

SEMANTIC SEGMENTATION OF URBAN TEXTURED MESHES THROUGH POINT SAMPLING

Grégoire Grzeczakowicz^{1,2,*}, Bruno Vallet¹

¹ LASTIG, Univ Gustave Eiffel, IGN, ENSG

² Direction Générale de l'Armement
Gregoire.Grzeczakowicz@ign.fr

KEY WORDS: Mesh, Semantic Segmentation, Point Sampling.

ABSTRACT:

Textured meshes are becoming an increasingly popular representation combining the 3D geometry and radiometry of real scenes. However, semantic segmentation algorithms for urban mesh have been little investigated and do not exploit all radiometric information. To address this problem, we adopt an approach consisting in sampling a point cloud from the textured mesh, then using a point cloud semantic segmentation algorithm on this cloud, and finally using the obtained semantic to segment the initial mesh. In this paper, we study the influence of different parameters such as the sampling method, the density of the extracted cloud, the features selected (color, normal, elevation) as well as the number of points used at each training period. Our result outperforms the state-of-the-art on the SUM dataset, earning about 4 points in OA and 18 points in mIoU.

1. INTRODUCTION

Textured 3D meshes of urban areas are becoming more and more common and of ever better quality. A mesh defines a continuous 3D surface (2-manifold) through a set of triangles sharing edges and vertices, which representation is more appropriate than its usual alternatives:

- Digital Surface Models (DSM) represent the geometry of the scene by an elevation (z) sampled on a regular horizontal grid. This representation is simple and efficient but discards information on vertical surfaces (walls, facades, ...) and overhangs (bridges, tunnels, ...).
- Point clouds simply consist of discrete samples of the geometry without recovering the continuous nature of the real surface.

Meshes are light and adaptive data structures where large areas can be represented by few large faces while details can be captured by a larger number of small faces. Meshes can be textured by creating a bijection between subsets of faces and their projection in 2D images called textures (Waechter et al., 2014), allowing them to hold a radiometric information in the same way that DSMs (a color per pixel) and point clouds (a color per point). Textured meshes are now becoming the standard output of the photogrammetric reconstruction pipeline, as the preferred way to gather radiometric and geometric data of an urban scene (Laupheimer and Haala, 2021). Last but not least, textured meshes are the main representation used by the video games and animation industries and numerous techniques have been developed to process and visualize them. For all these reasons the semantic segmentation of textured meshes have received a growing interest. While state of the art algorithms currently rely on random forests and Markov Random Fields, the most recent algorithms leverage graph convolutions and deep neural networks. Semantic segmentation consists in associating to each piece of data the semantic class to which it belongs

(for example, in the urban context, *buildings*, *ground* or *vegetation*). While these pieces of data are natural for DSMs (pixels) and point clouds (points), different choices can be made for textured meshes: vertices, triangles or texels (pixels of the texture images), which will be discussed in the paper. Semantic segmentation of urban meshes raises open questions in terms of adapted network architectures for learning and poses some important challenges in terms of scaling. Moreover, to the best of our knowledge, radiometric information is only weakly exploited at the moment, most algorithms simply extracting features from the radiometric information per face, rather than using the whole texture. In this paper, we propose to use the textured mesh to sample a point cloud of very good quality with colors and normal in order to use a high-performance semantic segmentation algorithm on point clouds. We investigate different techniques for sampling points on a textured mesh and compare the results obtained using the KPConv algorithm (Thomas et al., 2019) on the resulting point clouds.

2. RELATED WORK

2.1 Mesh Semantic Segmentation

The pioneer works on mesh semantic segmentation were performed using handcrafted features (geometrical and radiometric) with a Random Forest classifier and Markov Random Fields (Rouhani et al., 2017). The selected features are elevation, planarity, verticality, average color, standard deviation, and color distribution in the HSV color space. (Tutzauer et al., 2019) use a 1x1 CNN to learn one features vector per face from multi-scale features aggregation from neighbors faces. These learned features are combined with handcrafted features to feed a Random Forest classifier. (Laupheimer et al., 2020) use the same approach to compute per face features vectors, but then use PointNet++ (Qi et al., 2017b) to segment the mesh view as a point cloud where each face is a point.

As a mesh represents a 2D surface, it is a 2-manifold which locally resembles (2D) Euclidean space. This characteristic

* Corresponding author

can be exploited to use standard CNNs, by locally parameterize the surface to 2D (Masci et al., 2015, Maron et al., 2017). (Tatarchenko et al., 2018) introduce tangent convolution, where a small neighborhood around each point is used to reconstruct the local function upon which convolution is applied.

As the structure of a mesh carries two graphs (the primal, formed by vertices connected by edges and the dual, formed by faces connected by dual edges), classical Graph Convolutional Neural Networks can be adapted to work on a mesh. (Verma et al., 2018) implement the pooling operation through a generic graph clustering algorithm (Dhillon et al., 2007). (Ranjan et al., 2018) use classic mesh-simplification techniques for surface approximation (Garland and Heckbert, 1997) to define their pooling operator. (Milano et al., 2020) designed a primal-dual framework GCN named PD-MeshNet. PD-MeshNet takes the edges and faces of the 3D meshes as the features of the graph nodes, and then adds an attention mechanism to achieve classification and segmentation. MeshCNN (Hanocka et al., 2019) introduce a mesh-specific convolution and pooling layers that are applied over the edges of a mesh. (Knott and Groenendijk, 2021) add radiometric features to face inspired by (Rouhani et al., 2017).

2.2 Point Cloud Semantic Segmentation

Several approaches are used to encode point clouds for semantic segmentation. The simplest one is to use voxels (cubic volume elements of a regular 3D grid). The interest is that voxels generalize the notion of pixel to 3D, thus allowing for a trivial extension of Convolutional Neural Networks (CNNs). The first to use this approach are (Wu et al., 2015) with a CNN that performs shape classification. Even if this approach is theoretically simple, it has a very large memory footprint because a 3D point cloud is very sparse when discretized on voxels. Different approaches have been used to exploit this sparsity to use reduced space representation (FPNN (Li et al., 2016), OctNet (Riegler et al., 2017), O-CNN (Wang et al., 2017), SparseConvNet (Gramham et al., 2018)).

Recent progress has been made to use a point cloud directly as input to a neural network. (Qi et al., 2017a) designed PointNet, a solution to the point order invariance problem that uses a 1x1 convolution followed by global max pooling. PointNet++ (Qi et al., 2017b) introduces a partition of points to better consider local structures. This approach has inspired many sophisticated neural modules to learn per-point local features. These modules can be generally classified as neighboring feature pooling (So-net (Li et al., 2018), PointWeb (Zhao et al., 2019), ShellNet (Zhang et al., 2019)), graph message passing (LocalSpecGCN (Wang et al., 2018), GAC (Wang et al., 2019a), EdgeConv (Wang et al., 2019b), ClusterNet (Chen et al., 2019)), kernel-based convolution (SPLATNet (Su et al., 2018), PointConv (Wu et al., 2019), A-CNN (Komarichev et al., 2019), KPConv (Thomas et al., 2019), LatticeNet (Rosu et al., 2020)) and attention-based aggregation (ShapeContextNet (Xie et al., 2018), PCAN (Zhang and Xiao, 2019), GSS (Yang et al., 2019)).

Nevertheless, most of these approaches are often unsuitable for processing the large-scale point clouds arising from high resolution LiDAR scans or dense matching on urban areas. Super Point Graph (SPG) (Landrieu and Simonovsky, 2018) introduces the notion of superpoints, an equivalent of superpixels for point clouds to partition the point cloud into subsets of points that share the same semantic class, then semantize the resulting

SPG which size is much more reasonable than the initial point cloud with graph convolutions. RandLA-Net (Hu et al., 2020) use both random point sampling (for memory and computation efficiency) and a novel local feature aggregation module that progressively increases the receptive field for each point (for preserving geometric details). These approaches allow the processing of much larger point clouds.

3. METHOD

3.1 Mesh Sampling with points

While for image and point cloud semantic segmentation, the “piece of data” to semantize is natural (pixels and points), the question is more open for textured meshes: do we look for a semantic labelling of vertices, triangles or texels? In fact we propose another alternative: semantizing discrete points sampling the mesh. This has multiple benefits:

- We can leverage the abundant literature on point cloud semantic segmentation on the resulting samples
- By sampling more or less points, we can control the performance vs quality compromise.
- By sampling points regularly we can overcome typical limitations of point cloud semantic segmentation that are sensitive to sampling anisotropy and strongly varying point densities.
- The mesh gives an unambiguous normal at each point.
- The texture gives a color at each point.
- The labels obtained per point samples can easily be interpolated to the vertices, faces or texels.

In this paper, we study two types of sampling: texel sampling based on the texture and a random but homogeneous sampling called Poisson disk sampling.

A first way to obtain a homogeneous sampling is to sample the points on a regular grid defined on the surface of the mesh. For this we can take advantage of the texture of our mesh which defines a natural grid globally homogeneous on the surface of the mesh (Waechter et al., 2014). This texture grid has the advantage of being continuous between some faces that share the

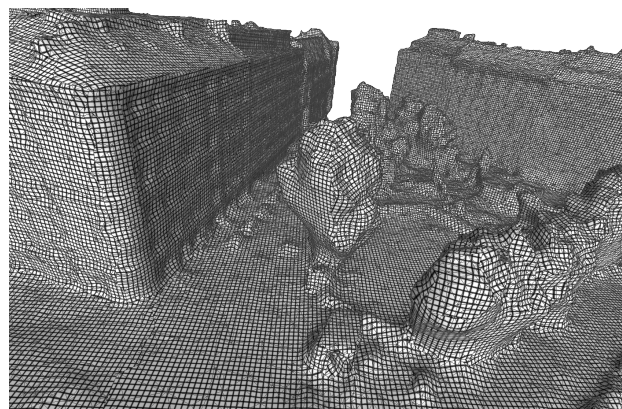
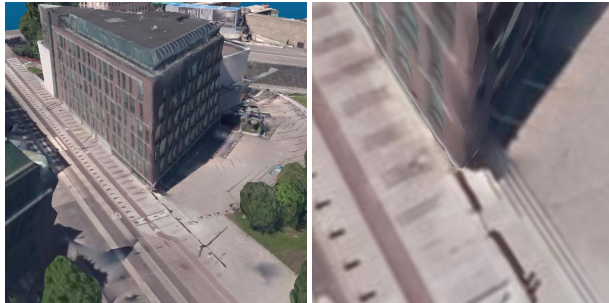
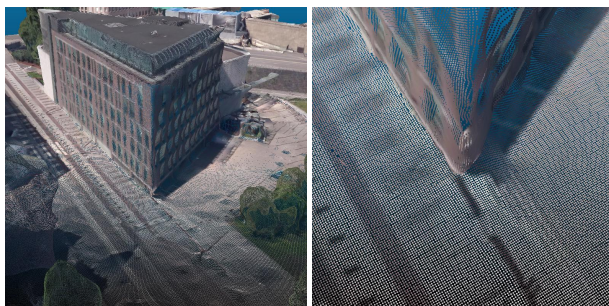


Figure 1. Texture grid on the mesh.

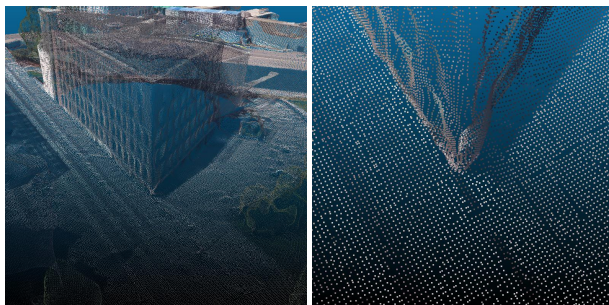
same texture patch (Figure 1). Sampling one point at the center of each texel, which we call texel sampling, is moreover perfectly adapted to the quantity of texture information as each pixel is converted into a point (Figure 2b). By adapting the resolution of the texture used, we can choose the density of the point cloud produced (Figure 2c). We have introduced a parameter s which represents the ratio of the retained pixel size to the original size.



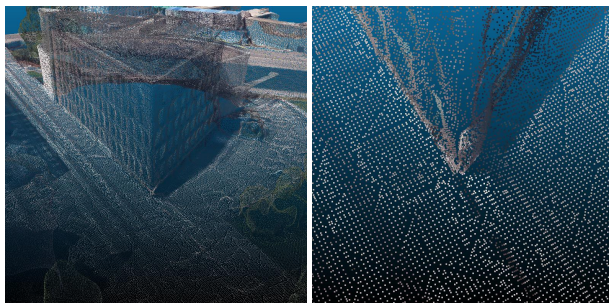
(a) Mesh



(b) Texel sampling ($s = 1.0$)



(c) Texel sampling ($s = 2.0$)

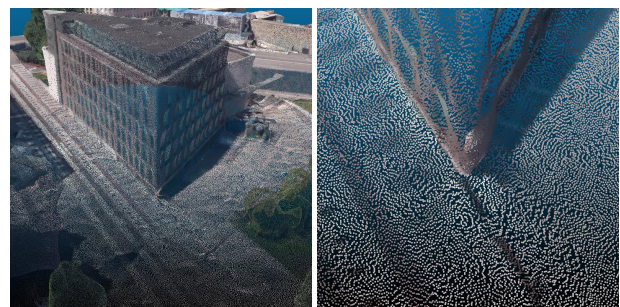


(d) Texel sampling ($s = 1.0$) + grid sub-sampling ($g = 0.2$)

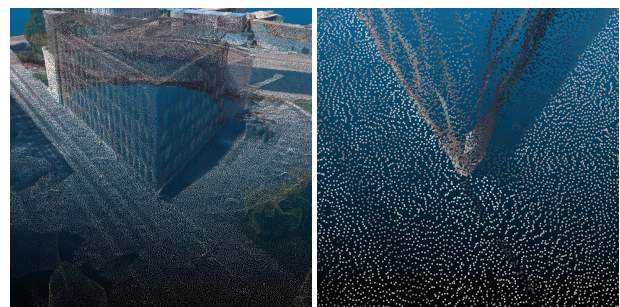
Figure 2. Sampling methods comparison.

A second approach is to perform a random but homogeneous sampling on the surface. A well known solution to this prob-

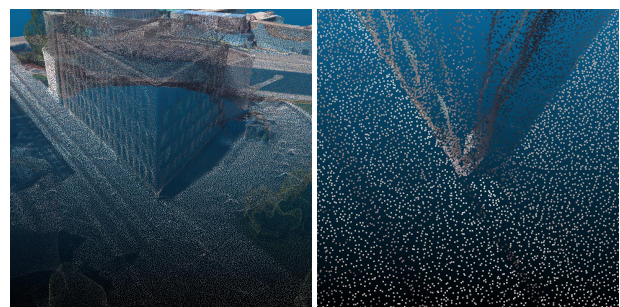
lem is the Poisson disk sampling algorithm. A Poisson disk sampling is a set of points such that all samples are at least at a distance r apart for some user-supplied density parameter r (Figures 2e and 2f). The naïve rejection-based approach for generating Poisson disk samples, dart throwing, is impractically inefficient and requires a stop condition. Over the years, a large number of methods for generating Poisson disk distributions have been proposed (Lagae and Dutré, 2008). We choose to use the Constrained Sample-based Poisson disk Sampling developed specifically for meshes and integrated in Meshlab (Corsini et al., 2012). The main idea of the algorithm is to sample the mesh with uniformly random points and then to sub-sample this set of points by choosing the points to keep one by one while removing from the set the points whose distance to the previously chosen points is less than r . The sampling stops when all the initial points have been either chosen or removed.



(e) Poisson disk sampling ($r = 0.1$)



(f) Poisson disk sampling ($r = 0.15$)



(g) Poisson disk sampling ($r = 0.1$) + grid sub-sampling ($g = 0.2$)

Figure 2. Sampling methods comparison.

Whether it is pixel sampling or Poisson disk sampling, these two methods allow to adapt the density of the point cloud produced (by adapting the resolution of the texture used or by adapting the distance r). Nevertheless, the density of the produced cloud can also be adapted by performing a sub-sampling after the cloud generation. (Hu et al., 2021) has compared grid sub-sampling and random sub-sampling and obtained better results

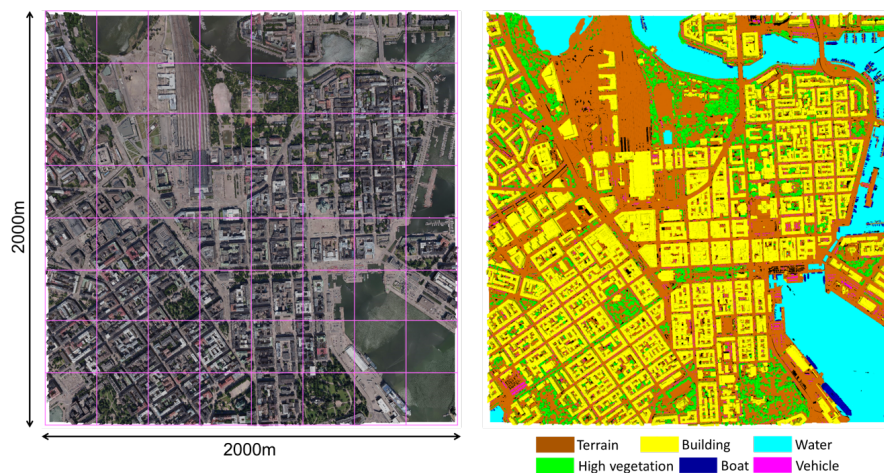


Figure 3. SUM dataset (Gao et al., 2021).

with grid sub-sampling, thus we tested grid sub-sampling (Figures 2d and 2g). By choosing the grid step g , we can adapt the density of the output point cloud. We have chosen to randomly keep one point per cell of the grid, rather than generating a centered point per cell that would average the points in the cell. This allows us to obtain a greater geometric fidelity to the initial mesh. If the initial sampling is not random, an aliasing effect may appear after sub-sampling, as in our case with texel sampling (Figure 2d).

3.2 Feature selection

The textured mesh sampling allows us to associate multiple input features to each point. We have studied the influence on the segmentation of several features such as the color, normal or the elevation. For the normal, we propose two alternatives: the face normal or an interpolation of the vertices normal. The face normal is directly the normal of the triangle on which the point is sampled, oriented towards the outside of the mesh. The normal at a vertex of a mesh is usually defined as the average of the normals of the adjacent faces weighted by their area. If this sum is null, we defined the normal of the vertex as the sum of the director vectors of the edges adjacent to this vertex, as these cases occur when two faces of the mesh fold in on themselves. In both cases, this vector is normalized to obtain a unit vector. The normal of a point is then defined as the average of the normals at the vertices of the face to which it belongs, weighted by the inverse distance to these vertices. The face normals are equal for all points on the same face, thus discontinuous between faces, while the interpolated normals are continuous even at the edges and vertices.

To calculate the elevation of a point, we can use two methods. The first is simpler, but less accurate. It consists of calculating the difference between the elevation of the point and the minimum elevation of the points contained in a neighborhood large enough to contain ground and small enough that the ground can be considered as globally flat in this neighborhood. The difficulty is to define this neighborhood, if it exists, and that sometimes points may be located below the surface of the ground (for example in the case of a river). The second method consists of deriving a digital terrain model (DTM) from the mesh (Beumier and Idrissa, 2016) and then determining the elevation of each point with respect to this model. The main problem with this method is that deriving a DTM from the mesh is a non-

trivial problem. We therefore chose in the case of our study to test only the elevation obtained using the first method.

4. EVALUATION

4.1 Dataset

While a large number of textured meshes of urban environments are publicly available, very few have been labeled to serve as training, validation and test data for semantic segmentation. Indeed, urban environment meshes are often quite large (they can contain from one to several tens of millions of faces according to the covered surface) and thus require many hours of work to be labelled manually. To evaluate our different semantic segmentation methods, we used the SUM dataset (Gao et al., 2021) which is a textured mesh of the city of Helsinki covering an area of 4 km^2 (Figure 3). It is a semi-automatic labeling of the raw Helsinki 3D dataset (City of Helsinki, 2017): a textured mesh that was generated in 2017 with ContextCapture (Bentley Systems, 2016) from oblique aerial images with a ground sampling distance (GSD) around 7.5 cm which covers 12 km^2 .



Figure 4. Tile distribution between set.

The source images have three colour channels (i.e., red, green, and blue) and the triangular faces are divided into 6 categories: terrain, vegetation, building, water, vehicle and boat. Ambiguous regions (which account for about 2.6% of the total mesh surface area), such as shadowed regions or distorted surfaces,

are labelled as unclassified. The entire mesh is split into 64 tiles, and each of them covers about 250 m². 40 randomly selected tiles are used as training data, 12 as validation data and 12 as test data (Figure 4).

For each of the seven semantic categories, we computed the total area in the training, validate and test dataset to show the class distribution. As shown in Figure 5, the classes are well distributed among the three sets, but they are not at all balanced. The Vehicle and Boat classes represent only less than 3 % of the surface, while the Building class covers more than 50 % of the surface.

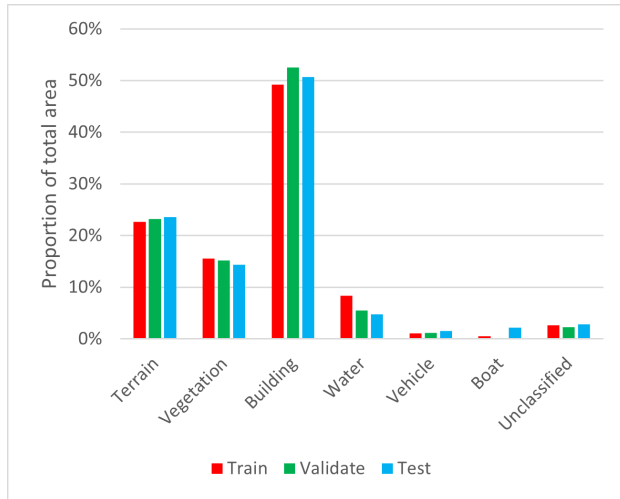


Figure 5. Surface distribution between classes for each set.

4.2 Evaluation method

To compare our different sampling methods and the importance of the selected features, we used the rigid KPConv model from (Thomas et al., 2019) to semantize the sampled point clouds, trained during 100 epochs. We chose KPConv because it was among those obtaining better results on the SUM dataset (Gao et al., 2021) and because the model was already implemented in Torch Points 3D (Chaton et al., 2020), the framework we chose to perform the semantic segmentation of our sampled point clouds. We adapted the model to sum the logit (the output of the last layer before the softmax) of each point belonging to the same face and thus obtain a segmentation by face instead of by point, since our mesh is labeled at each face.

As each tile contains on average 4 million points, it is not possible to train on a complete tile at each stage. The model therefore learns at each time from 1024 subsets of points distributed in batches of 16. As recommended by (Hu et al., 2021), to draw this subset we choose a point and its k nearest neighbors. This allows to obtain a fixed number of points per batch, whereas this would not have been the case if we had drawn all the points present in a sphere of given size. We tried several values for k , which affects the model's receptive field during learning.

As explained in Section 4.1, our classes are quite unbalanced. To counteract this, when we draw a subset at random, the center point is not drawn uniformly among all points, but inversely proportional to the frequency of each class. Thus, on average, each class is visited as much as the others. This is only true for the center point of each subset, so our draw is not perfectly balanced, but comes close.

The validation set is used to select the best model obtained over the 10 epochs. The validation metrics are also obtained from a draw of 1024 subsets in each epoch. However, the test operation is performed on all the test set. It is split into subsets of size k which overlap so that all points of the test tiles belong to at least one sphere. The logit obtained at the output of the model are then averaged for each point that belongs to more than one sphere.

To evaluate the results, we use three classical metrics: Overall Accuracy (OA), Intersection over Union (IoU) and class accuracy (Acc). The first one is defined for each class, while the last two evaluate the whole model. IoU and Acc compensates for different class frequencies as opposed to OA that does not balance different class frequencies giving higher influence on large classes.

If n_{ij} is the number of faces from ground-truth class i label as j by our network,

The IoU for class i is:

$$\text{IoU}_i = \frac{n_{ii}}{n_{ii} + \sum_{j \neq i} n_{ij} + \sum_{k \neq i} n_{ki}} \quad (1)$$

The OA is:

$$\text{OA} = \frac{\sum_i n_{ii}}{\sum_j \sum_k n_{jk}} \quad (2)$$

The Acc for class i is:

$$\text{Acc}_i = \frac{n_{ii}}{\sum_j n_{ij}} \quad (3)$$

The average with respect to each class of IoU and Acc is noted mIoU and mAcc.

All our codes are public. The sampling program is available at github.com/umrlastig/Mesh-2-Point-Cloud and the semantization part has been implemented with the Torch Points 3D framework (Chaton et al., 2020) and is available at github.com/umrlastig/torch-points3d. Our training statistics and results can be checked on WandB at wandb.ai/ggrzeczkwicz/sum-semantic-segmentation.

We performed all our experiments on a computer equipped with 184 GB of RAM, 6 CPUs and a Tesla V100 GPU with 32 GB of RAM. Each training session takes an average of 8 hours.

4.3 Results

We obtain a maximum mIoU with a Poisson disk sampling, $r = 0.4$, only RGB as feature and 10240 points per subset. In our results, Vehicle and boats are the main sources of errors as they are much less represented in the dataset and are smaller objects that suffer more from under-sampling. While water is also poorly represented, it is a much more identifiable object, which explains the much better results. From this result we conduct an ablation study (Table 1) to quantify the influence of each parameter.

We observe that the most important feature is the color whose absence greatly lowers the quality of the results. This result is quite logical as some surfaces such as water and ground are distinguishable only by radiometric features. The contribution of

	Density (pts/m ²)	IoU							mIoU (%)	OA (%)	mAcc (%)
		Terrain (%)	High Vegetation (%)	Building (%)	Water (%)	Vehicle (%)	Boat (%)				
SUM + KPConv (Gao et al., 2021)	10	86.5	88.4	92.7	77.7	54.3	13.3	68.8	93.3	73.7	
SUM (Gao et al., 2021)	10	83.3	90.5	92.5	86.0	37.3	7.4	66.2	93.0	70.6	
PDS(0.4) with RGB and 10240 pts/sphere	11.7	90.6	93.5	96.9	96.9	71.2	68.0	86.2	97.0	91.6	
With face normal	11.7	91.0	94.3	96.8	96.9	71.8	64.7	85.9	97.1	91.1	
With normal	11.7	89.9	94.4	96.4	97.0	70.5	61.6	85.0	96.9	90.9	
With 5120 pts/sphere	11.7	89.0	91.8	96.1	97.2	69.6	62.4	84.3	96.6	92.2	
With PDS(0.2) and 20480 pts/sphere	47.0	89.9	94.0	96.6	97.0	57.2	63.0	82.9	96.9	87.4	
With TS(4.0)	15.4	91.5	93.2	96.4	97.9	68.7	48.5	82.7	97.0	90.7	
With elevation	11.7	85.9	91.5	94.3	96.1	55.2	47.1	78.4	95.4	81.5	
Without RGB	11.7	86.4	88.4	94.7	91.9	59.4	34.9	76.0	95.0	84.3	

Table 1. Comparison of various sampling and feature selection. PDS(r) = Poisson disk sampling, TS(s) = texel sampling.

the elevation seems to decrease the performance, especially on small objects such as vehicles and boats. This can be caused by several phenomena. First, as we discuss in 3.2, the elevation we have chosen is the simplest, but not necessarily the most rigorous. Also, when we look at the training statistics, the elevation seems to confuse our network more than anything else, causing our loss to decrease much more slowly. In the future, it might be worth trying to integrate a true DTM, as related work on urban classification (Rouhani et al., 2017, Tutzauer et al., 2019, Laupheimer et al., 2020) has shown that elevation is generally among the most important features for classification. The contribution of the normals is marginal, they improve the IoU of some classes while decreasing that of others, causing a slight decrease in mIoU.

Reducing the number of points per sphere by two does not seem to decrease the quality of the results significantly (about two IoU points in each class) while it allows to divide the learning time by two (4 hours instead of 8). This could be interesting for GPUs with less RAM. However, it should be noted that this increases the number of sub-samples to be classified in the test phase. A priori, increasing the density of the point cloud does not lead to better results either. However, we must take into account that it is the receptive field of each point that could be decisive, and that despite the fact that we have doubled the size of the subsamples, it is by $2^3 = 8$ that we must multiply the size of the subsamples to keep a receptive field of the same dimensions when we divide by two the distance between the sampling points. For a comparable density, texel sampling seems to obtain results of the same order of magnitude, but on average slightly worse. This sampling being slightly less uniform than Poisson disk sampling and anisotropic, it could introduce a slight bias that degrades the results. Grid subsampling does not have a measurable effect, at equal density, in our results.

We also compare our results to (Gao et al., 2021) who established a first benchmark of their dataset. They used the same algorithm as for their automatic labeling and also performed a point sampling and then used different semantic segmentation algorithms. In this benchmark, KPConv obtains the best results (which partly explains our choice to use this method), but their results are far below the ones that we obtain (cf Table 1). As this benchmark uses a comparable density of points (10 pts/m²) and also Poisson disk sampling, the difference may be explained by a different setting of the semantization algorithm as the chosen hyper-parameters are not publicly available. Their results on the boat class may also suggest an unresolved class

imbalance problem.

The qualitative results (Figure 6) give us a better idea of the sources of error. The errors appear at the interface between two classes. It should be noted that the ground truth is labeled at the face level, but sometimes the boundary between two classes is not exactly at the interface between two faces, which can explain very slight differences. We see here an advantage of classifying at the point level, allowing to classify the mesh at the texel level and not at the face level. Sometimes it is a real error and sometimes the labeling is more delicate. In the first example, a boat that is too close to the water is classified as a building by mistake. In the second example, the terrace of the monument is labeled as a building when it is classified as a ground. In the third example, there is less error than in the first example with the boat, but there are still some concerns. The fourth example shows us a roof falsely classified as ground, which makes explicit the importance of having a good elevation feature. Finally, the last example is a tent in a parking lot that has been classified as a vehicle.

5. CONCLUSION AND PERSPECTIVES

Textured meshes are becoming an increasingly popular representation combining the 3D geometry and radiometry of real scenes. When analysing the underlying scene, and in particular its semantics, it is still an open question to know if the best strategy is to semantize the pixels of the input images, the 3D points of the photogrammetric point cloud, both at the same time in a multi-modal approach (Robert et al., 2022), or through the textured mesh as proposed in this paper. However we believe that the raw (images) or intermediate (point clouds) products might often be lost or simply discarded by the production pipeline, such that being able to semantize the textured mesh directly has an inherent benefit.

We investigated other alternatives to textured mesh semantic segmentation, in particular leveraging graph convolutions on the primal or dual structure of the mesh, but the patch structure of the texture makes it always complicated to integrate textual information, more accurate than just the average texel of the face, at node level and our best effort could not reach the same level of quality as with the much simpler sampling approach. The main perspective of this work is to extend it to instance segmentation (separating instances of adjacent objects such as buildings and vehicles) and structured reconstruction (going from a textured mesh to a LoD2 city model).

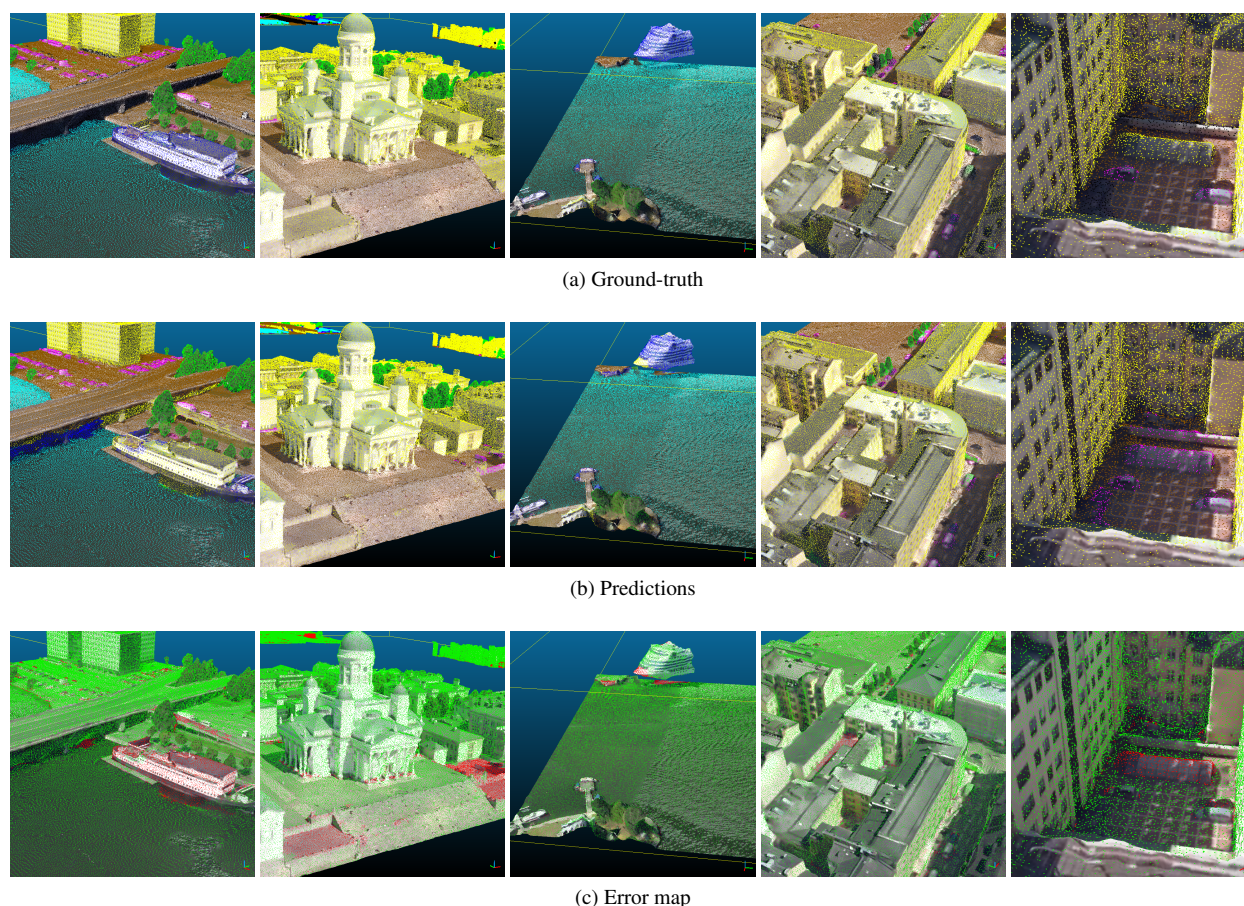


Figure 6. Results obtain with Poisson disk sampling, $r = 0.4$, RGB and 10240 pts/sphere.

REFERENCES

- Bentley Systems, 2016. ContextCapture Software. bentley.com/products/brands/contextcapture (1 January 2022).
- Beumier, C., Idrissa, M., 2016. Digital terrain models derived from digital surface model uniform regions in urban areas. *International Journal of Remote Sensing*, 37(15), 3477–3493.
- Chaton, T., Chaulet, N., Horache, S., Landrieu, L., 2020. Torchpoints3d: A modular multi-task framework for reproducible deep learning on 3d point clouds. *2020 International Conference on 3D Vision (3DV)*, 1–10.
- Chen, C., Li, G., Xu, R., Chen, T., Wang, M., Lin, L., 2019. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4989–4997.
- City of Helsinki, 2017. 3D city model of the city of Helsinki. City of Helsinki, Urban Environment Department, Services and Permits, Urban Measurement Services, Finland. CC BY 4.0. kartta.hel.fi.
- Corsini, M., Cignoni, P., Scopigno, R., 2012. Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6), 914–924.
- Dhillon, I. S., Guan, Y., Kulis, B., 2007. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 1944–1957.
- Gao, W., Nan, L., Boom, B., Ledoux, H., 2021. SUM: A benchmark dataset of Semantic Urban Meshes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 179, 108–120.
- Garland, M., Heckbert, P. S., 1997. Surface simplification using quadric error metrics. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97*, ACM Press/Addison-Wesley Publishing Co., USA, 209–216.
- Graham, B., Engelcke, M., Maaten, L. v. d., 2018. 3d semantic segmentation with submanifold sparse convolutional networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9224–9232.
- Hanocka, R., Hertz, A., Fish, N., Giryas, R., Fleishman, S., Cohen-Or, D., 2019. MeshCNN: A Network with an Edge. *ACM Trans. Graph.*, 38(4), 12.
- Hu, Q., Yang, B., Khalid, S., Xiao, W., Trigoni, N., Markham, A., 2021. Towards semantic segmentation of urban-scale 3d point clouds: A dataset, benchmarks and challenges. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4975–4985.
- Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A., 2020. Randa-net: Efficient semantic segmentation of large-scale point clouds. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11105–11114.
- Knott, M., Groenendijk, R., 2021. Towards Mesh-based Deep Learning for Semantic Segmentation in Photogrammetry. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2021, 59–66.
- Komarichev, A., Zhong, Z., Hua, J., 2019. A-cnn: Annularly convolutional neural networks on point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7413–7422.

- Lagae, A., Dutré, P., 2008. A Comparison of Methods for Generating Poisson Disk Distributions. *Computer Graphics Forum*, 27(1), 114–129.
- Landrieu, L., Simonovsky, M., 2018. Large-scale point cloud semantic segmentation with superpoint graphs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4558–4567.
- Laupheimer, D., Haala, N., 2021. Juggling with representations: On the information transfer between imagery, point clouds, and meshes for multi-modal semantics. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, 55–68.
- Laupheimer, D., Shams Eddin, M. H., Haala, N., 2020. The importance of radiometric feature quality for semantic mesh segmentation. *DGPF annual conference, Stuttgart, Germany. Publikationen der DGPF*, 29.
- Li, J., Chen, B. M., Lee, G. H., 2018. So-net: Self-organizing network for point cloud analysis. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9397–9406.
- Li, Y., Pirk, S., Su, H., Qi, C. R., Guibas, L. J., 2016. Fpnn: Field probing neural networks for 3d data. D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (eds), *Advances in Neural Information Processing Systems*, 29, Curran Associates, Inc.
- Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V. G., Lipman, Y., 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph.*, 36(4), 10.
- Masci, J., Boscaini, D., Bronstein, M. M., Vandergheynst, P., 2015. Geodesic convolutional neural networks on riemannian manifolds. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 832–840.
- Milano, F., Loquercio, A., Rosinol, A., Scaramuzza, D., Carlone, L., 2020. Primal-dual mesh convolutional neural networks. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (eds), *Advances in Neural Information Processing Systems*, 33, Curran Associates, Inc., 952–963.
- Qi, C. R., Su, H., Kaichun, M., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 77–85.
- Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413*.
- Ranjan, A., Bolkart, T., Sanyal, S., Black, M. J., 2018. Generating 3d faces using convolutional mesh autoencoders. V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (eds), *Computer Vision – ECCV 2018*, Springer International Publishing, Cham, 725–741.
- Riegler, G., Ulusoy, A. O., Geiger, A., 2017. Octnet: Learning deep 3d representations at high resolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6620–6629.
- Robert, D., Vallet, B., Landrieu, L., 2022. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Rosu, R. A., Schütt, P., Quenzel, J., Behnke, S., 2020. LatticeNet: Fast Point Cloud Segmentation Using Permutohedral Lattices. *Proceedings of Robotics: Science and Systems*, Corvallis, Oregon, USA.
- 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.
- Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., Kautz, J., 2018. Splatnet: Sparse lattice networks for point cloud processing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2530–2539.
- Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.-Y., 2018. Tangent convolutions for dense prediction in 3d. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3887–3896.
- Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L., 2019. Kpconv: Flexible and deformable convolution for point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 6410–6419.
- 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. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2598–2606.
- Wachter, M., Moehrl, N., Goesele, M., 2014. Let there be color! large-scale texturing of 3d reconstructions. D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (eds), *Computer Vision – ECCV 2014*, Springer International Publishing, Cham, 836–850.
- Wang, C., Samari, B., Siddiqi, K., 2018. Local spectral graph convolution for point set feature learning. V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (eds), *Computer Vision – ECCV 2018*, Springer International Publishing, Cham, 56–71.
- Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J., 2019a. Graph attention convolution for point cloud semantic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10288–10297.
- Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., Tong, X., 2017. O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.*, 36(4), 11.
- Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., Solomon, J. M., 2019b. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5).
- Wu, W., Qi, Z., Fuxin, L., 2019. Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9613–9622.
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1912–1920.
- Xie, S., Liu, S., Chen, Z., Tu, Z., 2018. Attentional shapecontextnet for point cloud recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4606–4615.
- Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., Tian, Q., 2019. Modeling point clouds with self-attention and gumbel subset sampling. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3318–3327.
- Zhang, W., Xiao, C., 2019. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12428–12437.
- Zhang, Z., Hua, B.-S., Yeung, S.-K., 2019. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1607–1616.
- Zhao, H., Jiang, L., Fu, C.-W., Jia, J., 2019. Pointweb: Enhancing local neighborhood features for point cloud processing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5560–5568.