TOWARDS MESH-BASED DEEP LEARNING FOR SEMANTIC SEGMENTATION IN PHOTOGRAMMETRY

Manuel Knott[1,2*], Rick Groenendijk[1]

[1] University of Amsterdam, The Netherlands; [2] Cloudflight Germany GmbH, Germany
manuel.knott@protonmail.com, r.w.groenendijk@uva.nl

Commission II, WG II/4

KEY WORDS: Semantic Segmentation, Textured Meshes, 3D Computer Vision, Deep Learning, MeshCNN

ABSTRACT:

This research is the first to apply MeshCNN – a deep learning model that is specifically designed for 3D triangular meshes – in the photogrammetry domain. We highlight the challenges that arise when applying a mesh-based deep learning model to a photogrammetric mesh, especially w.r.t. data set properties. We provide solutions on how to prepare a remotely sensed mesh for a machine learning task. The most notable pre-processing step proposed is a novel application of the Breadth-First Search algorithm for chunking a large mesh into computable pieces. Furthermore, this work extends MeshCNN such that photometric features based on the mesh texture are considered in addition to the geometric information. Experiments show that including color information improves the predictive performance of the model by a large margin. Besides, experimental results indicate that segmentation performance could be advanced substantially with the introduction of a high-quality benchmark for semantic segmentation on meshes.

1. INTRODUCTION

Semantic segmentation is one of the fundamental problems in computer vision; it is extensively researched both in two-dimensional images and three-dimensional representations such as voxel grids, point clouds, or mesh grids. Much like with 2D image understanding, 3D understanding has greatly benefited from the recent advances in machine learning (Griffiths and Boehm, 2019). One area of application for 3D scene understanding is semantic segmentation in topographic landscape models generated by remote sensing technologies such as Airborne Laser Scanning (ALS) or Digital Image Matching (DIM). In the context of aerial imagery, most publications focused on the segmentation of point clouds or their voxelized representations, respectively. These are obvious choices since those data representations – especially voxel grids – enable the adaption of approaches proven to be effective on 2D pixel data. However, in the field of remote sensing and photogrammetry, triangular textured meshes gradually replace point clouds as a final user product (Laupheimer et al., 2020a,b).

From a machine learning perspective, meshes have valuable properties. For one, they can represent data more efficiently than point clouds or voxel grids: Few data points can represent large, flat areas while at the same time, detailed areas can be represented by a sufficient number of elements. Furthermore, meshes can distinguish different entities of an object or scene that are close to each other since they allow the geodesic separation of objects, despite their proximity in the euclidean space. At the same time, meshes require different encoding to use in deep learning methods, since their elements do neither have a deterministic (such as voxel grids) nor a stochastic (such as many types of point clouds generated from aerial scans) uniform distribution in the Cartesian space. Therefore, graph-based machine learning approaches are preferable, as they are better suited to the mesh representation. (Hanocka et al., 2019)

* Corresponding author

In recent years, there have been publications on deep learning approaches that aim to directly encode and process meshes without omitting their particular properties, such as connectivity. However, in a predominant number of studies, the evaluated data are fully synthetic and generated using modeling software. On the other hand, meshes that are generated by photogrammetry could include topological errors (such as zero-length edges or non-watertight surfaces); could tend to a uniform distribution of elements; or have a high degree of detail, such that a substantial amount of computing power is required.

This research is the first one to apply a deep learning framework that is specially designed for 3D meshes as input – MeshCNN (Hanocka et al., 2019) – to perform semantic segmentation of a 3D textured mesh of an urban scene. We address the compatibility of meshes generated by photogrammetry with MeshCNN and, thereby, highlight the impact of the mesh properties on the model's predictive performance. These results could be used by other researchers to improve their meshing algorithms such that the pre-processed meshes are better suited for machine learning methods. From another perspective, we also discuss the model performance of MeshCNN on a data set that is substantially different from those which are used for benchmarking the method in the original work of Hanocka et al. (2019): The photogrammetry domain requires the analysis of complete scene chunks rather than synthesized objects. Meshes generated from real-world scenes contain a significantly higher amount of noise compared to manually designed geometries. Moreover, the current version of MeshCNN is designed for learning purely geometric representations and does not include any kind of texture or color information. The following research question is addressed:

Research Question: *How can mesh-based deep learning be applied to semantic segmentation of remotely-sensed textured 3D meshes of urban scenes?*

The contributions of this work are threefold:

- An evaluation of the key constraints of processing remote-sensed 3D meshes is performed.
- Practically, a novel approach to divide and reassemble a topographic mesh into computable subsets, that can be processed by a mesh-based deep learning model, is proposed.
- Experiments show that photometric information of textures, incorporated as features for the input of a mesh-based deep learning method, improves the quality of semantic segmentation.

2. RELATED WORK

3D computer vision is a rapidly emerging field of research but there are comparably few papers in the area of photogrammetry treating the classification and segmentation of textured meshes. Section 2.1 provides an overview of existing approaches of semantic segmentation of landscape meshes via machine learning. Section 2.2 summarizes MeshCNN and how a mesh geometry is encoded into features.

2.1 Semantic Segmentation of Landscape Meshes

Semantic segmentation is a central research topic in photogrammetry. Early publications propose probabilistic methods in which the segmentation is jointly done alongside 3D reconstruction for either voxel (Häne et al., 2017; Bláha et al., 2016) or mesh (Cabezas et al., 2015) representations. In recent years, there has been a clear trend of using deep learning approaches for semantic segmentation while it is most of the time decoupled from the reconstruction task (Griffiths and Boehm, 2019). While particular studies apply the segmentation directly on point clouds (Winiwarter et al., 2019; Zhao et al., 2018), others use a voxelized representation (Hackel et al., 2016; Huang and You, 2016; Tchapmi et al., 2018; Schmohl and Sörgel, 2019). In most cases, the underlying deep learning models are variations of Convolutional Neural Networks (CNNs). There are far fewer publications that investigate the segmentation of 3D meshes in remote sensing; some studies present pipelines that result in semantic meshes, but the actual segmentation task is performed on a preceding data representation (e.g. Leotta et al., 2019).

Rouhani et al. (2017) presented an approach of semantic segmentation of 3D textured meshes of urban scenes based on a Random Forest classifier and Markov Random Fields. They combine three geometrical features (elevation, planarity, and verticality) and three photometric features (average color, standard deviation, and color distribution in the HSV color space) from pre-clustered mesh faces. It is the first paper, to the best of our knowledge, to combine geometric and photometric properties of a textured mesh.

Tutzauer et al. (2019) present a deep learning approach for mesh segmentation, using a 1D CNN for the segmentation of meshes. Classes are assigned per face and radiometric and geometric features are calculated for each face and corresponding vertices. However, since both kinds of features are associated with the Center of Gravity (COG) of the face, they essentially create a point cloud. Laupheimer et al. (2020a,b) apply a similar approach. They use the same method for creating a COG cloud but do not manually add contextual features. Instead, they apply deep learning frameworks already capable of handling point clouds (i.e., "Pointnet++" (Qi et al., 2017)). The work of Laupheimer et al. (2020a,b) is used as a baseline in the current research. While their approach is robust against inherently mesh-based errors such as disconnected or intersected faces, it omits the geodesic relationship between mesh elements: Point-based methods do not exploit the potential information triangular meshes hold.

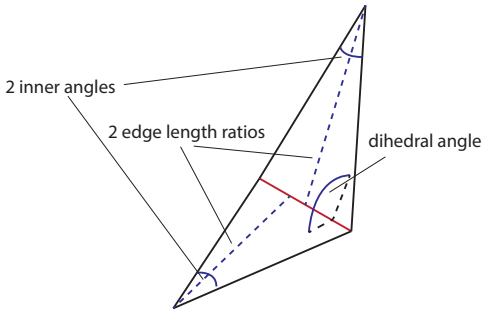


Figure 1. The five geometrical edge features for each face-pair used by MeshCNN (Hanocka et al., 2019)

2.2 MeshCNN

Fundamental research in the area of 3D computer vision investigates how a triangular mesh can be processed by a deep learning model (Masci et al., 2015; Verma et al., 2018; Tatarchenko et al., 2018). These approaches have in common that they address the irregularity of the mesh grid in the Euclidean space by introducing convolution operations to fit the data properties intrinsic to the mesh.

MeshCNN (Hanocka et al., 2019) is one of the latest and most mature deep learning architectures that is specifically designed to be applied to triangular meshes. The authors introduce mesh-specific convolution and pooling layers that are applied over the edges of a mesh. The five-dimensional feature vector for every edge is related to the 1-ring neighboring edges, depicted in Figure 1. It comprises the following geometrical features: the dihedral angle, two inner angles, and the two edge-length ratios (ratio of the base edge w.r.t to the height of the perpendicular face) for each face. Since all those features are relative, they are invariant to translation, rotation, and uniform scale (Hanocka et al., 2019). All shapes are required to be manifold meshes, possibly with boundary edges. Manifold meshes do not exhibit impossible geometries, such as zero-length edges, zero-area faces, or intersecting faces. Unlike other approaches, MeshCNN exploits the unique mesh properties, such as irregularity and non-uniformity.

Hanocka et al. (2019) perform multiple experiments on benchmark data sets for both classification and segmentation of objects. Experiments show that MeshCNN outperforms point- and voxel-based approaches. According to the authors, these results indicate that MeshCNN can exploit geometric properties of a mesh – the geodesic separation of elements and the adaptive non-uniform representation –, which could be of benefit during segmentation. However, the data sets used on those benchmarks are all artificially designed shapes; remotely sensed real-world scenes usually contain a significant amount of noise. Moreover, photometric features such as color or texture information were not incorporated into MeshCNN. Other publications indicate that the color information from high-resolution textures is an essential feature for remotely sensed meshes and might be more

relevant for the segmentation quality than per-point color information in point clouds (Laupheimer et al., 2020b).

3. METHODOLOGY

Even though there are benchmark data sets for both point cloud segmentation of outdoor scenes (e.g. Hackel et al., 2017) and mesh segmentation of general objects (e.g. Chen et al., 2009), to the best of our knowledge, there is no publicly available suitable and annotated textured mesh data set of outdoor scenes at the time this research is conducted. For this reason, an annotated point cloud data set – the ISPRS Vaihingen 3D data set (Niemeyer et al., 2014) – is used to generate a textured mesh and the associated classes. The mesh generation and the label generation are described in Section 3.1 and 3.2, respectively. Section 3.3 investigates how a large mesh can be effectively chunked into parts that can be processed by mesh-based deep learning approaches.

In this work, MeshCNN is applied to the remote sensing domain. The original version of MeshCNN only encodes geometric features. However, in this work, one of the goals is to exploit texture information also. Section 3.4 covers the inclusion of photometric features. Finally, Section 3.5 presents the complete experimental setup.

3.1 Data Set Preparation

The International Society for Photogrammetry and Remote Sensing (ISPRS) provides a data set of the town of Vaihingen for 3D semantic labeling of point clouds (Niemeyer et al., 2014). The data set comprises 2D Nadir images as well as a 3D point cloud that was generated by airborne laser scanning. Two independent regions of the point cloud are annotated as train and test set respectively with nine different semantic labels. As a first step, the 2D Nadir images were used to generate a 3D textured mesh via the commercial product SURE Aerial[1]. Since the regions of interest are too big to be output in a single file, chunking is applied by SURE. This chunked output causes multiple problems when the geometry is intended to be used as input for MeshCNN: The generated chunks can each have a significantly different number of edges as the area is uniformly partitioned in the Euclidean space rather than w.r.t. the number of mesh elements. In contrast, MeshCNN requires the input chunks to have an approximately equal number of edges. Therefore, before all else, the chunks need to be joined into a single big mesh. There is no automated way to cleanly connect the overlapping chunk borders. The result would be intersecting elements and, therefore, non-manifold geometries. Besides standard non-manifold geometries (namely zero-area faces and intersecting faces), the mesh is also not watertight, which means that holes exist. Even though MeshCNN can handle a certain number of holes, the training process is likely to crash during the pooling step when there are too many holes. This is because edges with less than four connected adjacent edges are ignored during pooling and, therefore, MeshCNN might run out of collapsible edges within a particular area.

For those reasons, as an additional step, a watertight error-free mesh is generated by applying Screened Poisson Surface Reconstruction (Kazhdan and Hoppe, 2013) on the vertices of the existing mesh via the open-source mesh processing tool MeshLab (Cignoni et al., 2008). The original texture is obtained by using the Texture Baking functionality of the open-source tool Blender[2]. Please note that the surface reconstruction was only applied due to the poor quality of the mesh: the raw SURE output contained many non-manifold artifacts. In an automated post-processing procedure, those often can only be cleaned by accepting holes in the mesh. If the mesh quality is sufficient (i.e., if there are no non-manifold geometries and only a few holes), the remeshing step should be skipped. In principle, Poisson Surface Reconstruction could directly be applied to a photogrammetric point cloud. Nevertheless, a reference mesh is needed to transfer the texture to the new mesh.

ISPRS Vaihingen class	Simplified class
Power line	(n/a)
Low vegetation	Low vegetation
Impervious surface	Ground/Other
Car	Ground/Other
Fence/Hedge	Low vegetation
Roof	Building
Facade	Building
Shrub	High vegetation
Tree	High vegetation

Table 1. ISPRS point cloud labels are mapped to mesh edge labels. Because of the mesh generation, a number of classes are collapsed to abstracted classes.

3.2 Label Generation

MeshCNN requires class annotations per edge. Since the generated mesh and the ALS point cloud of the ISPRS Vaihingen data set using the same reference coordinate system, edge annotations can be generated by estimating the k-nearest points to an edge by calculating the shortest distance between a line segment and a point in the three-dimensional space. The edge label is assigned via majority voting of the k-nearest points; classes are weighted by the inverse distance of a point w.r.t. the corresponding line segment ($w_i = \frac{1}{d_i}$).

Since the meshing process eliminates the geometries of small objects that are a separate class in the point cloud annotations – such as cars and power lines – the segmentation problem is simplified to four classes. Table 1 maps the point cloud labels, provided by the ISPRS Vaihingen data set to the class labels used in this research.

3.3 Chunking

To process a sizeable topographic mesh via MeshCNN, it needs to be divided into chunks. The tiles generated by SURE are not suitable as an input for our machine learning model for two reasons: 1) their size is defined by bounding boxes rather than the number of elements within a chunk; 2) there is the possibility that the mesh is split in z-direction within areas with high elevation which can lead to a highly fragmented mesh. For those reasons, we propose an alternative chunking procedure based on the breadth-first search algorithm. It uses the inherent mesh properties to divide the mesh into chunks containing an equal amount of elements while ensuring that no disconnected elements are created. The procedure can be applied on any coherent error-free mesh, which in our case is after Poisson reconstruction is applied.

The partitioning of a large topographical data set is usually done on the XY-plane while the number of samples per chunk is controlled in a way such that each chunk has the same amount of

[1] https://www.nframes.com/products/sure-aerial/

[2] https://www.blender.org/

elements (Winiwarter et al., 2019). In this approach, the density of elements needs to be evenly distributed over the scene model. While this assumption holds deterministically for voxel grids and stochastically for most kinds of point clouds generated by aerial scans, it is in the nature of mesh grids that its vertex density may differ substantially. Therefore, it is not practical to use the three-dimensional Cartesian space for chunking. In this work, an adaption of the well-known Breadth-First Search (BFS) algorithm for graphs is used to generate overlapping mesh chunks with an equal number of vertices. Algorithm 1 describes the procedure formally.

Algorithm 1 BFS mesh chunking

```
1:  function GetChunks(Mesh, ChunkSize, SeedCoord)
2:      let Chunks be a list
3:      let VisitedFaces be a set
4:      while Length(VisitedFaces) < Length(Mesh.Faces)
        do
5:          Start ← closest unvisited face to SeedCoord
6:          Chunk ← BuildChunk(Mesh, Start,
                               ChunkSize)
7:          Chunks.append(Chunk)
8:      end while
9:      return Chunks
10: end function

11: function BuildChunk(Mesh, Start, ChunkSize)
12:     let Q be a queue
13:     let ChunkFaces be a set
14:     let ChunkEdges be a set
15:     Q.enqueue(Start)
16:     while Length(Q) > 0 do
17:         f ← Q.dequeue()
18:         if Length(ChunkEdges) == ChunkSize then
19:             exit while
20:         end if
21:         if Length(ChunkEdges ∪ f.edges) > ChunkSize
            then
22:             continue while
23:         ChunkFaces.add(f)
24:         ChunkEdges.extend(f.edges)
25:         for AdjFace in Mesh.adjacentFaces(f) do
26:             if not ChunkFaces.contains(AdjFace) then
27:                 if not Q.contains(AdjFace) then
28:                     Q.enqueue(AdjFace)
29:                 end if
30:             end if
31:         end for
32:     end while
33:     return ChunkFaces
34: end function
```

The `GetChunks` function receives the following input: 1) a mesh object, holding information about vertices, edges, and faces as well as their relationships; 2) the desired chunk size defined by the number of edges; 3) a reference coordinate in the Cartesian 3D space (`SeedCoordinate`) which defines the starting node for each BFS iteration. Only complete faces are added to a chunk to ensure the generation of manifold geometries. Starting from the nearest face to the reference coordinate[3], the procedure visits all elements in a queue until the desired chunk size is reached. When a face is selected, all adjacent faces not part of the chunk nor queue are added to the queue. Two faces are defined as adjacent if they share at least one vertex.

It is important to note that faces can be part of multiple chunks, but as soon as a face is part of at least one chunk, it cannot be selected as a starting node. This procedure ensures all edges are selected at least once, while each chunk has exactly the desired size. At the same time, an overlap between the chunks is likely. This overlap is particularly desired in semantic segmentation to enhance reliability via face-class voting. Adding a face to the chunk will either add one or two edges, depending on which of its neighbors are already present. For this reason, the procedure needs to check the size of the union of the chunk and the queued face before adding it (see Algorithm 1, Line 21).

[3] w.r.t. to the origin at $x, y, z = 0, 0, 0$. This ensures that the face closest to the origin and not part of any chunk is chosen as the starting face in each iteration.

3.4 Inclusion of Photometric Features

MeshCNN is purely based on geometric information and does not encode any photometric features from the mesh texture. Rouhani et al. (2017) and Laupheimer et al. (2020b) present semantic segmentation approaches for urban scenes in which texture color information is included. Both argue that the HSV color space is to be preferred over the RGB color space since it effectively discriminates against objects with different reflectivity and is less sensitive to illumination differences.

MeshCNN is augmented with color features from transformed HSV space. The hue channel (H) in HSV space represents a value between $0°$ and $360°$ on the color circle. In order to ensure a correct proximity interpretation of the model, this circular feature is encoded into two continuous features by deriving sine and cosine transforms. The saturation channel (S) is included in those two features and not added as a separate one. This ensures that the arithmetic mean applied during MeshCNN's pooling operation of two instances results in a correct color encoding. In other words: if the angle of the vector on the color circle would be encoded separately from its length, the arithmetic mean would not be a valid pooling operator for those features. The value channel (V) representing brightness is added as a separate feature. In order to generate HSV color features for an edge, all texture pixels of its two adjacent faces are first extracted and subsequently averaged. Let $x_1, ..., x_n$ be all the pixels that belong to the texture of one of the two adjacent faces of an edge, then the three photometric edge features can be expressed as follows:

$$f_{edge,1} = \frac{1}{n}\sum_{i=1}^{n}\left(\sin\left(\frac{2\pi \cdot H(x_i)}{360°}\right)\cdot S(x_i)\right)$$

$$f_{edge,2} = \frac{1}{n}\sum_{i=1}^{n}\left(\cos\left(\frac{2\pi \cdot H(x_i)}{360°}\right)\cdot S(x_i)\right)$$

$$f_{edge,3} = \frac{1}{n}\sum_{i=1}^{n} V(x_i)$$

3.5 Experimental Setup

For experiments, a training set and test set were generated with 96 and 52 chunks respectively. Each chunk consists of 5,700 edges but chunks can overlap in their respective sets. Training of MeshCNN was executed on a single GeForce RTX 2080 Ti GPU with 11 GB of GPU memory. Three variations of the model were trained, including either geometric or color features or both. All three variations are set up with hyperparameters selected based on initial evaluation on a validation data set: Models were trained with a batch size of 8 chunks over 20 epochs with a starting learning rate of 0.001 and 30 additional epochs with a decaying learning rate. The resolution of the pooling layers was 4000, 2500, and 1250 while the convolution filters

		Classes				$\sum$
		Ground	Low Veg	High Veg	Building	
edges	train	0.168	0.158	0.288	0.386	518,700
	test	0.150	0.109	0.225	0.516	279,300
faces	train	0.196	0.173	0.270	0.361	339,451
	test	0.175	0.122	0.220	0.483	182,710

Table 2. Class distribution after data pre-processing. The four class columns are denoted by the ratio of that class. The final column denotes the total amount of primitives.

had sizes of 32, 64, 128, and 256. No data augmentation was applied. All other hyperparameters were set according to the default settings of MeshCNN. Python 3.7 and Pytorch (version 1.1.0) were used to execute the training.

The COG Cloud model (Laupheimer et al., 2020a,b), described in Section 2.1, is used as a baseline for this research. In the COG approach, the mesh is interpreted as a point cloud of the centers of gravity of its faces and classified by *Pointnet++* (Qi et al., 2017) in its single scale grouping (SSG) setup. Next to the X, Y, and Z coordinates of the center of gravity, the normal of a face is added as a geometric feature to encode face orientation. Photometric information of the texture is encoded as HSV features, as described in Section 3.4. Two variations of the baseline model are evaluated: one solely based on the geometric information, a second one where both the geometric and the photometric features are taken into account. Please note that a *color-only* model is not possible since the spatial relation of the faces is internally represented through the x, y, and z coordinates. In contrast, MeshCNN provides the spatial relation of neighboring edges externally and not through the geometrical features. The baseline model was trained with a batch size of four samples in 200 epochs with a learning rate of 0.001. All other hyperparameters were set according to the default settings of Pointnet++. The same train and test samples, as in the MeshCNN setup, were used. Since a constant number of edges does not necessarily result in a constant number of faces, the number of faces to be considered during training was downsampled to 3685 for all train and test samples. Python 3.5 and Tensorflow (version 1.13.0) were used to execute the training with Pointnet++.

Since the baseline model assigns labels to faces rather than edges, the ground truth labels need to be recalculated. Analogous to the procedure that is described in Section 3.1 the labels are derived from the ALS point cloud of the ISPRS Vaihingen data set with the sole difference that nearest neighbors are not calculated based on the line segment distance w.r.t. an edge but based on the distances between a face's center of gravity and the ALS points. Table 2 shows the class distribution w.r.t. edges and faces, respectively.

Both Pointnet++ and MeshCNN use classification accuracy as a metric to select the best model based on the evaluation set. However, the authors of MeshCNN introduced a soft accuracy measure as an evaluation metric for the prediction of edge labels to counteract the effect of ambiguous edge classes between faces with different semantic classes. The soft accuracy considers a prediction to be correct when at least one of the first ring neighbors of an edge or the target edge itself is of that class. In order to meet the slight class imbalance as described in Table 2, the Jaccard index – also known as Intersection over Union (IoU) – per class and the mean IoU of all classes are used for additional evaluation.

Five model training runs per setup are executed in order to investigate the robustness of the metrics. Since this sample size is too small to conclude the distribution, the exact Wilcoxon rank-sum test is used to test the statistical significance of the differences between setups ($\alpha = 0.05$). For that purpose, the mean Intersection over Union (mIoU) is the relevant metric, as it is robust against class imbalance.

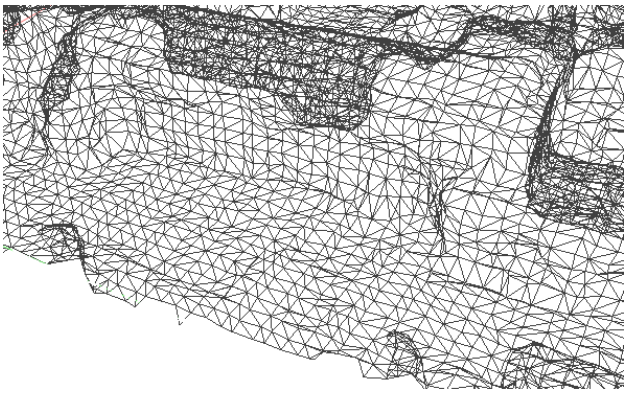


Figure 2. Wireframe view of a partial house and street in the test set. The generated uniformity of the mesh might affect the ability of MeshCNN to learn geometric representations correctly.

4. EVALUATION

In this section, the results of the experiments are presented and discussed. Table 3 shows the evaluation metrics on the test set for the different experimental setups.

From Table 3, it can be seen that both PointNet++ and MeshCNN perform similarly poorly when only geometric features are used. The reason for this might be two-fold: On the one hand, the data set contains classes that are very similar in terms of their geometrical shape. Low vegetation (in most cases grass) usually cannot be differentiated from solid ground without photometric or radiometric information. On the other hand, the mesh is considerably uniform w.r.t. its face areas (see Figure 2). This property was already present to a certain degree in the original SURE output and was likely enhanced by the Poisson Surface Reconstruction. This uniformity is critical since mesh encoding does not exploit the beneficial mesh property of non-uniformity. Detailed areas are overly smoothed, and at the same time, large flat areas are represented by too many elements. At least for MeshCNN, it can be expected that a uniform mesh harms predictive performance.

The COG cloud baseline model and MeshCNN's predictive performances are significantly better when photometric features are added. Since geometric features have shown to be rather weak predictors and both models use the same encoding of color features, it is expected that the performances of both models are very similar when both geometric and photometric features are used. A two-sided Wilcoxon rank-sum test does not show a significant difference (p-value=0.063).

However, the MeshCNN setup, where geometric features were omitted, performs significantly worse than the setup where both types of features were considered (p-value=0.031). This means that even though the relevance of geometric features in this data set is rather low, they do contribute to predictive performance to some extent.

Further, a comparison between the IoU's per class (Table 3) and the class distributions (Table 2) shows that the more often

Model	Features		Accuracy		IoU				
	Geometric	Color	Soft	Hard	All	Ground	LowVeg	HighVeg	Building
Pointnet++	✓	✓	-	0.746±.004	0.529±.007	**0.567±.027**	**0.314±.016**	0.506±.012	0.729±.001
Pointnet++	✓		-	0.504±.006	0.287±.008	0.321±.038	0.164±.026	0.175 ±.023	0.488±.022
MeshCNN	✓	✓	**0.814±.001**	**0.751±.001**	**0.539±.002**	0.561±.001	0.311±.010	**0.532±.013**	**0.754±.001**
MeshCNN	✓		0.542±.013	0.446±.014	0.278±.001	0.383±.019	0.101±.011	0.307±.001	0.316±.033
MeshCNN		✓	0.785±.001	0.723±.006	0.501±.001	0.497±.018	0.283±.001	0.516±.011	0.727±.012

Table 3. Evaluation metrics for Pointnet++ and MeshCNN using either geometric features, photometric feature or both. The **first** and second best scoring methods are highlighted for each metric. Note inclusion of color features boosts performance significantly.

a class occurs in the data set, the higher is its Jaccard index. Before declaring this effect as systematic, further investigation is required. This behavior is likely related to the selection of models based on the accuracy metric. More importantly, setting a different loss function could remedy this model bias.

Figure 3 shows the confusion matrices of the best performing models of each of the experimental setups considering only geometric features. Generally speaking, both models show a poor segmentation performance. The predictions for the two classes Ground and Building are better than random. Furthermore, the classes Ground and Low Vegetation are confused considerably more often than in Figure 4a and Figure 4b. The models behave similarly for the classes High Vegetation and Building. An intuitive explanation is that those pairs of classes are similar w.r.t. their geometry.

Figure 4a and Figure 4b represent the results of the models using both geometric and photometric features. The models show a similar result as the Ground and Building classes are less often confused with other classes than the two vegetation classes. This can be explained by the similar photometric appearance of the latter classes, also apparent from Table 3. A further intuitive explanation is that it is hard to distinguish between low and high vegetation as there might be many edge cases in between those. Errors are further compounded by the smoothing introduced by the data pre-processing.

Figure 4c shows the confusion matrix for MeshCNN when only photometric features are used. Even though geometric features are omitted, MeshCNN still exploits the connectivity of mesh elements. Intuitively, one might expect similar color information between the two vegetation classes and between the Ground and Building classes, resulting in much predictive confusion. Surprisingly, only a marginal difference is visible. This indicates that either there is a significant difference in color information or that the contextual information – i.e., the adjacent elements – improve the segmentation. Intuitively, low vegetation (grass) is more likely to be adjacent to ground or buildings than high vegetation. Then again, bushes and trees are likely to be adjacent to low vegetation.

Figure 5 depicts a sample from the test set. In this sample, it can be seen that the boundaries between different classes are challenging to classify correctly. This might be caused by the poor quality of the mesh, and especially by the blurred textural boundaries between regions.

5. CONCLUSION AND OUTLOOK

This research is the first to apply MeshCNN – a deep learning model that exploits the unique properties of a mesh, such as irregularity and non-uniformity – in the photogrammetry domain.

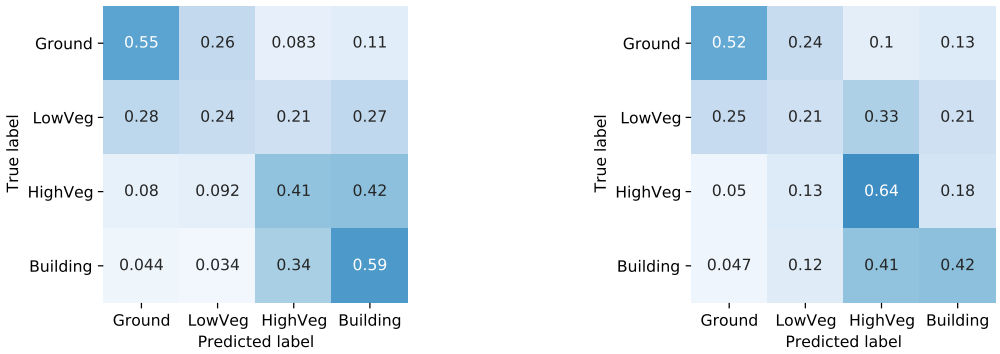
Numerous challenges arise when applying a mesh-based deep learning model to a photogrammetric mesh, especially w.r.t. data set properties. First, due to the lack of publicly available benchmarking data sets for semantic segmentation of photogrammetric meshes, a mesh from the ISPRS Vaihingen 3D data set was created by using the commercial product SURE Aerial. This data set is not ideal for mesh creation due to the lack of oblique images and trajectory data. Second, mesh errors, such as zero-length edges and intersecting faces, require post-processing, which – in the current work – resulted in the creation of a uniform Poisson mesh. As a consequence, conclusive statements on the usefulness of mesh-based deep learning for exploiting purely geometrical details in urban scenes are difficult to make. However, the generated mesh can be used to evaluate the general properties of photogrammetric meshes, as it can be assumed that those are independent of the mesh quality.

We believe several things can advance mesh-based deep learning in photogrammetry. For one, we hypothesize that MeshCNN specifically is better suited to exploit an adaptive non-uniform mesh than a uniform mesh. Therefore, future research could address the use of more suitable mesh post-processing to yield a less uniform mesh appearance. For example, mesh reconstruction might make use of Delauney triangulation, or apply edge collapse on the Poisson mesh. Another idea is the use of explicit shape primitives during mesh construction, similar to the work of Lafarge and Mallet (2012). Besides, a new benchmarking data set for photogrammetric meshes could be of tremendous help to further advance mesh-based deep learning in photogrammetry. The field is moving rapidly and new data sets are being already being announced or have been released recently (e.g. Gao et al., 2021; Kölle et al., 2021, that were published after this work was concluded). In order to ease the integration of photogrammetric meshes into automated machine learning pipelines, providers of meshing software could focus on fulfilling the mesh properties described in this research: Enabling error-free, watertight, and adaptive non-uniform meshes as well as offering a mesh-oriented chunking procedure, compared to the one described in this paper.

Finally, additional research could be conducted on how to incorporate more advanced texture information into MeshCNN instead of simple color features. There exist cases where faces have visibly different textures but the same average (or median) colors. Future research could develop a combined model – for example one that uses convolutional layers as textural encoders – that calculates photometric features directly from the texture.

ACKNOWLEDGEMENTS

We would like to thank the company nFrames for allowing the usage of SURE Aerial for this research, and the colleagues at Cloudflight for the organisational support. Also, we would like to thank to the anonymous reviewers for their helpful feedback.



(a) Confusion matrix of Pointnet++ using geometric features only

(b) Confusion matrix of MeshCNN using geometric features only

Figure 3. Confusion matrices using geometric features only. Both models show a rather weak predictive performance. Without texture information, the models struggle to distinguish buildings from high vegetation and ground from low vegetation, which is not least caused by the comparably poor mesh quality.

(a) Confusion matrix of Pointnet++ using geometric and photometric features

(b) Confusion matrix of MeshCNN using geometric and photometric features

(c) Confusion matrix of MeshCNN using photometric features only

Figure 4. Confusion matrices with inclusion of photometric features. Both models show a similar predictive performance when geometric and photometric features are used together (**a** and **b**). The two vegetation classes are confused with each other the most. There might be two reasons: 1) In some cases it might be hard to draw the line between the two classes during annotation; 2) Photometric features are obviously dominating geometric ones in this setup. An evaluation of MeshCNN with color features only (**c**) supports the second argument and also shows that geometric features at least have some impact as the vegetation classes do get confused more often without them.

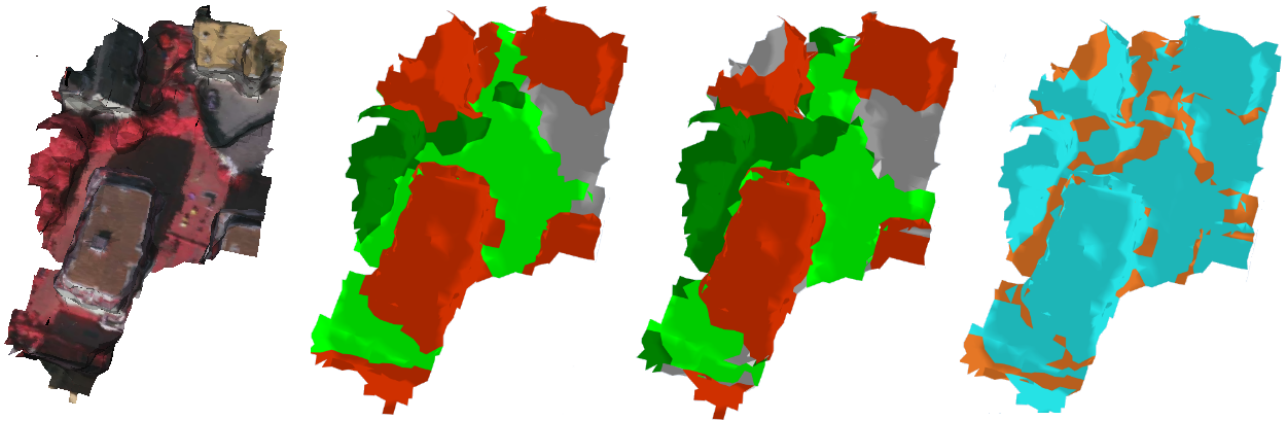


Figure 5. A test set sample classified by MeshCNN with photometric and geometric features: **(left)** The textured mesh; **(mid-left)** ground truth; **(mid-right)** prediction, where dark green, light green, gray, and red colors encode high vegetation, low vegetation, ground, and buildings respectively; **(right)** difference plots between ground truth and prediction, where orange are wrongly predicted labels.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2021-59-2021 | © Author(s) 2021. CC BY 4.0 License.

REFERENCES

Bláha, M., Vogel, C., Richard, A., Wegner, J. D., Pock, T., Schindler, K., 2016. Large-scale semantic 3d reconstruction: An adaptive multi-resolution model for multi-class volumetric labeling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3176–3184.

Cabezas, R., Straub, J., Fisher, J., 2015. Semantically-aware aerial reconstruction from multi-modal data. *International Conference on Computer Vision (ICCV)*, 2156–2164.

Chen, X., Golovinskiy, A., Funkhouser, T., 2009. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3), 1–12.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., 2008. MeshLab: An open-source mesh processing tool. *6th Eurographics Italian Chapter Conference 2008 - Proceedings*, 129–136.

Gao, W., Nan, L., Boom, B., Ledoux, H., 2021. SUM: A Benchmark Dataset of Semantic Urban Meshes. *arXiv Preprint*.

Griffiths, D., Boehm, J., 2019. A Review on deep learning techniques for 3D sensed data classification. *Remote Sensing*, 11(12), 1499.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J., Schindler, K., Pollefeys, M., 2017. Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1, 91-98.

Hackel, T., Wegner, J. D., Schindler, K., 2016. Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 177–184.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D., 2019. MeshCNN: A network with an edge. *ACM Transactions on Graphics*, 38(4).

Huang, J., You, S., 2016. Point cloud labeling using 3d convolutional neural network. *23rd International Conference on Pattern Recognition (ICPR)*, 2670–2675.

Häne, C., Zach, C., Cohen, A., Pollefeys, M., 2017. Dense Semantic 3D Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9), 1730-1743.

Kazhdan, M., Hoppe, H., 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3), 1–13.

Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., Ledoux, H., 2021. The Hessigheim 3D (H3D) Benchmark on Semantic Segmentation of High-Resolution 3D Point Clouds and Textured Meshes from UAV LiDAR and Multi-View-Stereo. *arXiv Preprint*.

Lafarge, F., Mallet, C., 2012. Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1), 69–85.

Laupheimer, D., Shams Eddin, M. H., Haala, N., 2020a. On the Association of LiDAR Point Clouds and Textured Meshes for Multi-Modal Semantic Segmentation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2, 509–516.

Laupheimer, D., Shams Eddin, M. H., Haala, N., 2020b. The Importance of Radiometric Feature Quality for Semantic Mesh Segmentation. *40. Wissenschaftlich-Technische Jahrestagung der DGPF in Stuttgart – Publikationen der DGPF*, Stuttgart, 205–218.

Leotta, M., Long, C., Jacquet, B., Zins, M., Lipsa, D., Shan, J., Xu, B., Li, Z., Zhang, X., Chang, S.-F., Purri, M., Xue, J., Dana, K., 2019. Urban semantic 3d reconstruction from multiview satellite imagery. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 1451–1460.

Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P., 2015. Geodesic convolutional neural networks on riemannian manifolds. *Proceedings of the IEEE international conference on computer vision workshops*, 37–45.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87, 152–165.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, December, Neural information processing systems foundation, 5100–5109.

Rouhani, M., Lafarge, F., Alliez, P., 2017. Semantic segmentation of 3D textured meshes for urban scene analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 123, 124–139.

Schmohl, S., Sörgel, U., 2019. Submanifold Sparse Convolutional Networks for Semantic Segmentation of Large-Scale ALS Point Clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5, 77–84.

Tatarchenko, M., Park, J., Koltun, V., Zhou, Q. Y., 2018. Tangent Convolutions for Dense Prediction in 3D. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3887–3896.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2018. SEGCloud: Semantic segmentation of 3D point clouds. *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, 537–547.

Tutzauer, P., Laupheimer, D., Haala, N., 2019. Semantic Urban Mesh Enhancement Utilizing a Hybrid Model. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W7, 175–182.

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

Winiwarter, L., Mandlburger, G., Schmohl, S., Pfeifer, N., 2019. Classification of ALS Point Clouds Using End-to-End Deep Learning. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 87(3), 75–90.

Zhao, R., Pang, M., Wang, J., 2018. Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science*, 32(5), 960–979.