AMIR BARDA*, Tel-Aviv University, Israel
YOTAM EREL*, Tel-Aviv University, Israel
AMIT H. BERMANO, Tel-Aviv University, Israel

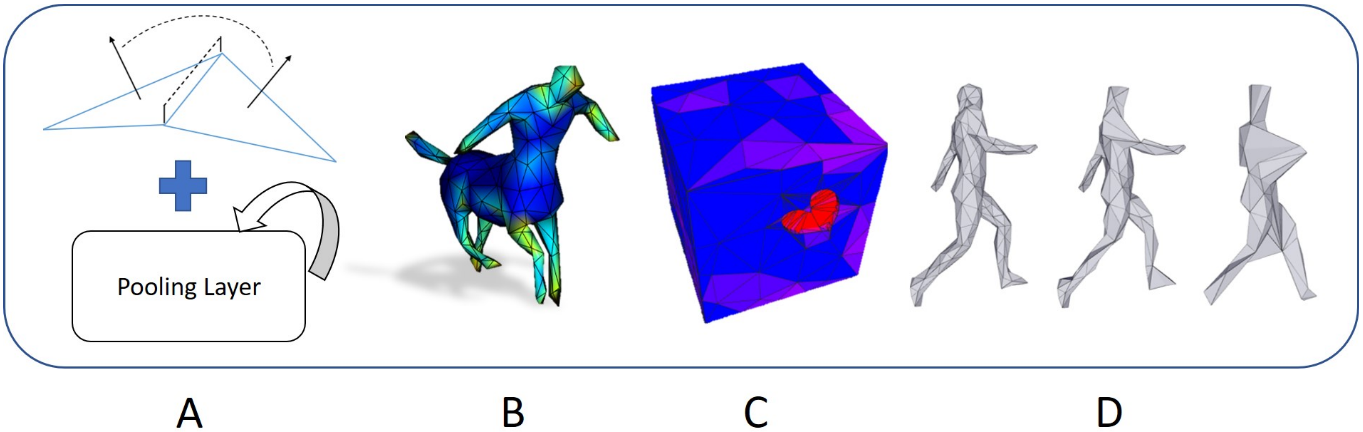


Fig. 1. Our Proposed Framework, consisting of a pooling scheme and the Fundamental Forms representation (a), based on the MeshCNN [2019] backbone. Our representation demonstrates high activations in salient and indicative regions (b). Using this scheme, the network indeed focuses on salient regions in deep layers (c). Our Pooling scheme results in more coherent pooling process (d)

Mesh-based learning is one of the popular approaches nowadays to learn shapes. The most established backbone in this field is MeshCNN. In this paper, we propose infusing MeshCNN with geometric reasoning to achieve higher quality learning. Through careful analysis of the way geometry is represented through-out the network, we submit that this representation should be rigid motion invariant, and should allow reconstructing the original geometry. Accordingly, we introduce the first and second fundamental forms as an edge-centric, rotation and translation invariant, reconstructable representation. In addition, we update the originally proposed pooling scheme to be more geometrically driven. We validate our analysis through experimentation, and present consistent improvement upon the MeshCNN baseline, as well as other more elaborate state-of-the-art architectures. Furthermore, we demonstrate this fundamental forms-based representation opens the door to accessible generative machine learning over meshes.

1 INTRODUCTION

What is a good representation for 3D geometry? This question stands at the core of the Computer Graphics field. A meaningful representation leads to powerful solutions to shape-related task, such as retrieval, analysis, manipulation, and generation. As in many other fields, the revolution of Deep Learning has led to many advancements, due to neural networks' semantic understanding and generalization capabilities. Since geometric deep learning is highly non-trivial [Bronstein et al. 2017], several recent research directions have emerged, employing different geometric representations [Çiçek et al. 2016; Hanocka et al. 2019; Maturana and Scherer 2015; Mildenhall et al. 2020; Park et al. 2019; Qi et al. 2017b].

We focus our discussion on learning over 2-manifold triangular meshes. Triangle meshes are the most popular choice for rendering and geometry processing pipelines. Their sparse yet continuous surface representation offers completeness and scalability. While currently not displaying state-of-the-art performance for shape analysis tasks, this representation has drawn notable attention over the last few years [Hanocka et al. 2019; Lahav and Tal 2020; Milano et al. 2020; Monti et al. 2018; Smirnov and Solomon 2021]. Perhaps the most seminal work in the field is MeshCNN [Hanocka et al. 2019]. This work defines edge-centric convolution and pooling operations for meshes, along with a set of features that describe the mesh for each edge. This work has inspired many follow-up research, proposing anything from novel attention layers [Milano et al. 2020], to employing the network's concepts for mesh editing tasks [Hertz et al. 2020; Liu et al. 2020].

In this paper, we analyze the way geometry is represented throughout the MeshCNN network, and introduce higher quality learning through geometric reasoning.

We start by experimenting with the five features proposed by the MeshCNN framework that describe the geometry, and quickly realize that, as can be expected, not all features are created equal. Through these experiments, along with investigations of explicit

*Both authors contributed equally to the paper

Authors' addresses: Amir Barda, Tel-Aviv University, Israel, amirbarda@mail.tau.ac.il; Yotam Erel, Tel-Aviv University, Israel, erelyotam@gmail.com; Amit H. Bermano, Tel-Aviv University, Israel, amberman@tauex.tau.ac.il.

and differential coordinates, we submit that features fed to the network should have two key ingredients: they should be rotation and translation invariant, and should be reconstructable, i.e., should provide enough geometric description that at least allows for the reconstruction of the original geometry. As demonstration, we propose the minimal yet elegantly fitting representation of the first and second fundamental forms [Keenan 2015] — namely the length and dihedral angle of each edge. As we demonstrate in Section 5, these features yield higher accuracy for most of the popular mesh analysis tasks. In addition, since the fundamental forms naturally lend themselves better to geometric functions, they enable learning reconstructable representations, or in other words, they open the door to better mesh generation capabilities. We demonstrate this claim through de-noising experiments, where the network is asked to reconstruct a noisy input mesh through a vanilla auto-encoder paradigm.

We argue that since the network should understand 3D geometric shapes, equipping it with geometry-driven processes acts as powerful guidance. Under this light, we inspect the pooling operation as well. Observing that the pooling layer's decision making process does not follow geometric reason, we adapt its scheme to achieve consistent performance improvement across all our experiments.

Our main contributions can be summarized as follows:

- Introduction of the first and second fundamental forms as a representation fitting for learning meshes.
- Analysis of the properties this representation holds that lend themselves to learning shape analysis tasks, as well as generative ones.
- Enhancing the pooling layer to update the mesh after each edge collapse, thus better preserving the accumulated information throughout the process.

2 RELATED WORK

The success of deep neural networks in the realm of 2D image processing and machine vision has inspired many to investigate applying these techniques to various tasks over 3D geometry. The differences between the various approaches primarily stem from the geometric representations used by each one. Next we briefly introduce the works done along these axes. For an in depth survey of recent work in geometric deep learning, we refer the reader to Gezawa et al. [2020].

Image-Based Representations. Perhaps the most natural way to apply established 2D methods to 3D data is to render the geometry into a set of 2D images, and learn from them. The first works rendered the geometry from several directions and stacked the results together for a network to learn [Bai et al. 2016; Boulch et al. 2017; Feng et al. 2018; He et al. 2018; Kanezaki et al. 2018; Qi et al. 2016; Yavartanoo et al. 2018]. Later approaches added other modalities, such as depth maps or planar parameterization techniques in order to add more geometric meaning to the 2D images [Ezuz et al. 2017; Gomez-Donoso et al. 2017; Haim et al. 2019; Maron et al. 2017; Sarkar et al. 2018; Zanuttigh and Minto 2017]. While still an active field of research [Han et al. 2019; Wang et al. 2019c], we argue that 2D images are inherently unfit to represent 3D geometry. Their regular sampling rate does not represent different frequencies well, e.g. gaps or sharp corners. In addition, proper representation of a geometry typically requires several images, which burdens the process in terms of computational load and memory footprint.

Volumetric Representation. Another natural approach is to extend the established 2D convolutions neural networks employ so well to 3D ones. This typically means equally dividing the space into a 3D grid, or a *voxel* grid, and employing 3D convolutions over the regularly sample data [Brock et al. 2016; Çiçek et al. 2016; Park et al. 2019; Sedaghat et al. 2016; Wang et al. 2019a; Wu et al. 2015]. While this approach is a better fit for 3D geometric, as it is inherently structured to reflect 3D relations, it still suffers from the curse of dimensionality. An image of 4K resolution (i.e., roughly eight million pixels) bears the same amount of elements as a $200 \times 200 \times 200$ voxel grid. Hence, the complexity of voxel grids inhibit their practicality. Some works have started exploiting tree structures to significantly reduce memory footprint through adaptive cell sizes [Roynard et al. 2018; Wang et al. 2017, 2018]. These works essentially seek to balance the networks' need for regular sampling with the data's typical irregularity. However, as of now no such method has presented a practical, scalable and highly accurate model the is successful in finding this balance.

Point clouds. Point clouds are the currently leading representation in terms of accuracy of the models that employ them. Points clouds are sets of points (and optionally normals) that reside on the objects boundary. Their dense sampling rate lends itself naturally to the neural networks' preference for regular data, while the fact that they only reside on the surface provides sparsity. The challenge pointclouds hold for neural networks is that they are unordered, implying that the learned model should be permutation invariant. PointNet [Qi et al. 2017a] was the first to introduce a permutation invariant model for geometric learning of pointclouds. Since then this core concept has been utilized and developed further many times, employed for tasks such as classification and segmentation [Atzmon et al. 2018; Ben-Shabat et al. 2018; Engelmann et al. 2020; Guerrero et al. 2018; Li et al. 2018; Liu et al. 2019a,b; Qi et al. 2017b; Tchapmi et al. 2017; Thomas et al. 2019; Wang et al. 2019b], but also for generation [Li et al. 2019; Shu et al. 2019; Wang et al. 2019b]. Even though the state-of-the-art in shape analysis and generation lies with this representation, we argue that it still holds inherent drawbacks. First, even though its dense sampling rate is orders of magnitude more efficient than voxel grids, it is still wasteful. More often than not, shapes are composed of regions with much variation and features, but also with large regions that are flat or are otherwise less informative. A geometry-aware adaptive sampling rate is something current networks are not able to handle. Secondly, pointclouds do not describe connectivity. Typically in pointclouds, connectivity is implicitly assume through local proximity to other points. While this prior is usually accurate, it fails to describe well highly curved regions, or different parts of the shape that are very close to each other. Meshes, on the other hand, offer solutions to these two drawbacks.

Graph Representations. A natural approach to learning on 3D triangular meshes is to view them as graphs, which allows employing variants of GCNs (Graph Convolutional Networks) [Henaff et al.

2015; Kipf and Welling 2017; Mo et al. 2019; Monti et al. 2018; Simonovsky and Komodakis 2017]. These works define kernels that operate typically over each vertex and its 1-ring neighborhood, and compensate their weights according to vertex valence and edge lengths. However, graphs alone consider only the vertices and edges of the mesh for the learning process. They do not consider perhaps the most important aspect of the mesh representation, which are the faces. Hence, graph-based methods that better accommodate for the connectivity information that the mesh has to offer typically show preferable performance to purely graph-based ones.

Mesh Representations. Convolutions that exploit mesh connectivity structures were first introduced by Masci et al. [2015]. Since then, several approaches addressed the irregularity of the mesh structure and sampling rate, proposing ideas such as sampling the neighborhood of each vertex uniformly, or achieving regularity through spectral decomposition [Boscaini et al. 2016; Gong et al. 2019; Lim et al. 2018; Poulenard and Ovsjanikov 2018; Schult et al. 2020; Verma et al. 2018; Zhou et al. 2020]. The seminal work of MeshCNN [Hanocka et al. 2019], which we base our investigation on, proposed a novel convolution and pooling operations that exploited the mesh structure, yielding a rotation invariant, simple, and highly accurate shape analysis architecture. Using the ideas of this work as a backbone, several publications have applied this engine to downstream applications, such as mesh super-resolution and geometric texture synthesis [Hanocka et al. 2020; Liu et al. 2020].

Being innovative, MeshCNN is a proof-of-concept, and is not thoroughly optimized in terms of architecture and features. For example, follow-up work has suggested adding attention layers to it [Milano et al. 2020]. While we compare our accuracy to the latter work, we argue it is orthogonal to the claim of this paper, as one could also add additional layers and teaks to the system we propose. Furthermore, unlike GCNs, MeshCNN and its followup cannot currently perform tasks which require recovering the mesh geometry at the output - such as de-noising. Another recently proposed novel approach suggests accumulating geometric features over a random walk performed over a mesh using an RNN paradigm [Lahav and Tal 2020]. This approach poses several advantages, such as low memory footprint, accuracy and robustness. However, besides the burden of training the notoriously hard to train RNN architecture, this approach can not be used for generative downstream tasks, and because the machinery described works for any type of graph, it is less geometrically motivated and fails to exploit strong priors existing in triangle meshes like regularity of the edge or face neighborhoods. Hence again, we compare our performance to this work, but again make the same claim of orthogonality. Lastly, we note the difference between this line of work, which we belong to, that seeks to understand the structure and connectivity of the mesh, to the approaches that deform the same mesh throughout the process (i.e., networks that learn to manipulate the geometry of a constant connectivity) [Gupta and Chandraker 2020; Yuan et al. 2020; Zhou et al. 2020]. Most real work application cannot be efficiently described using a constant connectivity.

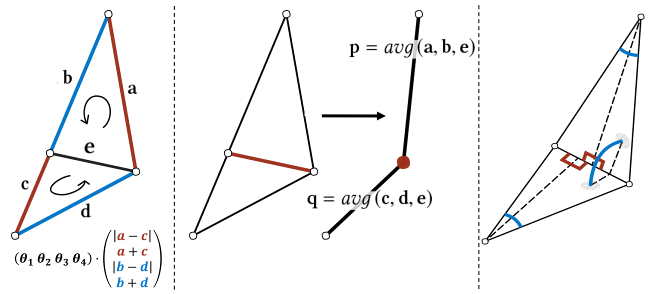


Fig. 2. MeshCNN definitions (figures from Hanocka et al. [2019]). Left: a convolution centered at edge e is defined as formulated below. Note the operator is symmetric with regards to the blue and the red edges. Middle: edge pooling operation, the five edges of the 1-ring become two, with averaged feature values. Right: the input representation. Each edges is described through 5 numbers. The dihedral angle (blue), the two opposite angles (blue), and the two edge length-to-triangle height ratios (red).

3 PRELIMINARIES

3.1 MeshCNN Review

For the sake of completeness, we give a brief overview of the MeshCNN pooling and convolution layers[Hanocka et al. 2019]. For full details, we refer the reader to the original paper.

3.1.1 Convolutions. MeshCNN is based on the observation that every edge in a watertight 2-manifold mesh has a fixed amount of incident faces. Therefore, unlike previous art, MeshCNN defines the convolution on the edges of the mesh. For every edge e, the convolution is performed on its 1-ring neighborhood (i.e., on all edges of the incident triangles), in counter clockwise order (see Figure 2, left). This creates an ambiguity as the order can be either (a, b, c, d) or (c, d, a, b). To resolve this, The convolution operator is defined to be the order invariant functions:

$$\begin{gathered} e \cdot \theta_0 + \sum_{j=1}^{4} \theta_j \cdot e^j, \\ (e^1, e^2, e^3, e^4) = (|a-c|, a+c, |b-d|, b+d), \end{gathered} \tag{1}$$

where θ_i are the learned weights.

3.1.2 Pooling. One of the distinguishing features of MeshCNN is its pooling layer, which collapses edges according to their feature norms. A strong feature norm signals an area with important information for the next layer, so the pooling layer chooses the weakest ones and removes them according to a simple mesh simplification scheme [Hoppe 1997]. After all the edges to-be-pooled are chosen (i.e. the target number of edges is reached), the pooling layer updates to features on the edges by averaging the features in the collapsed 1-ring, as shown in Figure 2, middle.

3.1.3 Representation. As portrayed in Figure 2, right, each edge is represented by five features:

- The angle between the normal vectors of its two incident faces.
- The two angles opposite to the edge.

- The two ratios between the edge length and the corresponding triangle's height.

These features are invariant to rotation, translation and uniform scaling (although scaling invariance can always be achieved through bounding box normalization).

3.2 Fundamental Forms

Intuitively, the first fundamental form I measures lengths on the surface induced by the euclidean metric, and the second fundamental form II measures the rate of change of the normal field. The two fundamental forms are established measures, known to be coordinate invariant and to fully characterize the shape up to translation and rotation. In the continuous case, for a surface M with a parameterization (u^1, u^2), if there is a local embedding x in R^3, I can be induced by $I_{\alpha\beta} = (x_{,\alpha}, x_{,\beta})$ and II is induced by $II_{\alpha,\beta} = (x_{,\alpha\beta}, N)$, where x is a position on the surface, $\alpha, \beta = 1, 2$ are indices of the parameterization, and an index after comma denotes a partial derivative with the respect to the index. In the discrete case of a triangular mesh, it turns out that the fundamental forms conveniently reside on mesh edges, are simply the edge length, and the edge dihedral angles respectively [Wang et al. 2012].

4 METHOD

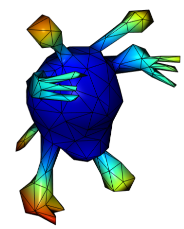
The premise of this report is that since the network is expected to learn geometry, imposing upon it geometrically consistent operations would offer powerful guidance for the training process. We investigate and apply this principle to two points in the MeshCNN framework — its pooling operation and its geometric representation. In the following, we describe the investigation, the conclusion, and its application for each of the two aspects.

4.1 Representing Geometry

The investigation begins with examining the features proposed by the original work (see Section 3.1). As a test-bed we inspect the task of mesh classification, using the SHREC dataset [Lian et al. 2011]. The performance of various combinations of the proposed features are listed in Table 1. The results clearly indicate that the features are not equal. The opposite angle features yield the best accuracy, while the other two yield lower scores. In contrast, we that the dihedral angle is more effective in conjunction with the other two features, while they do not contribute to one another. Following the main premise of geometric reasoning, we observe that the effects on accuracy corresponds to geometry reconstructability, i.e, features that allow recovering the mesh geometry will perform better. For this reason, the opposite angles and length ratios do not contribute to each other; since they accordingly do not contribute much to reconstructing the geometry. We hence postulate that reconstructability is a good measure of the potential information the features contribute to the learning process. We argue that features that provide full **reconstructability** also provide useful information to the network. Indeed, inspecting the behavior of the originally proposed features reinforces the suspicion that these non-reconstructable features convey weaker signals to the learner. Attacking this from another angle, we visually inspect the spatial distribution of the information conveyed by the originally proposed features. Figure 4 top, portrays the L_2 norms of the five features on each edge. No observable pattern can be seen in regions that are expected to be unique or indicative. This again cues that these features are less informative for the network, in correlation to reconstructability.

Table 1. Feature Combinations. The accuracy results on SHREC test set using various features.

Method	Accuracy
Dihedral angle	85.0%
Length ratios	85.0%
Opposite angles	90.83%
Opposite + ratios	91.67%
Dihedral + ratios	96.6%
Dihedral + Opposite	95.0%
MeshCNN	**98.6%**



We therefore propose using more traditional reconstructable features, such as the coordinates themselves, or the widely established Laplacian coordinates. Laplacian coordinates are vectors which give the difference between the vertex and its neighbors' (possibly weighted) average, and are employed in countless geometry processing operations [Sorkine 2005]. By definition, the norm of these coordinates is directly linked to the local rate of change (See heat-map in inset). The results of these experiments over the same aforementioned setup are depicted in Table 2. In contradiction to the aforementioned suspicion, these reconstructable features do not yield better performance. Our insight is that the reason is rotation and (or) translation invariance. These transformations are natural symmetries for the task, since a rotated chair is still as it perceived by us. Indeed, as can be seen, when adding random rotations to the test and training process, the performance of these features is lowered. Even more so, we see that when converting the explicit coordinates to rotation invariant ones, simply by taking their dot products and average of norms instead of a simple average, accuracy is dramatically improved. This is even though the latter operations reduce how descriptive the representation is. This strongly indicates that **rotation and translation invariance** is a necessity.

To validate this claim, we propose the slimmest representation that is both rotation and translation invariant, and is reconstructable — the first and second fundamental forms. Indeed, when examining this representation's spatial behavior, we can observe a strong correlation between it and salient and distinctive regions in the mesh (see Figure 4 bottom). As we demonstrate in Section 5, these two numbers alone induce higher quality learning than the alternatives. In addition, we also demonstrate that since this representation is reconstructable, it naturally lends itself to tasks of reconstruction or generation.

Table 2. Performance of various reconstruction-abling features on SHREC.

Feature Type	Accuracy
xyz coordinates	76.73%
xyz w/ random rotations	69.78%
xyz dot and norms	82.85%
Laplacian coordinates	54.44%
MeshCNN	**98.6%**

We note that adding more features or architectural elements to framework would probably improve the learning even further, however this investigation is out of the scope of this paper, and is left for future work.

4.2 Pooling

The originally proposed MeshCNN pooling layer chooses the N edges with the lowest norm values to collapse, where N is a predefined hyper-parameter. This imposes the popular assumption that informative regions should bear stronger activations. This also reinforces the feature selection process described in Section 4.1 and Figure 4. The problem with this approach, however, is in the selection mechanism, which does not follow geometric reasoning. As opposed to traditional 2D pooling, where pixels are simply omitted. The pooling operation is designed maintain geometric integrity, by sifting information from the collapsing edge into its neighborhood. However, choosing the N edges to collapse probably implies collapsing many edges in the same area. This contradicts the geometric meaning of the pooling operation. Presumably, after enough edges of a region are collapsed, the aggregated information reaches a certain threshold and should not be collapsed again (see Figure 3). Hence we propose to update the features after each edge collapse. Since the collapse operation is local, so is the required update after each iteration, making the computational load of the update operation to be negligible. As we demonstrate in Section 5, this lightweight step consistently improves accuracy across all our experiments. In addition, when qualitatively following the collapsed edges throughout the network, a clearer representation of the shape can be observed using the updated pooling scheme compared to the original one, as depicted in Figure 6. This indicates the model is more attentive to the entire shape during classification, suggesting robustness to disturbances.

5 RESULTS

We perform several experiments to validate our results. We employ our system for classification on the SHREC [Lian et al. 2011] and Cube engraving[Hanocka et al. 2019] datasets. We then perform segmentation on the Human Segmentation dataset prepared in an identical fashion as Hanocka et al. [2019]. Importantly we use identical network architectures that were used by Hanocka et al. [2019], to isolate the effects of the discussed aspects alone (i.e., the representation and pooling scheme). Additionally, we compare our results to state-of-the-art works [Lahav and Tal 2020; Milano et al. 2020; Smirnov and Solomon 2021] when applicable. Finally, we perform ablation studies, and demonstrate the ability to perform a generative task of mesh de-noising.

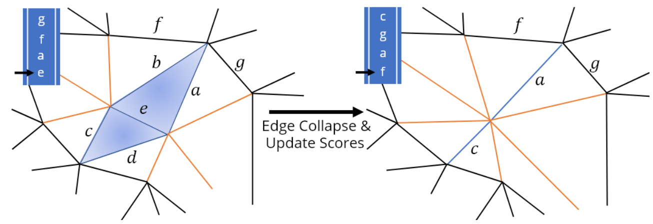


Fig. 3. The enhanced pooling layer. Left: an edge e with its 1-ring neighborhood $\{a, b, c, d\}$ was selected to be collapsed due to its low features norm. Notice that edge a had the second lowest score prior to collapse. Orange edges were marked to emphasize the fact they will also be affected by the collapse geometrically, but we do not take this into account when computing the new scores. Right: After collapsing e, we calculate the new scores for a and c (by averaging upon the old a, b and e for a, and the old c, d and e for c) suddenly a doesn't have the lowest score, and instead some other edge f will be selected next. This is on contrary to the original layer which would have selected a.

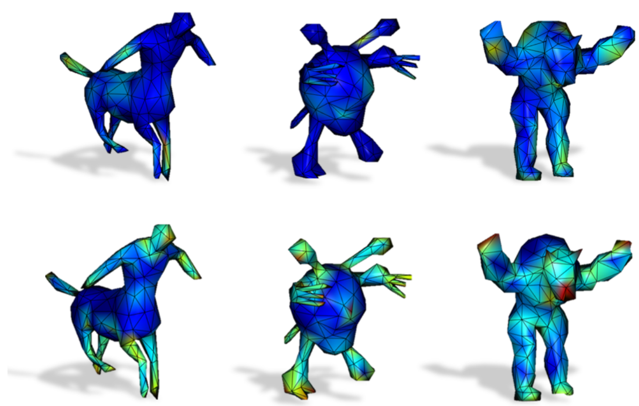


Fig. 4. Feature norm comparison between the original 5 edge features in [Hanocka et al. 2019] (top) and our 2 proposed edge features (fundamental forms, bottom) on several models from the SHREC16 dataset. All feature channels are normalized by subtracting the mean and dividing by the standard deviation. MeshCNN's pooling is performed according the feature norms, with edges with lower norms pooled first. The heat maps indicate that our features provide better indication of "areas of interest" such as hands, legs and heads in the shown meshes. This provides a better starting point for the network and improves performance overall, as we observed empirically.

5.1 SHREC classification

We first test our performance on the SHREC dataset [Lian et al. 2011], and similarly to [Hanocka et al. 2019] split the dataset into 16/4 training/test examples per class (Split 16) and 10/10 split as well (Split 10). The total amount of samples is 600. We train using the exact setup and hyper parameters proposed by Hanocka et al. [2019], only replacing the features and the pooling layer with ours

Table 3. Test set accuracy for various methods and datasets. Human segmentation results are soft edge accuracies.

Method	SHREC Split 16	SHREC Split 10	Cubes Engraving	Human Segmentation
Ours	**100%**	**99.33%**	**99.24%**	93.23%
MeshCNN	98.6%	91.0%	92.16%	92.3%
MeshWalker	98.6%	97.1%	98.6%	**94.8%**
HodgeNet	99.17%	94.67%	-	-
PDMesh	99.7%	99.1%	94.39%	91.11%

for fairness. See Table 3 for summary of results. It is worthy to note that the SHREC classification dataset is relatively well aligned globally and ablation studies performed by [Hanocka et al. 2019] have shown that using naive Cartesian coordinates as features yield quite good results. This would not be the case were the input meshes were randomly rotated. See section 5.6 for more information.

5.2 Cubes Engraving classification

This dataset contains 23 classes of shapes embossed into a random face of a cube, consisting of 4600 meshes. Although being somewhat contrived, the connectivity element of the 3d structure plays a much more important role for these sort of meshes (the engraved areas). Attempts to solve this task using a topology agnostic solution yield inferior results. It is clear from the results (Table 3) that using our features and pooling layer improves accuracy with respect to [Hanocka et al. 2019], to the extent that it surpasses all other state-of-the-art as well.

5.3 Human Segmentation

This dataset contains 370 models of humans for training from SCAPE [Anguelov et al. 2005], FAUST [Bogo et al. 2014], MIT [Vlasic et al. 2008] and Adobe Fuse [ado 2020], and the test set consists of 18 models from SHREC07 [Giorgi et al. 2007] humans dataset. The models are annotated with a per-face labels segmented using Kalogerakis et al. [2010]. The 3D meshes include humans in various poses, with the same topological properties: 752 vertices, 2252 edges and 1500 faces. Note The meshes are all manifolds with genus zero, with no self intersecting faces. See Table 3 for results. Note we use soft edge labels as in Hanocka et al. [2019] that take into account edge lengths. The results for Milano et al. [2020] were taken from their supplementary material.

5.4 Mesh De-noising

For mesh de-noising, we use the SHREC dataset. In addition, we employ the segmentation architecture of MeshCNN, only instead of outputting L channels that correspond to the L labels of the segmentation, we output the geometry for every edge. We compare using explicit XYZ coordinates as output to our reconstructable fundamental forms. Quantitative results for the test set are reported in Table 4. As can be seen, learning the fundamental forms is significantly simpler for the network, compared to trying to figure out the explicit coordinates. In fact, the explicit XYZ coordinates even fail to achieve the precision a simple identity reconstruction would have. This results continues reinforcing our rotation and translation invariance claim. In contrast, generating the Fundamental Form representation is significantly simpler for the training process to learn, yielding an error that is 10 times smaller than the identity. The proposed fundamental forms also bear merit over employing the representation proposed by MeshCNN.

Table 4. De-noising Results. We applied random Gaussian noise to all vertices with variance 0.1. FF stands for Fundamental Forms, MeshCNN features are described in 3.1 and XYZ are cartesian coordinates. Notice Identity entries indicate what would be the Average MSE between a noisy mesh and its clean counter part (serves as baseline).

Experiment	Average MSE
FF Identity	0.05
FF -> FF	0.0096
MeshCNN -> FF	0.012
XYZ Identity	0.01
FF -> XYZ	0.08
MeshCNN -> XYZ	0.082

5.5 Feature Validation and Comparisons

Of course, many choices exist for reconstructable features. In this work we choose the slimmest possible ones, to demonstrate their effectiveness. We compare this representation to a few popular alternatives.

Cartesian Coordinates. The most naive choice is to simply use the Cartesian midpoint of each edge as its features, namely, 3 input channels corresponding the x,y and z coordinate value respectively. This was attempted in the original meshCNN paper, and achieved a surprisingly good result of 91% test accuracy. Upon further examination, we have found this result is largely due to the alignment between meshes in the SHREC dataset. Randomly rotating the meshes during training lowers the training accuracy to 70%.

We have also experimented with a rotation invariant version of this representation - through taking the dot product of the two incident vertices, and the average of their norms. Significant improvement can been seen by this seemingly less informative representation, which strongly indicates the need for rotation invariance.

Laplace Coordinates. Laplace coordinates δ, are the result of multiplying the Laplacian matrix L, with the Cartesian coordinates V. Laplacian coordinates are vectors which give the difference between the vertex and its neighbors' (possibly weighted) average. Nevertheless the coordinates themselves are not rotation and translation invariant, which hinders performance on datasets without mesh alignment.

MeshCNN Features. Additionally, we try using several combinations the originally proposed features. The conclusions from these experiments are depicted in Section 4.1.

Test accuracy results on the SHREC16 dataset for each of these experiments is given in Tables 2 and 4.

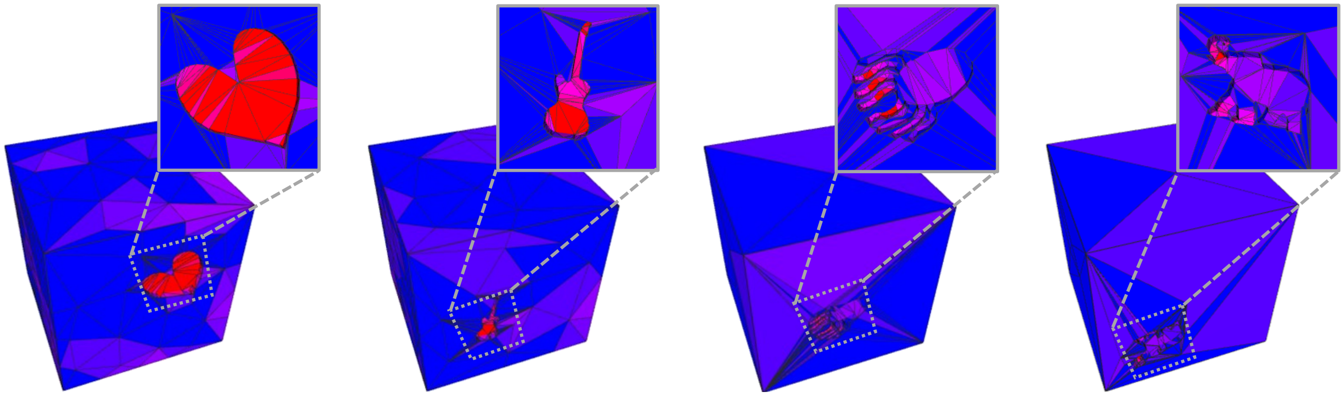


Fig. 5. Visualization of our network's shape understanding. During a forward pass on a test mesh from the cubes dataset, we measure when in the process was every edge collapsed. Colder (more blue) edges were collapsed early in the pipeline, while warmer (more red) regions were collapsed later, or not at all. As can be seen, the network learns to focus on distinctive features in the engravings (and not on the engraved cube itself) such as the octopus's legs and the elephant's trunk.

Table 5. Ablation Study

Method	SHREC Split 16	SHREC Split 10	Cubes Engraving	Human Seg.
MeshCNN	98.6%	91.0%	92.16%	92.3%
PL	**100%**	96.33%	96.5%	91.5%
FF	**100%**	**99.33%**	94.08%	92.49%
Ours (PL, FF)	**100%**	**99.33%**	**99.24%**	**93.23%**

5.6 Ablation Study

We perform an ablation study that verifies incremental improvement from Hanocka et al. [2019] by using our enhanced pooling layer (PL) and by using the fundamental forms as features (FF). See table 5 for detailed results.

6 DISCUSSION AND FUTURE WORK

We presented a representation and geometric reasoning to introduce better learning to the decorated MeshCNN framework [Hanocka et al. 2019]. Our contributions are in the form of a reconstructable representation and a better pooling mechanism. Another benefit of our work is enabling generative tasks to be carried out using the same backbone, exposing the field of geometric learning over meshes to a new frontier. While MeshCNN certainly performs better under the proposed paradigm, the pooling layer is relatively computationally demanding, and does not scale well to larger models. This is due to the sequential nature of selecting edges from the data structure sorted by the scores, whereas updating the scores of the neighbors has proven to have negligible difference in computational time. We do believe future work must focus on the optimization of the pooling layer, as models in larger scales are almost always represented by a mesh due to their inherent sparsity, which isn't fully exploited by our technique. Another important factor that distinguishes the task of shape understanding using our technique, is the enhanced explain-ability of the decision making made by the network enabled by geometrically motivated operations. For example, tracing the pooled edges back to their parent (Figure 5) to visualize the networks attention, or being able to add a reconstruction loss term to the training scheme since reconstruction is possible at the output. We believe that the convolution operation itself could also gain from better geometric reasoning, and leave this investigation for future work. Lastly, we argue that the reconstructable features could open up the door to a plethora of generative tasks that could be carried out by the MeshCNN backbone, and are anxious to see what this line of work could yield in the near future.

REFERENCES

2020. Adobe Fuse 3D Characters, Adobe. 2016. https://www.mixamo.com.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. *ACM Trans. Graph.* 24, 3 (2005), 408–416. https://doi.org/10.1145/1073204.1073207

Matan Atzmon, Haggai Maron, and Yaron Lipman. 2018. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091* (2018).

Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. 2016. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5023–5032.

Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 2018. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3145–3152.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Columbus, Ohio, USA, 3794 –3801.

Davide Boscaini, Jonathan Masci, Emanuele Rodoià, and Michael Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 3197–3205.

Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. 2017. Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. *3DOR* 2 (2017), 7.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2016. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236* (2016).

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42. https://doi.org/10.1109/MSP.2017.2693418

Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 2016. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and

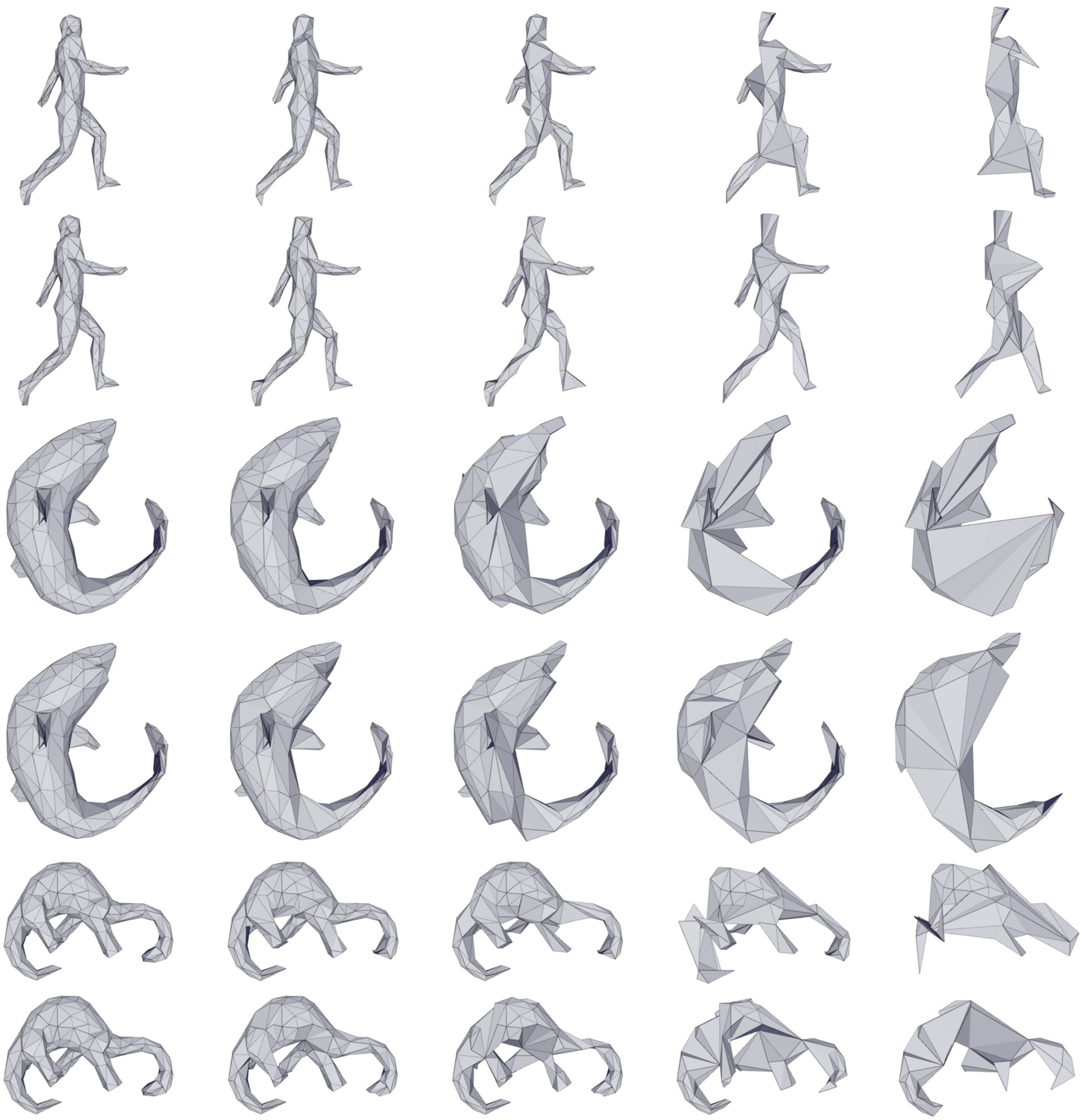


Fig. 6. Qualitative evaluation of pooling the edges. From left to right: an original mesh is being pooled into lower spatial dimensions. Top rows: MeshCNN pooling, Bottom rows: our pooling. By using the trained networks and passing test meshes through them, we can visualize how the pooling layers perform by applying the new topology after each layer to the geometry of the original mesh. By updating the scores of edges during the pooling operation, the global shape is preserved better.

William Wells (Eds.). Springer International Publishing, Cham, 424–432.
Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 2020. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9031–9040.
Danielle Ezuz, Justin Solomon, Vladimir G. Kim, and Mirela Ben-Chen. 2017. GWCNN: A Metric Alignment Layer for Deep Shape Analysis. *Comput. Graph. Forum* 36, 5 (Aug. 2017), 49–57. https://doi.org/10.1111/cgf.13244
Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. 2018. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 264–272.
Abubakar Sulaiman Gezawa, Yan Zhang, Qicong Wang, and Lei Yunqi. 2020. A review on deep learning approaches for 3D data representations in retrieval and classifications. *IEEE Access* 8 (2020), 57566–57593.
Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. 2007. Shape retrieval contest 2007: Watertight models track. *SHREC competition* 8, 7 (2007).
Francisco Gomez-Donoso, Alberto Garcia-Garcia, J Garcia-Rodriguez, Sergio Orts-Escolano, and Miguel Cazorla. 2017. Lonchanet: A sliced-based cnn architecture for real-time 3d object recognition. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 412–418.
Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. 2019. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 0–0.
Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. 2018. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 75–85.
Kunal Gupta and Manmohan Chandraker. 2020. Neural Mesh Flow: 3D Manifold Mesh Generation via Diffeomorphic Flows. arXiv:2007.10973 [cs.CV]
Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. 2019. Surface networks via general covers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 632–641.
Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and CL Philip Chen. 2019. 3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation. *IEEE Transactions on Image Processing* 28, 8 (2019), 3986–3999.
Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 90.
Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: a self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084* (2020).
Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. 2018. Triplet-center loss for multi-view 3d object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1945–1954.
Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).
Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep Geometric Texture Synthesis. *ACM Trans. Graph.* 39, 4, Article 108 (July 2020), 11 pages. https://doi.org/10.1145/3386569.3392471
Hugues Hoppe. 1997. View-Dependent Refinement of Progressive Meshes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 189–198. https://doi.org/10.1145/258734.258843
Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. 2010. Learning 3D Mesh Segmentation and Labeling. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 102, 12 pages. https://doi.org/10.1145/1833349.1778839
Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. 2018. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5010–5019.
Crane Keenan. 2015. *Discrete Diffenetial Calculus: An Applied Introduction.*
Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]
Alon Lahav and Ayellet Tal. 2020. MeshWalker: Deep Mesh Understanding by Random Walks. *ACM Trans. Graph.* 39, 6, Article 263 (Nov. 2020), 13 pages. https://doi.org/10.1145/3414685.3417806
Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2019. PU-GAN: A Point Cloud Upsampling Adversarial Network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems* 31 (2018), 820–830.
Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. 2011. SHREC'11 Track: Shape Retrieval on Non-Rigid 3D Watertight Meshes. In *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval* (Llandudno, UK) *(3DOR '11)*. Eurographics Association, Goslar, DEU, 79–88.
Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. 2018. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 0–0.
Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. 2020. Neural Subdivision. *ACM Trans. Graph.* 39, 4, Article 124 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392418
Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. 2019a. DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. 2019b. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8895–8904.
Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4 (2017), 71–1.
Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*. 37–45.
Daniel Maturana and Sebastian Scherer. 2015. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 922–928. https://doi.org/10.1109/IROS.2015.7353481
Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-Dual Mesh Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 952–963. https://proceedings.neurips.cc/paper/2020/file/0a656cc19f3f5b41530182a9e03982a4-Paper.pdf
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.
Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J. Guibas. 2019. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. arXiv:1908.00575 [cs.GR]
Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M. Bronstein. 2018. Dual-Primal Graph Convolutional Networks. arXiv:1806.00770 [cs.LG]
Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 165–174. https://doi.org/10.1109/CVPR.2019.00025
Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.
Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.
Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5648–5656.
Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5105–5114.
Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. 2018. Classification of point cloud scenes with multiscale voxel deep network. *arXiv preprint arXiv:1804.03583* (2018).
Kripasindhu Sarkar, Basavaraj Hampiholi, Kiran Varanasi, and Didier Stricker. 2018. Learning 3d shapes as multi-layered height-maps using 2d convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 71–86.
Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. 2020. DualConvMesh-Net: Joint Geodesic and Euclidean Convolutions on 3D Meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8612–8622.
Nima Sedaghat, Mohammadreza Zolfaghari, Ehsan Amiri, and Thomas Brox. 2016. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351* (2016).
Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 2019. 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. arXiv:1704.02901 [cs.CV]

Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: Learning Spectral Geometry on Triangle Meshes. arXiv:2104.12826 [cs.GR]

Olga Sorkine. 2005. Laplacian Mesh Processing. In *Eurographics 2005 - State of the Art Reports*, Yiorgos Chrysanthou and Marcus Magnor (Eds.). The Eurographics Association. https://doi.org/10.2312/egst.20051044

Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. 2017. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*. IEEE, 537–547.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6411–6420.

Nitika Verma, Edmond Boyer, and Jakob Verbeek. 2018. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2598–2606.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated Mesh Animation from Multi-View Silhouettes. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–9. https://doi.org/10.1145/1360612.1360696

Cheng Wang, Ming Cheng, Ferdous Sohel, Mohammed Bennamoun, and Jonathan Li. 2019a. NormalNet: A voxel-based CNN for 3D object classification and retrieval. *Neurocomputing* 323 (2019), 139–147.

Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. 2019c. Dominant set clustering and pooling for multi-view 3d object recognition. *arXiv preprint arXiv:1906.01592* (2019).

Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. 2019b. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10296–10305.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4, Article 72 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073608

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018. Adaptive O-CNN: A Patch-Based Deep Representation of 3D Shapes. *ACM Trans. Graph.* 37, 6, Article 217 (Dec. 2018), 11 pages. https://doi.org/10.1145/3272127.3275050

Yingxu Wang, B. Liu, and Yiying Tong. 2012. Linear Surface Reconstruction from Discrete Fundamental Forms on Triangle Meshes. *Computer Graphics Forum* 31 (12 2012), 2277–2287. https://doi.org/10.1111/j.1467-8659.2012.03153.x

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Mohsen Yavartanoo, Eu Young Kim, and Kyoung Mu Lee. 2018. Spnet: Deep 3d object classification and retrieval using stereographic projection. In *Asian Conference on Computer Vision*. Springer, 691–706.

Yu-Jie Yuan, Yu-Kun Lai, Jie Yang, Qi Duan, Hongbo Fu, and Lin Gao. 2020. Mesh variational autoencoders with edge contraction pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 274–275.

Pietro Zanuttigh and Ludovico Minto. 2017. Deep learning for 3d shape classification from multiple depth maps. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3615–3619.

Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. 2020. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *arXiv preprint arXiv:2006.04325* (2020).