The Importance of Radiometric Feature Quality for Semantic Mesh Segmentation

DOMINIK LAUPHEIMER[1], MOHAMAD HAKAM SHAMS EDDIN[1] & NORBERT HAALA[1]

Abstract: We propose a pipeline for the semantic segmentation of textured meshes in urban scenes as generated from imagery and LiDAR data. Key idea is to represent the mesh as a set of face centroids (COG cloud). This enables the comparison of various point-based classifiers of varying learning abilities. Fine-tuned PointNet++ showed the best results due to hierarchical feature learning. One of the main differences between meshes and point clouds is the availability of high-resolution texture. Hence, we evaluate the importance of radiometric feature quality as a proxy for texture importance. Color information increases performance by at least 5 % *(mIoU) for the used data. We achieved to double the performance gain by improving the radiometric feature quality, i.e. utilizing color information of the entire face. Our study shows that texture is beneficial for non-uniform dense and non-balanced data sets. However, it also shows the inherent limitations of textural features like occlusions, absence of imagery, and the quality of the geometric reconstruction.*

1 Introduction

The semantic segmentation of 3D data is an everyday issue in the domain of photogrammetry and remote sensing. Common 3D data representations are point clouds, meshes, volumetric representations, and projected views (i.e. RGB-D images or renderings). Amongst them, the semantic segmentation of point clouds may currently be the most popular topic.

The unstructured nature of 3D point sets prevents to apply well-established Deep Learning (DL) methods of the image space directly to point clouds. Accordingly, common approaches structure data into grid-like representations by voxelization (3D) or multi-view rendering (2D) (GRAHAM et. al. 2018; BOULCH et al. 2017). Both approaches enable the use of Convolutional Neural Networks (CNNs) but come along with information loss (discretization, occlusions, projection, etc.). Therefore, DL approaches that directly operate on 3D point clouds have been investigated recently (QI et al. 2017a; QI et al. 2017b; BOULCH 2019). The huge number of points and the related matter of subsampling are still big issues.

We claim that meshes are well-suited to tackle these issues due to their lightweight geometric representation. Generally, points on planar surfaces do not provide extra information and hence, do not have to be stored. During the meshing process, such points are eliminated resulting in a smaller memory footprint. For instance, the mesh footprint of the used high-resolution data set (cf. section 2.1) covers ~30 % of the respective LiDAR point cloud (considering *XYZ* only).

Compared to voxelization or multi-view renderings, structuring a point cloud in the mesh representation, ideally, does not come along with information loss since neither rasterization nor projection is involved. Quite the contrary, for closed surfaces, meshing may increase the

[1] Universität Stuttgart, Institut für Photogrammetrie, Geschwister-Scholl-Straße 24D, D-70174 Stuttgart, E-Mail: [Dominik.Laupheimer, Norbert.Haala]@ifp.uni-stuttgart.de, hakam.shams@hotmail.com

information content due to an explicit topology and an explicit surface description (i.e. unambiguous normal vectors).
Moreover, textured meshes carry high-resolution image information stored in so-called texture atlases. In summary, textured meshes provide geometric and textural information in a lightweight fashion and inherently enable data fusion from LiDAR and image data acquisition. According to our observation, meshes currently replace unstructured point clouds as a final user product.
For these reasons, we investigate the semantic segmentation of textured meshes in urban areas as generated from LiDAR data and oblique imagery. Despite their advantages, meshes are a mostly overlooked topic in the domain of photogrammetry and remote sensing. So far, only a few works deal with semantic mesh segmentation in urban scenes (ROUHANI et al. 2017; TUTZAUER et al. 2019). In comparison, in the domain of computer vision, meshes are a default data representation. However, that community deals with small-scale (indoor) data sets (KALOGERAKIS et al. 2017; GEORGE et al. 2017).
We establish a pipeline to segment meshes semantically with three different classifiers: Random Forest (RF), PointNet and PointNet++ (cf. section 2.3). Section 2 describes data preprocessing and feature calculation. The used data is described in detail in (CRAMER et al. 2018; TUTZAUER et al. 2019) and visualized in Fig. 1. Section 3 describes the implementation of the used classifiers and compares their performance. In this study, we focus on the texture importance for the semantic segmentation of meshes utilizing a PointNet++ classifier. We assume that color information in the form of high-resolution texture is very important for semantic segmentation. The color information may have more impact than per-point color information in case of point clouds. The results are discussed in section 4.

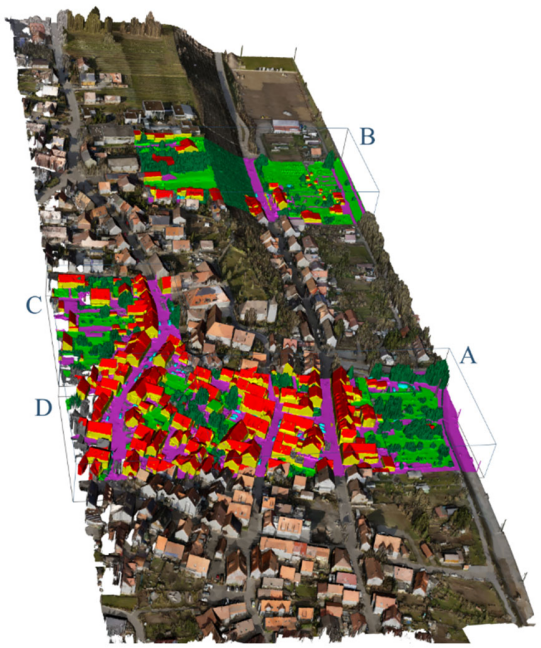

Fig. 1: Used data for training and evaluation ($800\,m \times 300\,m$). Tiles A, B, C, D (depicted in labeled fashion) are used as validation or test set (mutually exclusive). The remaining part (colored by median RGB per face) is used as training set. Classes: *building mass/facade* (yellow), *roof* (red), *impervious surface* (violet), *green space* (light green), *mid and high vegetation* (dark green), *vehicle* (light blue), *chimney/antenna* (orange), and *clutter* (gray).

2 Methodology

Compared to point clouds, meshes provide high-resolution texture instead of per-point color only. Hence, we want to investigate the importance of available color information per face. Our approach represents the mesh as a set of face centroids and leverages point-based classifiers. We evaluate radiometric feature importance at various qualities as a proxy for texture importance.

2.1 Data Set and Data Preparation

Our study uses the high-resolution data described in (CRAMER et al. 2018). The Airborne Laser Scanning (ALS) data consists of up to 800 points/m² for the entire area. The GSD of the oblique images is ~2.5 cm. TUTZAUER et al. (2019) obtained a textured 2.5D mesh by fusing the simultaneously acquired ALS data and aerial oblique imagery with software SURE 2 from nFrames (ROTHERMEL et al. 2012). They manually labeled the mesh and split the data into training, validation and test set (cf. Fig. 1). The considered classes are (relative frequency is given in parentheses): *building mass/facade* (9.28 %), *roof* (6.34 %), *impervious surface* (5.67 %), *green space* (5.97 %), *mid and high vegetation* (63.38 %), *vehicle* (0.83 %), *chimney/antenna* (0.31 %), and *clutter* (8.22 %). In regards to the *closed world assumption*, class *clutter* contains all faces that do not match the other classes.

Similar to TUTZAUER et al. (2019), we represent each face by its center of gravity (COG) associated with features (cf. section 2.2). To this end, the mesh is represented by a set of COGs (COG cloud). While still benefiting from mesh-based features like the availability of high-resolution texture, we can apply classifiers that have been designed for point clouds originally.

In our experiments, we focus on PointNet++ since it outperforms the other tested classifiers (cf. section 3.2). In accordance with experiments of (QI et al. 2017a), our experiments showed increased performance with an increased number of input points (i.e. number of sampled COGs in the first level of PointNet++). However, performance gain saturates at a certain input density. Furthermore, the number of input COGs is limited due to GPU memory. For this reason, we partition the data into spatially overlapping tiles with a fixed dimensionality of $50\ m \times 50\ m$ and an 80 % overlap. Based on these, we generate mini-batches in two different flavors to train PointNet/PointNet++: squared mini-batches or circular mini-batches (QI et al. 2017b; WINIWARTER et al. 2019). Squared mini-batches subsample the generated tiles using random sampling or Farthest Point Sampling (FPS) on the fly (cf. section 3.1). The circular-shaped mini-batches (viewed from above) are generated around the tile centers using kNN, i.e. no subsampling is involved. At the edge of acquired data, the mini-batches can have arbitrary shape since the tile center may already be at the edge.

Whereas memory limits the upper bound of mini-batch sizes, the scene and considered classes limit the lower bounds of mini-batch dimensions. For instance, mini-batches should cover the whole shape of a building, car, etc. and incorporate sufficient spatial context.

For evaluation, we aggregate predictions of overlapping areas for PointNet/PointNet++. We assume the results will be more stable due to redundant predictions (cf. Fig. 6).

Semantic mesh segmentation differs from semantic point cloud segmentation in several aspects. Generally, face density is significantly sparser than point density. Hence, the COG cloud is sparse and has a comparatively small memory footprint. For instance, tile A consists of ~39.6 million

LiDAR points or ~0.3 million faces/COGs respectively. The COG cloud is ~133 times smaller. On the other hand, meshing increases non-uniform density and class imbalance. Non-uniform density is a mesh-inherent issue. Meshes tend to have larger faces in planar areas and many small faces in vivid areas. Thereby, the meshing implicitly shifts class imbalance towards non-planar classes (e.g. vegetation classes). Therefore, handling the non-uniformity and class imbalance is of particular importance. We tackle non-uniform density and class imbalance with FPS, Multi-Scale Grouping (MSG), random dropout on the fly and class weighting (cf. section 3.1). Fig. 2 compares (non-uniform) densities of the dense LiDAR point cloud the respective COG cloud. It also shows the label distributions before (i.e. LiDAR point cloud) and after meshing (i.e. COG cloud) where both data representations are labeled. The LiDAR point cloud was manually labeled (KÖLLE et al. 2019). The labels have been transformed to fit our label scheme. Since the label scheme of the LiDAR point cloud does not consider *chimney/antenna* we manually labeled chimneys and antennas for the comparison. The standard deviation of the class frequencies can be interpreted as a measure for the deviation of equal distribution / class balance (23.6 % for the mesh, 14.6 % for the point cloud).

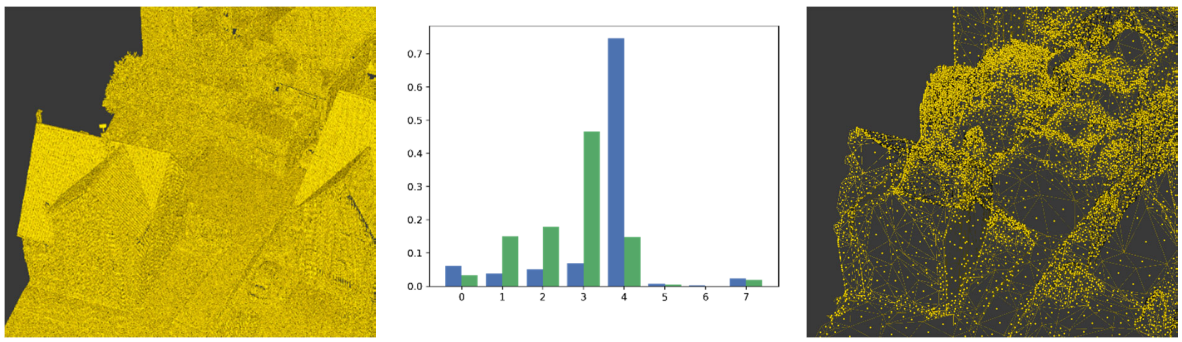

Fig. 2: Non-uniform density of the very dense LiDAR point cloud (*left*) and the mesh represented as COG cloud (*right*). The histogram (*center*) shows the class imbalance of the manually labeled point cloud (*green*) and manually labeled mesh (*blue*) where both representations overlap. Classes: *building mass/facade* (0), *roof* (1), *impervious surface* (2), *green space* (3), *mid and high vegetation* (4), *vehicle* (5), *chimney/antenna* (6), and *clutter* (7).

2.2 Feature Calculation

TUTZAUER et al. (2019) attach handcrafted contextual features to the COG cloud. They calculate various geometric and radiometric features based on several scales for each face. Similar to TUTZAUER et al. (2019), we associate per-face features with COGs. However, we do not use contextual features and adapt only the normal vector and the median HSV per face. The normal vector is computed by the cross product of per-face edges. The median HSV per face is extracted from the texture atlas. First, texture coordinates of vertices are transformed into image coordinates. Subsequently, we calculate the median HSV based on the entire face projected onto the texture atlas utilizing previously calculated image coordinates. Our approach can be extended easily by other handcrafted features. However, we limit ourselves to mesh-inherent features in order to reduce computation overhead.

We assume that color information in the form of high-resolution texture is very important for semantic mesh segmentation. The color information may have more impact than per-point color information in case of point clouds. Therefore, our objective is to evaluate the importance of

radiometric feature quality as a proxy for texture importance. Please note, the proxy is needed since the considered classifiers rely on 1D feature vectors. To the best of our knowledge, no classifier is able to combine 1D features with 2D texture information per face. For this reason, we calculate scalar radiometric features with varying granularity. That is why we additionally extract HSV features, which carry less color information. Besides the median HSV per face (*median HSV (face)*), we calculate the mean and median HSV based on the three vertex colors only (*mean HSV (vertices)* and *median HSV (vertices)* respectively). The latter two mimic a colored meshed point cloud, i.e. color information is only available for mesh vertices. We choose HSV space to be independent of illumination variations. Fig. 3 depicts the different qualities of color features transformed into RGB space. The median versions (robust against outliers) provide crisper colors whereas the mean version smears colors the most. *median HSV (face)* uses color information of the entire face and therefore represents faces at border areas (e.g. the transition of the roof to facade) better than *median HSV (vertices)*. In particular, for large faces, this may ensure correct colors (cf. schematic drawing in Fig. 3) and stabilize the semantic segmentation.

Fig. 3: Comparison of radiometric features at different qualities in RGB color space. The faces are colored based on mean/median RGB values. Quality of radiometric features increases from left to right. *Left*: Mean RGB of vertices, *center*: median RGB of vertices, *right*: median RGB of all pixels per face. A schematic drawing is attached to each subfigure in the upper left.

2.3 Classifiers Applied

Representing the mesh as COG cloud enables using point-based classifiers. We compare several classifiers with different capabilities of context mapping or hierarchical learning respectively. In this study, we compare the pre-deep-learning era classifier RF with the first neural networks applicable to unordered point sets (PointNet and its extension PointNet++).

The gist of PointNet is to use a symmetric function that is independent of set permutation. The entire COG cloud is encoded in a global feature vector, which is attached to each encoded per-face feature vector. The prediction bases on this concatenated feature vector. A downside of PointNet is the weak local context as it only operates on a global scale. Its extension PointNet++ hierarchically applies PointNets to the iteratively subsampled COG cloud and thus operates on several scales. This procedure enables hierarchical feature learning with increased contextual information similar to CNNs in image space. On the contrary, PointNet and RF cannot operate on

several scales. Moreover, vanilla RF is a pure per-point classifier and does not support feature learning at all. The provided feature set limits its performance. Therefore, RF relies more on expert knowledge/feature design than PointNet/PointNet++. PointNet++ needs the least expert knowledge. Both, PointNet and PointNet++ mainly rely on coordinates as features. Optionally, additional (handcrafted) features can be provided. Best classifier configurations are given in section 3.1.

3 Configuration and Comparison of Classifiers

In section 3.1, we describe configurations for each classifier, which we found to perform best for the used data set. In this respect, we focus on PointNet++ and the respective data preparation as we achieve the best results with this classifier (cf. section 3.2). We use recall, precision, and Intersection over Union (IoU) as per-class evaluation metrics for the COG cloud. Furthermore, we report overall accuracy (OA), mean recall (mR), mean precision (mP), and mean IoU (mIoU) at a global scale.

As we work with an imbalanced data set, OA suffers from the accuracy paradox. On the contrary, mIoU can be understood as an average per-class accuracy. The reported values might look low. However, this is due to adapting point cloud metrics to the COG cloud while not considering that each face covers a differently sized area. Thereby, each face has the same impact on the evaluation metrics although face areas vary significantly. As can be seen in Fig. 4 and Fig. 5, the major area is predicted correctly. For tile A, roughly 89 % of the surface area has been predicted correctly. Approximately 83 % of the entire test surface is predicted correctly.

PointNet/PointNet++ are implemented with the DL framework *TensorFlow* (QI et al. 2017a; QI et al. 2017b). RF relies on the *scikit-learn* implementation (python package). For all classifiers, both training and testing are performed on a machine with an NVIDIA GeForce GTX 1080 Ti GPU, 64 GB RAM, and a 12-core CPU.

3.1 Classifier Configurations

Like described in section 2.1, a major part for PointNet/PointNet++ is preparing mini-batches. We run experiments regarding different numbers of input COGs for our two mini-batch modes (squared and circular mini-batches). For squared mini-batches, FPS performed best for a small number of COGs since it provides a better distribution compared to random sampling. Circular mini-batches as generated by kNN do not give a sufficient representation of the neighborhood in that case. However, with an increased number of COGs, random sampling and FPS are almost on par (squared mini-batches) but worse than kNN (circular mini-batches). Random sampling is the fastest; FPS is the slowest method. In our case, circular mini-batches generated with kNN ($k = 18000$) provide reasonable sizes of mini-batches and give the best results at feasible computation times. The circular mini-batches perform better than the squared mini-batches since no subsampling is involved for the mini-batch creation. Therefore, the circular mini-batches provide higher information content.

Regardless of the mini-batch creation, mini-batches are subsampled with FPS in PointNet++ to define the centers of local regions. FPS is used since it is not agnostic to the data distribution. The

local regions are the input to the next level of PointNet++. Limited by GPU memory, 5000 local regions can be processed in the first level of PointNet++.
The local regions (demarcated around FPS-sampled COGs) can be defined in two variants: fixed radius (ball neighborhood) or fixed number of points (kNN). The definition of local neighborhoods with kNN did not perform well. We assume this may be due to non-uniform density limiting the capability of generalization (QI et al. 2017b). In case of fixed radius, MSG proved to be best performing with the following radii: [0.5 m – 1.0 m – 2.0 m] for the first level, [2.0 m – 4.0 m – 8.0 m] for the second level and [4.0 m – 8.0 m – 16.0 m] for the third level. The radii have been chosen heuristically according to the scale of the data set and dimensions of desired objects. MSG improves robustness against density variations but comes along with a great memory footprint and training/inferencing time. These quantities depend mainly on the size of the local regions (defined by number of points or radii), the number of scales, and the number of abstraction levels. To tackle non-uniform density further on, we apply random dropout on the fly. PointNet is less affected by the non-uniform density because of its global abstraction and weak local descriptors.
Both PointNet and PointNet++ use weighted sparse categorical cross-entropy as loss function. We tested several class weighting methods applied to the loss function during training. The best performance is achieved with the inverse square roots of the relative class frequencies in the training data as class weights (WINIWARTER et al. 2019; SCHMOHL & SOERGEL 2019). In order to avoid training on mini-batches that cover mainly a single class, we filter mini-batches whose standard deviation of relative class frequencies exceeds a predefined threshold, similar to (WINIWARTER et al. 2019). We found the best results with a threshold between 30-40%.
We found that PointNet and PointNet++ need distinct treatment for COGs. PointNet++ internally shifts COGs for each local region into a local frame centered at the FPS-sampled COGs. By this means, we achieve local learning based on relative coordinates while keeping the real scale. Counterintuitively, for PointNet, we achieved the best results when we scale the COGs to the unit sphere per tile. Thereby, we follow the original implementation (QI et al. 2017a).
Parameters of the default RF (acts on CPU) have been defined using a grid search with two parameters (*number of trees* and *depth of trees*). We trained an RF without *XYZ* and another with *XYZ* features. We achieved the best results with 30 trees with depth equals 13 (no *XYZ* features) and 46 trees and depth of 7 using *XYZ* features (OA = 70.80 % / mIoU = 23.49 % and OA = 69.54 % / mIoU = 22.46 % respectively). We report both versions since RF without *XYZ* is faster (~25 %) and performs better (~1 %). Nevertheless, for the sake of fair comparison, in section 3.2, we report only the version that uses *XYZ* features. However, we are aware of the fact that the used *XYZ* features for the RF cannot be compared directly with the used *XYZ* features of PointNet and PointNet++ due to the used scaling/normalization in the DL approaches.

3.2 Comparison of Classifiers

In section 2.3, we described the key differences of the considered classifiers. Fig. 4 shows ground truth and the predictions of all considered classifiers for tile C. Unavoidably, we suffer from label noise. For instance, in the upper left, a face on the roof is mislabeled. Some faces on the street are associated with wrong labels. Not correctly reconstructed cars are labeled as *clutter*, but are consistently predicted as *vehicle* for PointNet/PointNet++ (cf. oval in Fig. 4). Such faces will

decrease performance metrics. In particular, precision for class *vehicle* and recall for class *clutter* is decreased.

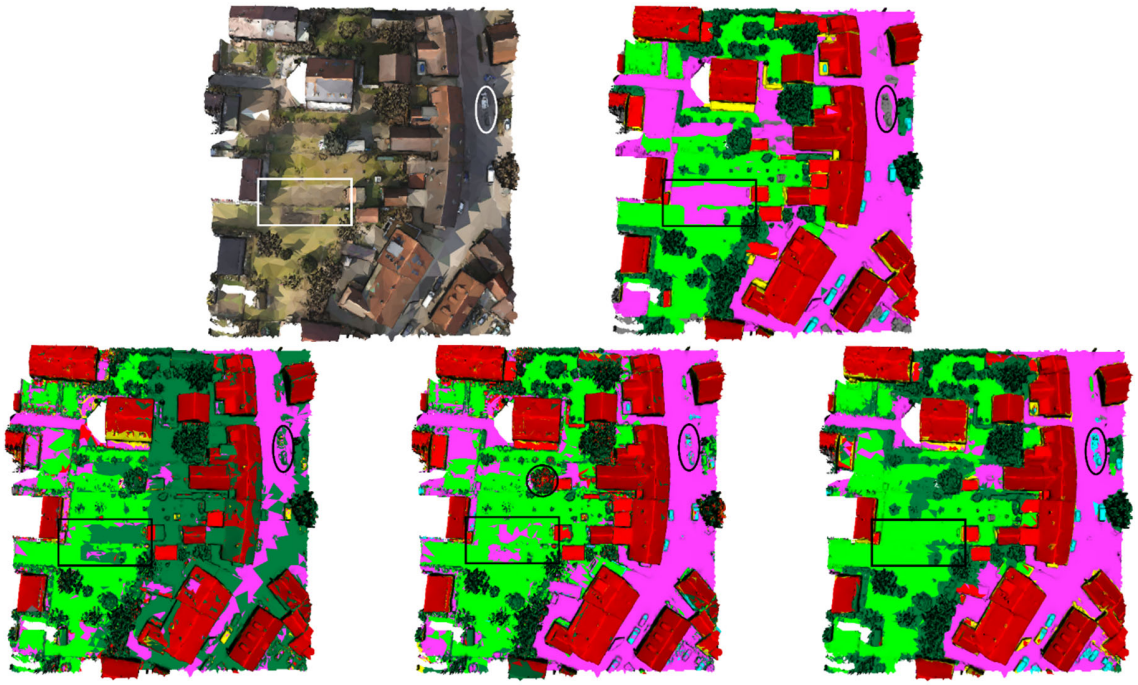

Fig. 4: Ground truth and predictions of several classifiers on tile C. The feature vector consists of *XYZ*, *normal vector* and *median HSV (face)*. *Top row*: Tile C colored with median RGB per face (*left*) and ground truth (*right*). The oval shows cars labeled as *clutter* (label noise) that are consistently predicted as *vehicle* for PointNet/PointNet++. *Bottom row (from left to right)*: RF (OA = 69.54 % and mIoU = 22.46 %), PointNet (OA = 70.89 %; mIoU = 29.44 %), PointNet++ (OA = 80.62 %; mIoU = 43.54 %). The circle shows a false prediction (*roof*, *chimney*/*antenna*) for a tree due to feature encoding at global scale (PointNet). The rectangle shows that all classifiers have issues separating *green space* from *impervious surface*. Best viewed digitally.

Section 3.1 denotes the best configurations for the used classifiers. For better comparability, the same feature vector configuration has been presented to all classifiers. The feature vector contains radiometric (*median HSV (face)*) and geometric (*XYZ*, *normal vector*) features. As expected, RF performs worst (OA = 69.54 %; mIoU = 22.46 %) and PointNet++ performs best (OA = 80.62 %; mIoU = 43.54 %). PointNet (OA = 70.89 %; mIoU = 29.44 %) performs ~14 % (mIoU) worse than PointNet++. This is due to the hierarchical feature learning of PointNet++ that causes spatially smoothed predictions at the same time. Furthermore, for PointNet/PointNet++, we aggregate predictions of overlapping tiles (which has a smoothing effect, too). However, even without aggregating, PointNet++ outperforms RF and PointNet. As can be seen in Fig. 4, PointNet++ outperforms the other classifiers in predicting classes *mid and high vegetation*, *vehicle* and *chimney/antenna* (i.e. classes with high and low support). For instance, the recall of *mid/high vegetation* is 88 % and 98 % for PointNet and PointNet++ respectively. A common issue of PointNet is to predict roofs and chimneys within trees due to its global feature encoding (cf. circle in Fig. 4). PointNet++ captures small details and therefore, detects classes with low support better (e.g. *vehicle* and *chimney/antenna*). All classifiers have issues separating *green space* from

impervious surface (cf. rectangle in Fig. 4). In particular, PointNet++ consistently labels dirt roads as *green space*. This indicates that even color information is not enough to separate those two classes to the full extent. The reason might be that dirt roads are brownish and texture of *green space* is greenish/brownish since images are captured under leaf-off canopy conditions. Hence, the separation of the respective features might be too difficult, particularly, since the information is smeared in the interpolation layers of PointNet++.

TUTZAUER et al. (2019) feed a multi-branch 1D CNN with many handcrafted contextual features for three different scales. The network is able to learn new features per scale. However, no hierarchical feature learning is possible. They report an OA of 74.76 % for tile C. This is better than PointNet (70.89 %) but worse than PointNet++ (80.62 %). This comparison is not entirely fair since PointNet/PointNet++ uses a smaller feature vector consisting only of *XYZ*, *normal vector* and *median HSV (face)*. Nevertheless, the achieved result emphasizes the superiority of PointNet++ that achieved significantly better results with significantly less handcrafted features.

However, when comparing training times we see that multi-branch 1D CNN is much faster than PointNet++ (~15 min vs. 15 h respectively) although using a multiple of features. The reason is that PointNet++ enables hierarchical learning, which includes feature encoding for iteratively increasing local regions, interpolation of encoded features, and MSG. MSG causes the main computational burden. For the 1D CNN, the heavy lifting is already done before training (i.e. calculation of contextual features). PointNet trains ~22 min and RF trains ~2 min.

TUTZAUER et al. (2019) also trained an RF on their feature vector. We achieve ~4% less OA with our RF. The worse performance is expected since we use fewer features and no contextual features at all.

4 Investigations on Radiometric Feature Quality

Section 3.2 shows that PointNet++ outperforms the other tested approaches. For this reason, we choose PointNet++ to analyze the importance of radiometric feature quality as a proxy for texture importance. Tab. 1 lists performance metrics for several feature vector configurations for tile A using PointNet++. Fig. 5 shows the respective predicted results for tile A.

Tab. 1 verifies our assumption that texture matters and increases performance on a global scale. Utilizing color information improves performance with respect to OA, mP, mR, and mIoU. Comparing the results of row 1 (geometry only) and row 2 (lowest quality of radiometric features) shows a performance gain at the global scale by approximately 5 % (mP, mR, mIoU). Furthermore, increased textural feature quality improves performance. The model trained on the feature vector with *median HSV (face)* (i.e. best radiometric feature quality) outperforms the model using only geometric features by ~10 % for mIoU and mR; mP is increased by roughly 5 %. By that, the configuration using the best radiometric feature quality (row 4) approximately doubles the achieved performance gain (for mIoU and mR) of the configuration using the lowest radiometric feature quality (row 2) with respect to the purely geometric configuration (row 1). This is quite astonishing since the improvement entirely depends on a substituted scalar radiometric feature. Thereby, the textural content of the entire face is used. However, high-resolution information is not used explicitly since we rely on one-dimensional feature vectors.

Tab. 1: Evaluation metrics for PointNet++ for Tile A using various feature vector configurations (*from top to bottom*: precision, recall, IoU [in %]). The last column lists mP, mR, and mIoU per feature vector configuration (the respective OAs (OAs with respect to surface area) from top to bottom: 84.13 % (85.98 %), 85.06 % (87.69 %), 85.61 % (88.23 %), and 85.68 % (88.56 %)). Geometric features consist of *XYZ* and *normal vector*. Radiometric feature quality increases from top to bottom.

Feature Vector Configuration	***Building Mass/ Facade***	***Roof***	***Impervious Surface***	***Green Space***	***Mid and High Vegetation***	***Vehicle***	***Chimney/ Antenna***	***Clutter***	**mP mR mIoU**
geometry only	70.31 **79.59** **59.57**	**86.26** 73.71 65.97	66.64 41.12 34.16	41.03 42.21 26.27	**90.49** 98.10 88.93	73.49 47.87 40.82	51.67 11.05 10.02	56.57 0.98 0.97	67.06 49.33 40.84
geometry, mean HSV (vertices)	70.44 78.34 58.96	84.70 75.02 66.06	**80.08** 47.53 42.50	**59.00** 31.85 26.08	89.02 **98.94** 88.18	71.60 **66.79** 52.80	56.96 31.37 25.36	65.43 3.72 3.65	72.15 54.20 45.45
geometry, median HSV (vertices)	73.10 74.27 58.33	81.16 **78.30** 66.26	76.07 **52.76** **45.25**	57.76 **45.06** **33.89**	90.39 98.56 **89.20**	72.81 63.43 51.28	**62.64** 30.48 25.79	**83.85** 4.28 4.24	**74.72** 55.89 46.78
geometry, median HSV (face)	**73.67** 74.96 59.12	86.00 76.16 **67.76**	76.41 52.41 45.11	56.52 40.66 30.97	89.86 98.81 88.90	**76.33** 64.11 **53.48**	49.56 **60.61** **37.49**	69.13 **11.16** **10.63**	72.19 **59.86** **49.18**

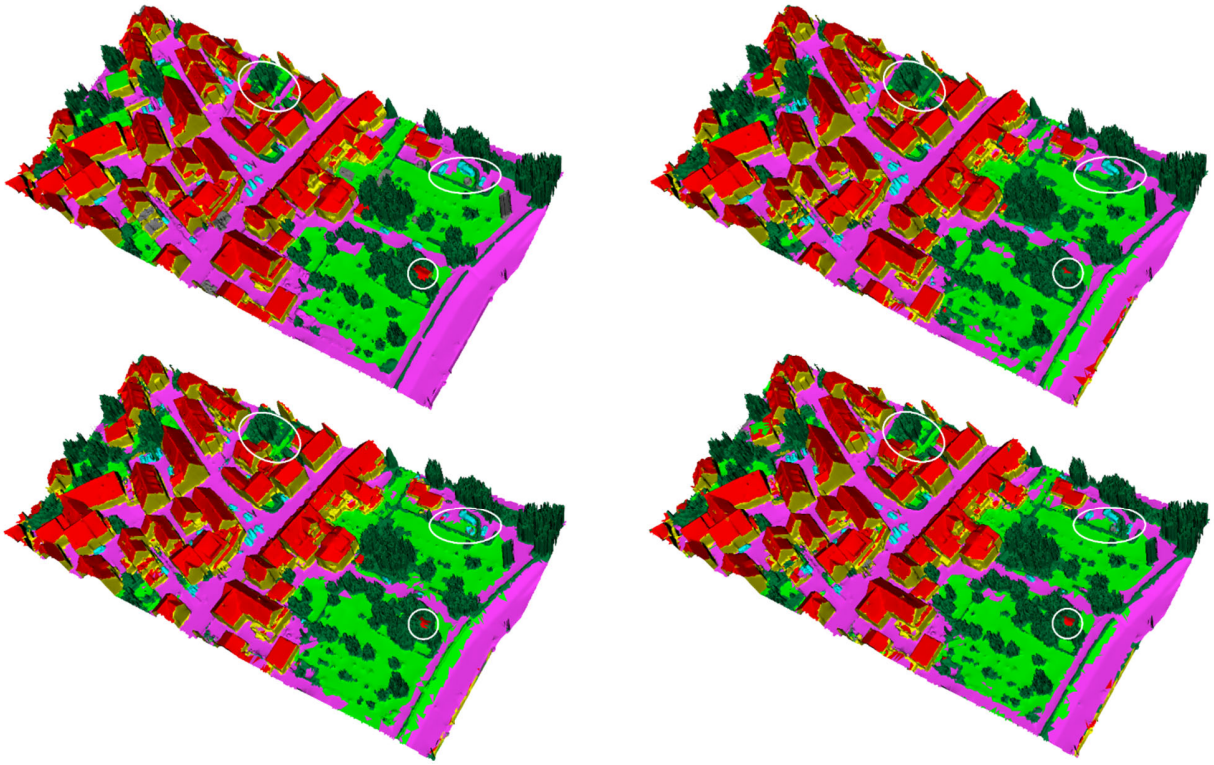

Fig. 5: Predictions of PointNet++ using geometric features and radiometric features of varying quality for tile A (cf. Tab. 1 for detailed numeric values). *From top left, to bottom right*: ground truth, prediction using *mean HSV (vertices)*, prediction using *median HSV (vertices)* and prediction using *median HSV (face)*. The marked areas show best results for *median HSV (face)*. Chimneys and antennas are best detected with the last configuration.

Median HSV (face) helps detect classes with low support. Per-class recall and IoU for *chimney/antenna* and *clutter* outperform the configurations using inferior radiometric feature quality to a large extent (up to factor 2 for class *chimney/antenna* or 3 for class *clutter*). This is the reason for a slightly better OA for this configuration. However, OA relies mainly on recall of the dominant class *mid and high vegetation*, which is almost on par for all configurations (~98-99 %). We deduce using per-face texture is beneficial for tackling imbalanced data sets.

Regardless of the radiometric feature quality, separation of classes *green space* and *impervious surface* improves significantly when color information is used. To give an example, 21 % of class *impervious surface* are mislabeled as class *green space* when only geometric features are used. The false positive rate drops by 15 % to 6 % when color information is incorporated. This is in accordance with the findings of (TUTZAUER et al. 2019). Furthermore, color information helps to detect buildings surrounded by high vegetation (cf. circle in Fig. 5). *Median HSV (face)* performs best since the colors are not smeared and better represent borders (cf. Fig 3).

To summarize, the best radiometric feature quality, i.e. *median HSV (face)*, globally achieves the best results. By this, using per-face color outperforms mimicked colored meshed point clouds that leverage per-vertex colors only. This shows the importance of available texture. Generally, the robust median features are better than the mean version. Nonetheless, on the class-level, the configuration using *mean HSV (vertices)* may outperform supposedly better configurations. For instance, it achieves the highest per-class precision for classes *impervious surface* and *green space*. At first glance, this may look counterintuitive. However, we claim that the 2.5D geometry of the mesh and occlusions in imagery are the main reasons why the feature configuration using *median HSV (face)* is not consistently the best (cf. ovals in Fig. 6). Both, "wrong" geometry and occlusions cause not-correctly textured faces. For instance, bushes in front of a building may occlude the facade. In that case, predicting class *mid and high vegetation* becomes more likely with increasing quality of the radiometric feature. The better radiometric feature makes it more likely to vote for a wrong class. To that extent, a better texture measure may dampen performance. In this context, the performance of textural features is naturally limited by occlusions, non-textured faces and the quality of texturing. The latter directly depends on the quality of geometric reconstruction and available imagery.

Regardless of the used feature vector configuration, prediction uncertainty increases at class borders (cf. Fig. 6). This is due to the interpolation of features in PointNet++. However, this is typical behavior in machine learning when vicinity information is incorporated. Furthermore, the effect of different color derivations (like sketched in Fig. 3) can be seen for the face that represents the garage door (cf. circle in Fig. 6).

In general, an increased color feature quality leads to more certain predictions. However, the correct predictions are more or less on par for the visualized fraction. For instance, for the garage door (cf. circle in Fig. 6), predictions using *median HSV (face)* are more certain than predictions based on *median HSV (vertices)*. However, we also see that improved color feature quality introduces uncertainty due to better color representation. For example, the dormers (cf. ovals in Fig. 6) become very uncertain for the configuration using *median HSV (face)*. In fact, a wrong label is predicted. Conversely, the other configurations achieved more stable results in this area. We assume that the better radiometric feature nudges the prediction to class *building mass*/*facade*.

Nevertheless, the dormers are annotated as *roof*. Thanks to this study, we reveal label noise on a fine-grained level or at least we detected a discussable class definition.

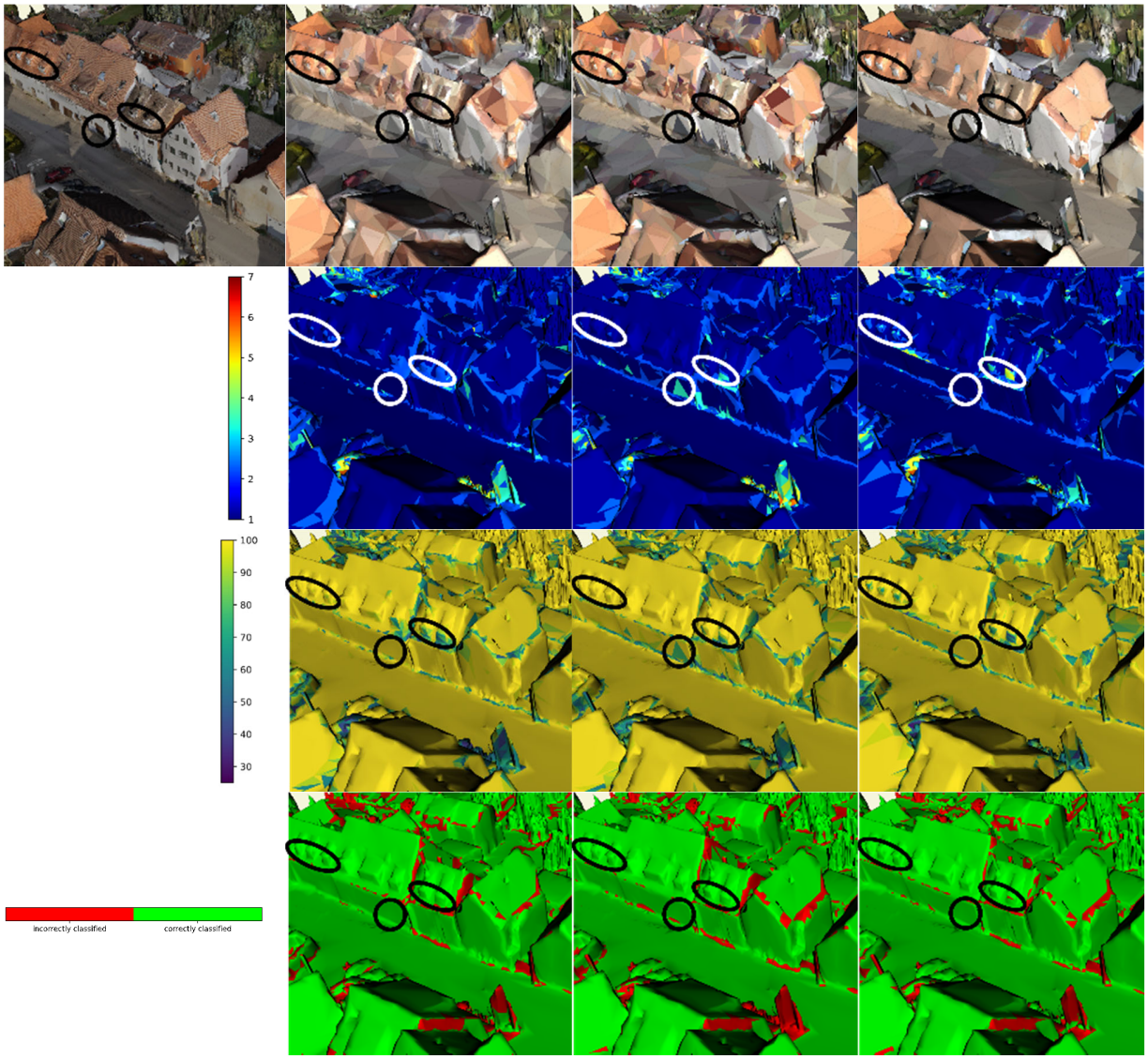

Fig. 6: Visualization of inconsistencies (*2nd row*) and maximum of averaged probabilities (*3rd row*) in aggregated predictions for PointNet++. First column depicts the textured view and the color schemes. The consecutive columns refer to feature vector configurations using *mean HSV (vertices)*, *median HSV (vertices)* and *median HSV (face)* respectively. The inconsistencies are depicted as number of predicted classes per face of overlapping mini-batches. The encircled area (garage door) shows the effect of different color derivations (like sketched in Fig. 3). The ovals show that improved color feature quality introduces uncertainty due to better color representation.

5 Conclusion & Outlook

We established a pipeline for semantic segmentation of textured meshes in urban scenes as generated from imagery and LiDAR data. Key idea is to represent each face by its COG associated with features. This enables using point-based classifiers. We compared several classifiers (RF, PointNet, PointNet++ and multibranch-1D CNN) with different capabilities of context mapping. PointNet++ performed best due to its capability of hierarchical feature learning (cf. section 3.2). Approximately 89 % of the surface area of a dedicated test tile has been predicted correctly.
In this study, we investigated in detail the importance/influence of radiometric feature quality. To this end, we derived radiometric features that utilize per-vertex or per-face color information. Regardless of the radiometric feature quality, separation of classes *green space* and *impervious surface* improves significantly. The false positive rate for class *green space* improves by 15 % when color information is used. We showed that increased radiometric feature quality improves performance on the global scale (cf. Tab. 1). Moreover, per-face color information improves detection rates for classes with small support (*vehicle*, *chimney*/*antenna*, *clutter*). We conclude that texture information matters and increases performance on a global scale. However, the study relies on one-dimensional radiometric features but still shows great performance gain. We claim that utilizing the inherent high-resolution information in the images further increases performance. Therefore, we plan to do semantic image segmentation in image space and back-project the predictions and/or extracted features to the mesh. Thereby, we leverage the entire available image content. Furthermore, we have seen the natural limit of texture due to errors in the geometric reconstruction or occlusions. Some faces carry wrong texture information (e.g. bushes may occlude facades) or do not carry texture at all. For this reason, we plan to make use of LiDAR features. The additional features may help to stabilize predictions and to detect vehicles more properly. Replacing our 2.5D mesh with a 3D mesh will inevitably improve results further on since texture quality depends on the geometric reconstruction. In particular, we plan to generate 3D meshes as generated from ALS data and multi-view stereo image matching.
We are aware of the fact that the investigations at hand are limited to one data set of a rather simple urban scene. We would like to extend investigations to more complex urban data captured with different sensors and flight configurations under different conditions (e.g. different seasons). However, such annotated reference data do not exist so far.

6 ACKNOWLEDGEMENTS

The urban mesh is a result of a research project in collaboration with the German Federal Institute of Hydrology (BfG) in Koblenz.

7 Bibliography

BOULCH, A., 2019: ConvPoint: Continuous Convolutions for Cloud Processing. CoRR (arXiv: 1904.02375).

BOULCH, A., LESAUX, B. & AUDEBERT, N., 2017: Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. Eurographics Workshop on 3D Object Retrieval, The Eurographics Association.

CRAMER, M., HAALA, N., LAUPHEIMER, D., MANDLBURGER, G. & HAVEL, P., 2018: Ultra-High Precision UAV-Based LiDAR and Dense Image Matching. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XLII**-1,115-120.

GEORGE,D., XIE, X. & TAM, G. K. L., 2017: 3D Mesh Segmentation via Multi-branch 1D Convolutional Neural Networks. CoRR (arXiv: 1705.11050).

GRAHAM, B., ENGELCKE, M. & VAN DER MAATEN, L., 2018: Semantic Segmentation with Submanifold Sparse Convolutional Networks. Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, 9224-9232.

KALOGERAKIS, E., AVERKIOU, M., MAJI, S. & CHAUDHURI, S., 2017: 3D Shape Segmentation with Projective Convolutional Networks. CoRR (arXiv: 1612.02808).

KÖLLE, M., LAUPHEIMER, D. & HAALA, N., 2019: Klassifikation hochaufgelöster LiDAR- und MVS-Punktwolken zu Monitoringzwecken. 39. Wissenschaftlich-Technische Jahrestagung der OVG, DGPF und SGPF, Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Band **28**, 692-701.

QI, C. R., SU, H., MO, K. & GUIBAS, L. J., 2017a: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. IEEE Conference on Computer Vision and Pattern Recognition, 652-660.

QI, C. R., YI, L., SU, H. & GUIBAS, L. J., 2017b: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. Proceedings of the 31st International Conference on Neural Information Processing Systems, 5105-5114.

ROTHERMEL, M., WENZEL, K., FRITSCH, D., HAALA, N., 2012: SURE: Photogrammetric Surface Reconstruction from Imagery. Proceedings LC3D Workshop, Berlin, 8, 2.

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. & SOERGEL, 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.

TUTZAUER, P., LAUPHEIMER, D. & HAALA, N., 2019: Semantic Urban Mesh Enhancement Utilizing a Hybrid Model. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences **IV**-2/W7, 175-182.

WINIWARTER, L., MANDLBURGER, G., SCHMOHL, S. & PFEIFER, N., 2019: Classification of ALS Point Clouds Using End-to-End Deep Learning. Journal of Photogrammetry, Remote Sensing and Geoinformation Science **87**, 75-90.