SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks

Alexandre Boulch, Joris Guerry, Bertrand Le Saux, Nicolas Audebert
ONERA - The French Aerospace Lab, FR-91761 Palaiseau, France

Abstract

In this work, we describe a new, general, and efficient method for unstructured point cloud labeling. As the question of efficiently using deep Convolutional Neural Networks (CNNs) on 3D data is still a pending issue, we propose a framework which applies CNNs on multiple 2D image views (or snapshots) of the point cloud. The approach consists in three core ideas. (i) We pick many suitable snapshots of the point cloud. We generate two types of images: a Red-Green-Blue (RGB) view and a depth composite view containing geometric features. (ii) We then perform a pixel-wise labeling of each pair of 2D snapshots using fully convolutional networks. Different architectures are tested to achieve a profitable fusion of our heterogeneous inputs. (iii) Finally, we perform fast back-projection of the label predictions in the 3D space using efficient buffering to label every 3D point. Experiments show that our method is suitable for various types of point clouds such as Lidar or photogrammetric data.

1 Introduction

The progress of 3D point cloud acquisition techniques and the democratization of acquisition devices have enabled the use of 3D models from real world in several economic fields such as building industry, urban planning or heritage conservation. Today's devices,

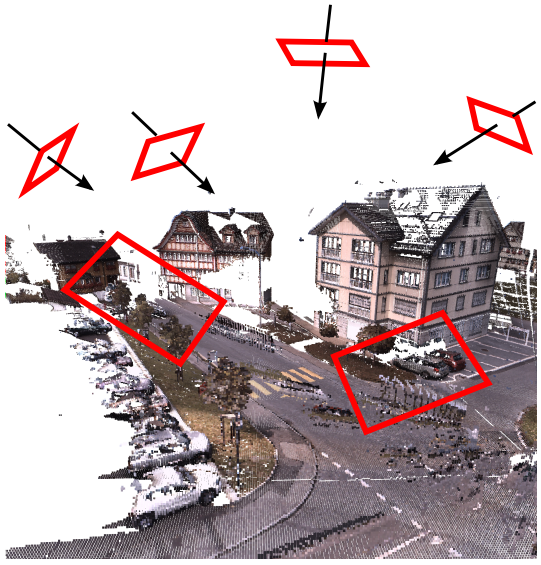


Figure 1: Generation of 2D snapshots for semantic labeling in the image space by taking random camera positions in the 3D space.

like laser scanners or photogrammetry tools, allow the production of very large and precise point clouds, up to millions of points, structured or not. Meanwhile, the last years have seen the development of algorithms and methodologies in order to reduce the human intervention for two of the most common processing tasks with point clouds: first, surface reconstruction and abstraction, and second, object recognition and scene semantic understanding. However, these tasks are still a pending research topic and in applied fields, point cloud processing remains at least partly manual.

This work address the second issue: we aim at discovering the semantics of the scene, i.e. classifying the content in the scene. In [1], the semantic discovery

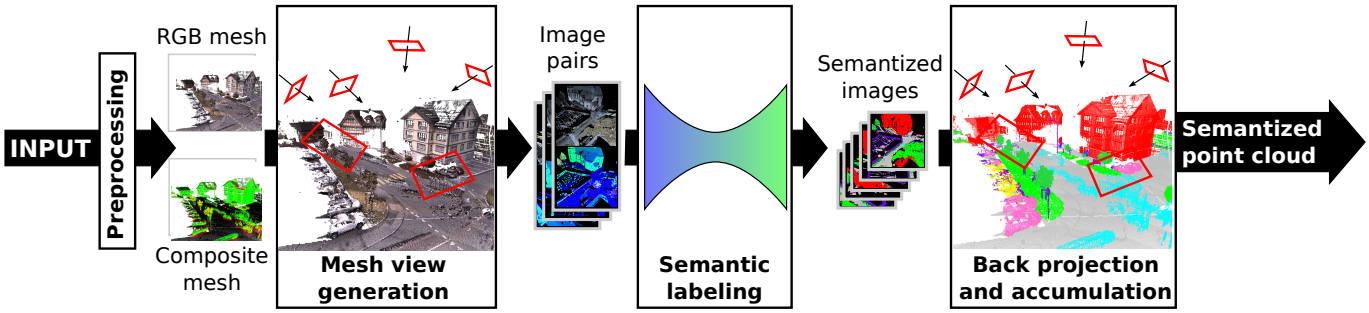


Figure 2: Work-flow of the approach.

of a scene is done using grammars on a 3D reconstructed model, so that the result is very dependent on the quality of the abstract model. Here, we adopt a different approach. Similarly to [2, 3, 4], we want to extract semantic information as soon as possible in the processing pipeline. As a matter of fact, knowing segmentation of the scene and the class of each object allows to direct the reconstruction according to each class: model or primitive fitting, regularity or symmetry constraints. More precisely, we aim at attributing a class label to each 3D point. In the image processing field, the similar task would be pixel wise labeling or semantic segmentation. Recent work on the subject focus on the design of efficient 3D descriptors by taking into account the neighborhoods of points [5, 6]. We propose a different approach based on Convolutional Neural Networks (CNNs) and particularly on segmentation networks [7, 8]. These networks reached the state of the art for image segmentation on different use cases such as KITTI [9] or aerial images [10].The originality of our approach is that our own features are simple 2D primitives: partial snapshots of the point-cloud. Then, we can do the labeling in a 2D image space (figure 1) where the segmentation networks proved to be very efficient. We show that this SnapNet framework, which extends our work presented in [11], can be applied to various scenes, outdoor and indoor, and for classes which may highly differ depending on the application. Moreover, by applying SnapNet to various types of point clouds, coming from laser sensors, photogrammetry or Red-Green-Blue-Depth (RGB-D) cameras, we show how generic and the robust the approach is.

Organization of the paper. The paper is organized as follows The section 2 presents the related work on point cloud semantic labeling. The overview of our 4-step semantic labeling method can be found in section 3. Then the four next sections detail the main steps of the algorithm: section 4 explains the preprocessing of the 3D point-cloud required to take the snapshots according to the strategy exposed in section 5, the semantic labeling and data fusion pipeline based on convolutional networks is exposed in section 6 and point-cloud labeling is detailed in section 7. Finally, in section 8, we evaluate our segmentation method.

2 Related work

Semantic segmentation of point clouds is a well known problem in computational geometry and computer vision. Starting in the 1990s, it gained in interest with the spread of acquisition devices and reconstruction techniques [12]. The objective is to identify the class membership of each 3D point. This problem is partially related to the 2D semantic segmentation, where the objective is to label each pixel of the image.

The early stages of semantic labeling for point cloud were mainly focused on aerial laser acquisition (Lidar). The objective was to discriminate building and roads from vegetation. A common approach is to discretize the point cloud on a regular grid to obtain a 2.5D elevation map authorizing the use of image processing algorithms as the image filters in [13] or maximum likelihood classification in [14]. Other low level primitives, such as planes [15], have also been used for bottom-up classification introduced in [16] or [17].

In a more general context, low level shape extraction in point clouds has also been investigated. The Hough transform, originally designed for line extrac-

tion, was successfully adapted to 3D for plane extraction in [18]. [19] proposes a generic RANSAC algorithm for geometric shape extraction in 3D point clouds. Hybrid shape extraction were investigated in [20, 4] where the surfaces which fit geometric primitives are replaced with the corresponding abstract model while voids remain as triangular mesh.

Many algorithms for extraction of higher level semantic information were published in the recent years. In urban classification [2, 21], classifying small objects like cars or street furniture [3] and discriminating between roads and natural terrain become decisive at the smallest possible scale: point level [2]. Most of the semantic labeling approaches rely on the same technique: designing the most discriminating features for the classification task. For example, in [22], the authors designed by hand a collection of expert features such as normalized height or luminance. Another approach is to a create generic descriptor space to represent the points and their neighborhood in order to learn a supervised classifier. Among these descriptors, the spin images [23], the fast point feature histograms [5] or the signature histograms [6] may be the most popular. With respect to these approaches, we use much more simple features: 2D views of the point cloud.

By using a deep learning framework, it is possible to learn not only the classifier but also the feature representation. While deep neural networks are now commonly used in image processing for classification or semantic labeling, there are only a few proposals on these tasks in 3D.

First, a family of approaches use a voxelization of the space to create 3D tensors in order to feed a 3D convolutional neural network (CNN) [24, 25, 26], mainly for object classification. However, those **voxel-based** approaches might be memory consuming. For this purpose, [27] proposed to use local, multiscale 3D tensors around the actual 3D points.

Second, **point-based** methods work directly on unordered point sets, using architectures with fully-connected and pooling layers instead of convolutional layers. Thus, PointNet [28] can output classes for the whole 3D shape or perform semantic segmentation of a 3D scene. Then, PointNet++ [29] introduced a feed-forward network which performs alternatively hierarchical grouping of points and PointNet layers optimization to try to capture local context. Indeed local configurations are highly important to be able to model complex scenes, and they are inherently captured by the multiple snapshots we take in the scene.

Third, the **multi-view** strategy consists in applying neural networks to 2D tensors which are selected views of the scene. For example, in [30], a deep framework is used to compute a metric for identifying architectural style distance between to building models. On a shape retrieval task, the multi-view approach of [31] takes several pictures of the 3D meshed object and then perform image classification using a deep network. The PANORAMA representation [32] introduces another trick: projections on bounding cylinders oriented following the 3 principal directions of the volume. Our approach has common features with these last works: we generate snapshots of the 3D scene in order to use a 2D CNN with images as input. But unlike [31] whose purpose is classification, i.e. giving a single label per 3D shape, we compute dense labeling in the images and back project the result of the semantic segmentation to the original point cloud, which results in dense 3D point labeling. Another conceptually different point is the strategy for choosing views. We do not aim to capture the whole scene (or object) in a few carefully selected views, but rather take lots of partial views and rely on the final vote to put together local predictions. This avoids introducing too much dataset-related bias into the algorithm.

3 Method overview

The core idea of our approach consists in transferring to 3D the very impressive results of 2D deep segmentation networks. It is based on the generation of 2D views of the 3D scene, as is someone was taking snapshots of the scene to sample it. The labeling pipeline is presented on figure 2. It is composed of four main processing steps: point-cloud preparation, snapshot generation, image semantic labeling and back projection of the segmentation to the original 3D space.

1. The **preprocessing** step aims at decimating the point cloud, computing point features (like normals or local noise) and generating a mesh.

2. **Snapshot generation:** from the mesh and the point attributes, we generate two types of views, Red-Green-Blue (RGB) and depth composite, by picking various camera positions (cf. Sec. 5).

3. **Semantic labeling** gives a label to each pair of corresponding pixels from the two input images. We use deep segmentation networks based on SegNet [8] and fusion with residual correction [10].

4. Finally, we **project back to 3D** the semantized images. For each point of the mesh, we select its label by looking at the images where it is visible (cf. Sec. 7).

Point cloud properties In this work we assume our point clouds have a metric scale such that voxelization outputs have the same point density. We also consider as known the vertical direction to compute the normal deviation to this vector. As presented in section 8, it is also possible to use the pipeline without RGB information but performances are downgraded.

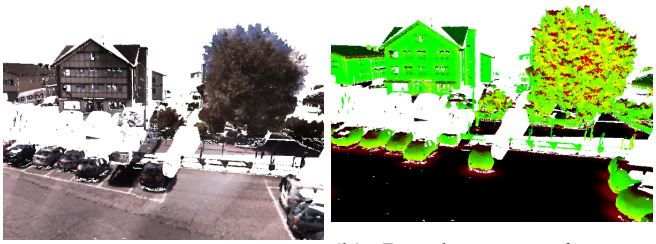
Figure 3: Point cloud (left) and mesh (right) seen from the same point of view: dense representations help understanding the scene.

4 Point cloud preprocessing

The main issue for image generation when dealing with point clouds is the sparsity. Indeed, let's assume the point density is uniform in the point cloud. When taking a snapshot, the objects on the foreground are poorly sampled and allow to see the background through them. This leads to images which are difficult to understand, even for a trained human expert. To overcome this issue, we generate a basic mesh of the scene which yields in better-looking images, as shown in Fig. 3. We now detail the algorithmic steps.

Point cloud decimation Point clouds, especially those captured with ground lasers, have varying point densities depending on the distance to the sensor. So, we first decimate the point cloud and get a lighter cloud so that subsequent processing can be applied in tractable times. To do that, we voxelize the scene, and keep the closest point to each voxel center (along with its class label at training time). In this paper, we chose a voxel size of 0.1m. It proved to produce relatively small point clouds while preserving most of the original features and shapes. Stronger decimations may lead to discarding small objects. In our experiments with semantic 3D, we reduce point cloud sizes from $20M/429M$ points to $0.4M/2.3M$ points.

Mesh generation The only a priori knowledge we have about our point-clouds is that they have homogeneous density due to decimation. For practical purposes,we chose the mesh generation algorithm from [33] among many standard methods.Although it does not give any guarantee about the topology of the generated mesh, it is not a concern for our snapshot application. It requires as input a point-cloud with normals, which we estimate by using the available code from [34]. We now denote the mesh by $\mathcal{M} = (V, F)$ with V the set of vertices and F the faces.



(a) RGB texture. (b) Depth composite texture.

Figure 4: Meshes for taking synthetic snapshots of the 3D scene.

Composite colors We aim at using both color and volume information for semantic labeling. To achieve that, we create two textures for the mesh (cf. Fig. 4). The more straightforward is the RGB texture, which takes the original point colors (cf. Fig. 4a). Then, we extract two generic features of point clouds: normal deviation to vertical and a noise estimation at a given scale. The normal deviation to the vertical at point p is:

$$\text{normdev}_p = \arccos(|\mathbf{n}_p.\mathbf{v}|)$$

where $\mathbf{n}_p$ is the normal vector and $\mathbf{v}$ is the vertical vector. The noise at a given point p is an estimation of the spread of the points in its neighborhood.

$$\text{noise}_p = \frac{\lambda_2}{\lambda_0}$$

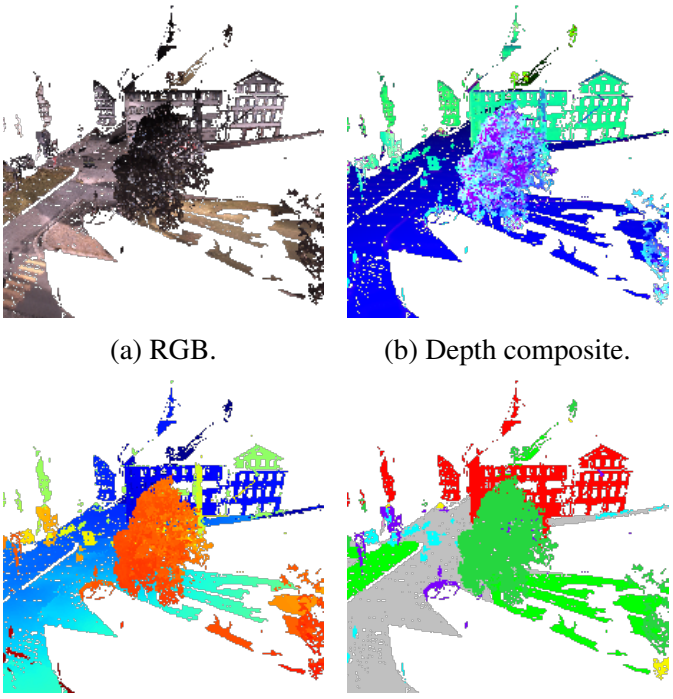
where λ_0 (resp. λ_2) is the highest singular value (resp. the lowest) obtained doing a principal component analysis estimation by singular value decomposition. Our *depth composite* texture encodes the normal deviation on the green channel and the local noise on the red one. The blue channel remains empty at this point, but later will be filled with depth (i.e. distance to the camera).

5 View generation

Once the meshes are constructed, we want to produce the images for semantic labeling. We used an approach similar to [31]. We load the model in a 3D mesh viewer and generate random camera positions and orientations to take various snapshots.

The camera parameters are generated according to two different strategies. First, in the *random* strategy, the camera center coordinates are randomly picked in the bounding box of the scene, with an altitude between 10 and 30 meters. The view direction is picked in a $45°$ cone oriented to the ground. To ensure the production of meaningful pictures, i.e. that the camera looks at the scene, we impose 20% of the pixels should correspond to actual points. Second, in the *multiscale* strategy, we pick a point of the scene, pick a line which goes through the point, and generate three camera positions on this line, oriented towards the point: thus ensuring each camera looks at the scene at various, increasing scale (allowing more and more details to be seen).

For each camera position, we generate three 224×224-pixel images, as shown on figure 5. The first one is a snapshot of the RGB mesh (figure 5a) and reflects the real texture of the model. The second one is the depth composite image (figure 5b), made of surface normal orientation and noise completed with the depth to the camera. In order to do the back projection efficiently, we also generate an image where the color of each face of F is unique so that we know which face is visible (figure 5c). Finally for training or validation purposes, when ground truth is available, we create the corresponding label image (Fig. 5d).



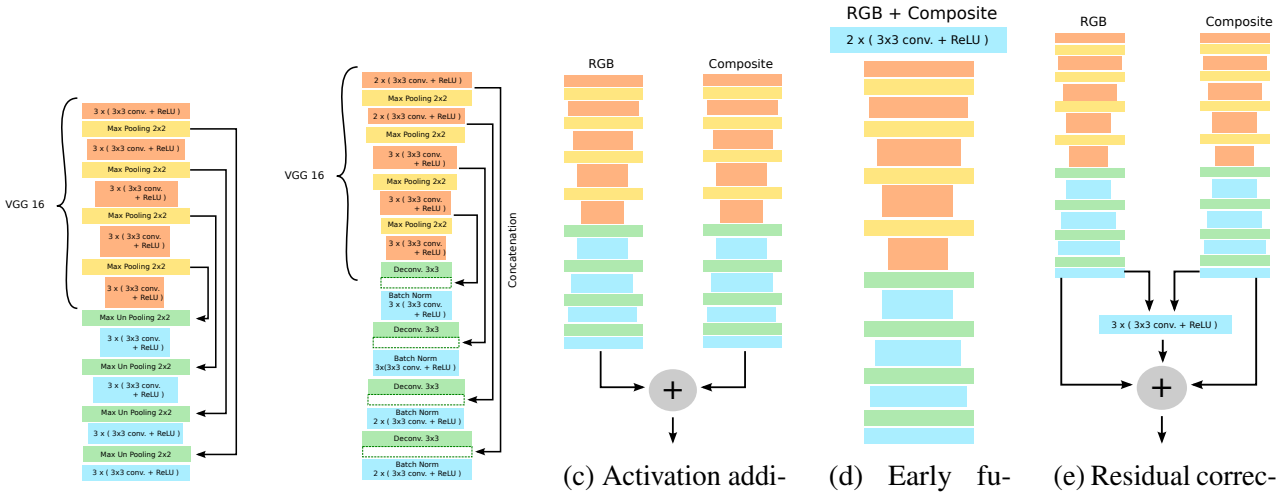
(a) RGB. (b) Depth composite.

(c) Unique face color. (d) Labels.

Figure 5: The various products of the preprocessing and view generation step.

6 Semantic Labeling

CNNs are feed-forward neural networks which make the explicit assumption that inputs are spatially organized. They are comprised of learnable convolution kernels stacked with non linear activations, *e.g.* ReLU ($max(0, x)$). Those filters perform feature extraction in order to build an internal abstract representation of the input, optimized for later classification.



(a) Segnet. (b) U-net. (c) Activation addition. (d) Early fusion. (e) Residual correction.

Figure 6: Various segmentation networks used in this paper: single-flow networks (a,b) vs. fusion networks (c,d,e)

Several deep convolutional neural networks architectures exist for semantic labeling, usually derived from the Fully Convolutional Networks [7]. Those models usually take RGB images in input and infer structured dense predictions by assigning a semantic class to every pixel of the image. In this paper, we use custom implementations of two network variants with a symmetrical encoder-decoder structure: SegNet and U-Net.

- SegNet [8] is illustrated in figure 6a. The encoder part of SegNet is based on VGG-16 [35], a deep CNN with 16 layers designed for image classification. Only the convolutional part is kept, while the fully connected layers are dropped. The decoder performs upsampling using the *unpooling* operation. During unpooling, the feature maps in the decoder are upsampled by placing the values into the positions given by the indices of the maximum during the symmetrical pooling in the encoder.
- U-Net [36] is shown in figure 6b. Also based on VGG-16 for the encoder part, it uses a different trick for upsampling. It concatenates the feature maps of the decoder convolutional layers upsampled by duplication with the symmetrical feature maps in the encoder. Later convolutions blend both types of information.

As we extract both RGB and depth composite information from the dataset, we want to fuse the data sources to improve the accuracy of the model, compared to only one source. We use several fusion strategies in order to exploit the complementarity of the depth and RGB information. Therefore, two parallel 3-channels segmentation networks are trained, one on the RGB data, the other on the composite data. The experimented strategies are the following :

- Activation addition fusion, *i.e.* averaging of the two models (figure 6c). The predictions of the two SegNet are simply averaged pixel-wise.
- Prediction fusion using residual correction [10] (figure 6e). A very short (3 layers) residual network [37] is added at the end of the two SegNet. It takes in input the before last feature maps and learns a corrective term to apply to the averaged prediction.

Moreover, we also experiment early data fusion using a pre-processing CNN that projects the two data sources into a 3-channel common representation (figure 6d). We then use this projection as input of the traditional SegNet.

Compared to model averaging, using a neural network to learn how to fuse the two predictions should achieve better results, as it will be able to learn when

to trust the individual sources based on the context and the classes predicted. As an example, figure 7 presents a case of interest for fusion. The RGB prediction is wrong on the road. The network is fooled by the texture similar to the building one. On the other hand, the depth composite predicts the good label on the road but fails on the natural terrain where the steep slope has the geometric attributes of a building roof.

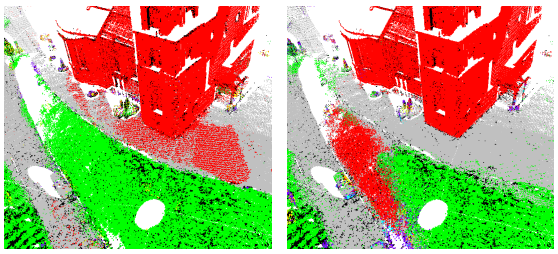


Figure 7: Mono input estimates: RGB (left) and composite (right).

7 3D back projection

This section presents how we project the pixel wise class scores obtained in section 6 on the original point cloud.

Projection to mesh. First we estimate the labels at each vertex of the mesh used to generated the images. Thanks to the unique-color-per-face images created at snapshot generation, we are able to quickly determine which faces are seen in each image pair and consequently the visible vertices of V. The score vector of the pixel is then added to the scores of each vertex of the face. This operation is iterated over all the images. Finally the vertex label is the class with the highest score.

Projection to the original point cloud. The second step is to project the labeled vertices to the original point cloud $\mathcal{P}$. We adopt a simple strategy. The label of a given point $p \in \mathcal{P}$ is the label of its nearest neighbor with label in V. For efficient computation, we build a k-d tree with V, and search for nearest neighbors. This allows not to load the whole $\mathcal{P}$, and avoid extensive memory allocation (particularly when dealing with hundreds of millions of points).

8 Experiments

In this section, we present the results of our experiments on semantic labeling of 3D point sets. In order to analyze and assess the genericity of our SnapNet approach, we used point clouds of various origins: Lidar sensors in Sec. 8.1, multiple 2D views and photogrammetry in Sec. 8.2, and low-cost RGB-D cameras in Sec. 8.3. We mainly experiment on the Semantic 3D dataset [40] (`http://semantic3d.net`).

8.1 Laser point clouds

The Semantic 3D dataset is composed of 30 laser acquisitions (15 for training and 15 for testing in the full *semantic-8* set-up) on 10 different scenes from various places and landscape types (rural, suburban, urban). A variant for computationally demanding algorithms, *reduced-8*, is also proposed: it contains the same 15 acquisitions for training but only 4 for testing. In both variants, the ground truth is available for the training set, and undisclosed for the test. There are 8 classes, namely: man-made terrain (gray), natural terrain (green), high vegetation (dark green), low vegetation (yellow), buildings (red), hardscape (purple) scanning artefacts (cyan), cars (pink).

For quantitative evaluation, we use the same metrics as the dataset benchmark. It includes the overall accuracy (OA): $OA = \frac{T}{|\mathcal{P}|}$ where $|\mathcal{P}|$ is the size of the point cloud, and T is the number of true positive i.e the number of points that received the good label. We also use the intersection over union (IoU) per class: $IoU_c = \frac{T_c}{|\mathcal{P}_c \cup P_c|}$ where T_c is the number of points of class c correctly estimated, $\mathcal{P}_c$ is the set of points with true label c and P_c is the set of points with estimated class c. Finally the global average IoU (AIoU) is defined as: $IoU = \frac{1}{|C|} \sum_{c \in C} IoU_c$

8.1.1 Architecture and parameter choice

Dataset and training. In these experiments, we defined our own custom validation set by splitting the training set: 9 acquisitions for training and 6 for validation. For each training acquisition, we generated 400 image pairs, so that we optimize the deep networks with 3600 samples. We used a stochastic gradient descent with momentum (momentum is set to 0.9). The learning rate varies according to a step

Method	RGB	Depth	AIoU	OA	IoU1	IoU2	IoU3	IoU4	IoU5	IoU6	IoU7	IoU8
SegNet RGB	✓		0.28	0.749	0.853	0.097	0.483	**0.075**	0.69	0.042	0.0	0.0
SegNet Depth Comp.		✓	0.326	0.763	0.902	0.342	0.597	0.013	0.503	0.178	0.066	0.003
SegNet add.	✓	✓	0.312	0.762	0.895	0.237	0.573	0.029	0.522	0.172	0.067	0.003
SegNet before	✓	✓	0.336	0.763	0.898	0.569	0.452	0.021	0.510	**0.179**	0.051	**0.009**
SegNet Res.	✓	✓	**0.427**	**0.805**	**0.948**	**0.739**	**0.763**	0.024	**0.710**	0.133	**0.097**	0.0

(a) Comparison of deep segmentation networks (single-input or various fusion schemes) on Semantic 3D, custom validation set.

Method	AIoU	OA	IoU1	IoU2	IoU3	IoU4	IoU5	IoU6	IoU7	IoU8
Graphical models [38]	0.391	0.745	0.804	0.661	0.423	0.412	0.647	0.124	0.000	0.058
Random forest [2]	0.494	0.850	**0.911**	0.695	0.328	0.216	0.876	0.259	0.113	0.553
Harris Net	0.623	0.881	0.818	0.737	0.742	**0.625**	**0.927**	0.283	0.178	0.671
SnapNet (SegNet / random) (ours)	0.516	0.884	0.894	**0.811**	0.590	0.441	0.853	0.303	0.190	0.050
SnapNet (U-Net / multiscale) (ours)	**0.674**	**0.910**	0.896	0.795	**0.748**	0.561	0.909	**0.365**	**0.343**	**0.772**

(b) Semantic 3D results on full test set (*semantic-8*).

Method	AIoU	OA	IoU1	IoU2	IoU3	IoU4	IoU5	IoU6	IoU7	IoU8
Graphical models [38]	0.384	0.740	0.726	0.730	0.485	0.224	0.707	0.050	0.000	0.150
Random forest [2]	0.542	0.862	0.898	0.745	0.537	0.268	0.888	**0.189**	0.364	0.447
DeePr3SS [39]	0.585	**0.889**	**0.856**	**0.832**	0.742	**0.324**	0.897	0.185	0.251	0.592
DeepNet [27]	0.437	0.772	0.838	0.385	0.548	0.085	0.841	0.151	0.223	0.423
SnapNet (U-Net / multiscale) (ours)	**0.591**	0.886	0.820	0.773	**0.797**	0.229	**0.911**	0.184	**0.373**	**0.644**

(c) Semantic 3D results on reduced test set (*reduced-8*).

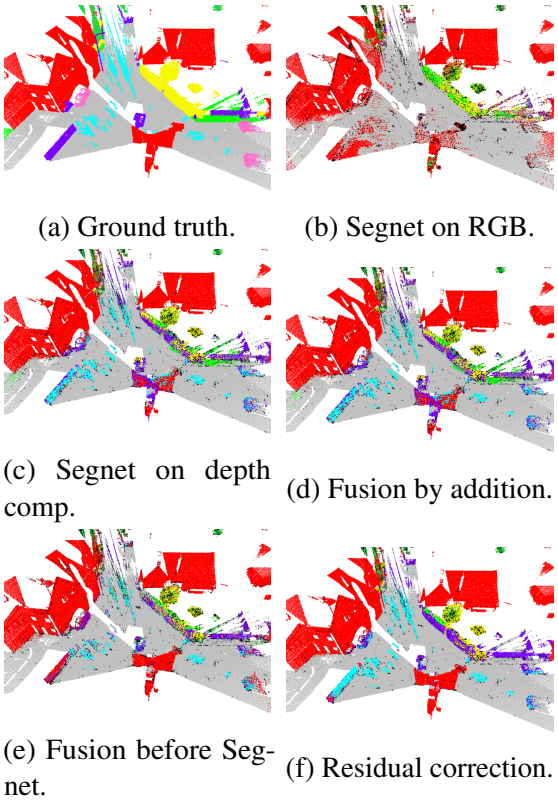
IoU: intersection over union (per class), AIoU: average intersection over union, OA: overall accuracy. Classes 1: man-made terrain, 2: natural terrain, 3: high vegetation, 4: low vegetation, 5: buildings, 6: hardscape, 7: scanning artefacts, 8: cars.

Table 1: Quantitative results on Semantic 3D.

down policy starting at 0.01. It is multiplied by 0.2 every 30 epoch. The encoder part of SegNet is initialized with the VGG16 weights [35].

At testing, we generate 500 views at 3 scales. For a point-cloud of 30M points, the computation times are the following (with: CPU Xeon 3.5GHz, GPU TitanX Maxwell): pre-processing 25 min. (7 min. with normal estimation by regression); view-generation 7 min.; inference 1 min.; back-projection 8 min.; which sum up to 41 or 23 min. for the whole point-cloud semantization. Most sensitive parameters are the number of voxels (for point-cloud decimation) and the number of snapshots.

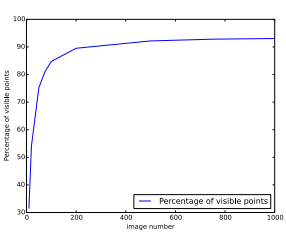
Fusion strategy choice. As explained in section 6, the different natures of the input images impose to define a fusion strategy. We quantitatively evaluate the different fusion options presented on figure 6. As a baseline, in the first result block, we trained two mono-input SegNets, taking as input the RGB or depth composite images.



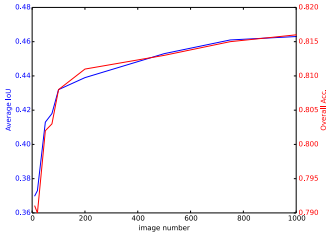
(a) Ground truth. (b) Segnet on RGB. (c) Segnet on depth comp. (d) Fusion by addition. (e) Fusion before Segnet. (f) Residual correction.

Figure 8: Same prediction view for the different fusion strategies.

View influence. The number of generated snapshots is a crucial parameter of the algorithm. Intuitively,



(a) Percentage of visible points function of snap number (Semantic3D Bildstein station 1).



(b) Overall accuracy and average IoU function of the snap number (Semantic3D custom set)

Figure 9: Influence of the number of snapshots.

the more generated views there are, the more accurate the prediction is. As shown on figures 9, it is verified. Figure 9a shows the percentage of points seen depending on the number of generated snapshots for a given scene, which reaches a plateau. Figure 9b presents the increase of accuracy and average IoU on our custom validation set with respect to the number of snapshots used for test. Base on our experiments, we chose to generate 500 views which correspond to an operating point in the curve saturation area.

8.1.2 Results and analysis

The mono-input and fusion results are presented on table 1a.

The composite network performs globally better except on buildings for which there is a great difference of texture compared to the rest of the scene. Moreover, depth composite images, that only contain geometric information, are not sensible to the texture of objects, so almost every vertical plane will be labeled as a building. This experiment shows that the two inputs are complementary and that the RGB network is not able to extrapolate the composite information only from the image texture.

The second block of table 1a is dedicated to fusion strategies. Due to the high distribution difference in the prediction maps, composite-only totally overcomes the RGB prediction, i.e. the depth composite is most of the time confident while the RGB is more hesitating. As a result, the addition of prediction scores does not improve the results compared to depth composite only. A visual glimpse of the phenomenon is presented on figures 8c and 8d, the two images are almost similar.

Operating the fusion before labeling via SegNet should overcome this issue by melting the two signals at an early stage. As expected the results are visually improved (figure 8), particularly on natural terrain class, where the association of the texture and geometric features is discriminatory. However, the fusion step before SegNet is not optimal. VGG-16 takes a 3-channel image as input, and the two convolutions added before SegNet operates a dimension reduction that may cause information loss. Moreover, the different nature of the input makes it uncertain that information from both are compatible for fusion this early in the process. Finally the best results are obtained by the residual correction network. The compromise between the fusion after Segnet (addition) and a more refined fusion using convolution (previous case) is successful. The residual correction compensates the difference of the two outputs, resulting in an increase of the performances on almost all classes.

For comparison with existing methods we confront our approach to public results on either the full *semantic-8* or the *reduced-8* datasets. We present the results for *semantic-8* in table 1b. The three other methods are the publicly available results. [38] is method for aerial images based segmentation on images descriptors and an energy minimization on a conditional random field. In [2], the authors use a random forest classifier trained on multi-scale 3D features taking into account both surface and context properties. Harris Net is briefly described as a Deep 3D Convolutional Network on the result board. Elaborating upon its name, we assume it might be a method based on 3D Harris point extraction followed by a classification using a deep framework. We present the results of two methods, SegNet with a purely random set of images, and a U-Net with zoom on snapshot strategy. To our knowledge the two networks performs equally and the main difference reside in the snapshot strategy. At redaction time, our U-Net took the first place in the leaderboard for global scores, average IoU and overall accuracy. Looking at the per class IoU, we take the lead on six out of eight categories. Among them, the performances on natural terrain, scanning artifacts and cars are drastically increased. On man-

made terrain and buildings, we place second with a comparable score as [2] and Harris Net. The use of the zoom strategy greatly improves the score on cars and scanning artifacts. The reason is that compared to the random strategy, the training dataset (and the test dataset) contains more images with small details, which makes them possible to segment. The only relative failure of the deep segmentation networks are the scanning artifacts and the hardscape classes. Even though we place first on these categories, the IoU score is low: we discuss this in section 8.4.

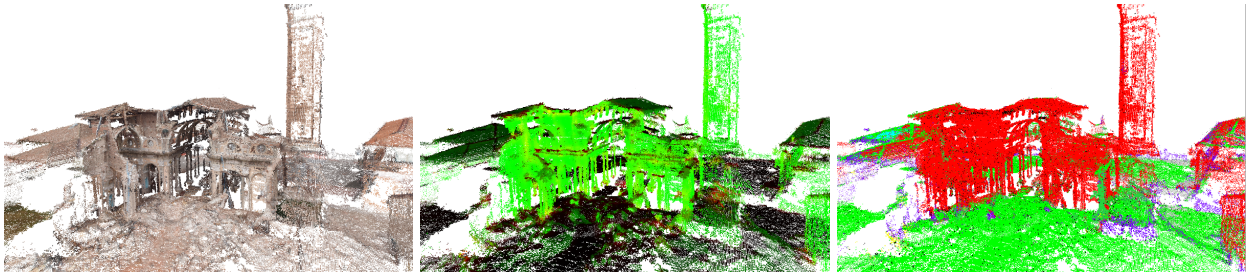
We also present the results for *reduced-8* in table 1c. While [38] and [2] are still present, different deep learning approaches are proposed for comparison. DeePr3SS [39] is also a multi-view approach, very similar to ours but later, with multi-stream fully convolutional networks for 2D classification of virtual views rendered by Gaussian point splattering. DeepNet [27] is a voxel-based approach with 3D convolutional networks applied to multiscale neighbourhoods of each scan point.SnapNet obtains the best average IoU (59.1%), and is close second in Overall Accuracy (88.6%). It gets 4 out of 8 best per-class accuracies (including buildings, cars and trees). DeePr3SS is the main competitor with the best OA and 3 out of 8 best per-class accuracies, which shows the primacy of multi-view approaches on this dataset. The *reduced-8* data raise interesting problems. First the 4 test point clouds have been decimated and so are sparser than the previous experiment. Since we used he same parameters for meshing as on the denser, original data, some rendering artifacts may occur. The good performances show that parameter choice is somewhat robust to various 3D resolutions. Second, the 4 selected scenes are the ones with the more variety between them and also with respect to the training set: different types of buildings, quite rare natural terrain (e.g. cliffs). Again, this may explain slightly inferior results than on *semantic-8*, but performances are still good enough to prove the adaptation capacity of the classifiers.

8.2 Photogrammetric point clouds

In order to evaluate the capacity of our method to be transferable, we also experiment on photogrammetric data. The figures 10a and 11a present a reconstruction of Mirabello's church destroyed after an earthquake in 2012 in Italy. Original images were taken from a drone flying over the town center and then used to extract key-points which were mapped to 3D. We use these data for two experiments: direct transfer of the network trained on *semantic-8* to photogrammetry and fine-tuning for labeling using new classes. We preprocess the data in a similar way as for laser data.

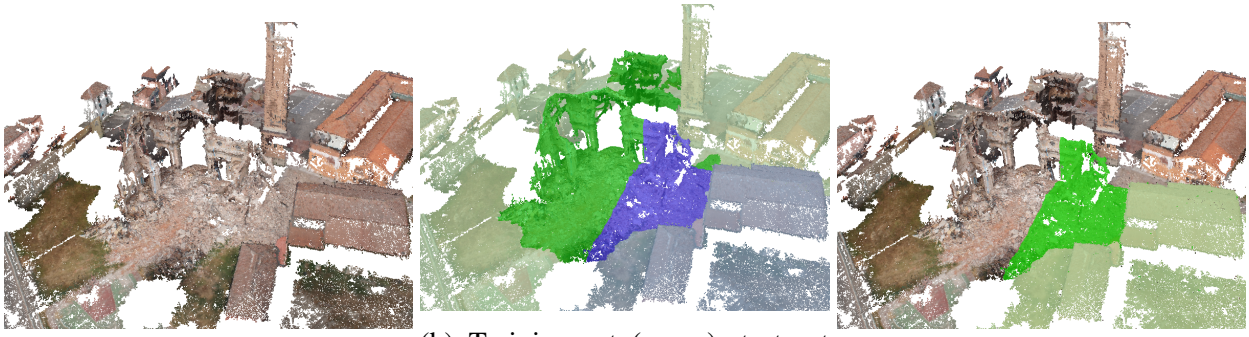
First, the network used for semantic labeling is the one trained on the full *semantic-8* training set. Figure 10 presents the visual results. Most of the visual error concentrates on ground classes and high vegetation. A lot of ground is covered by rubble coming from the destroyed building. Due to the chaotic structure of the debris, it is recognized as natural terrain. Part of the rooftops are also wrongly labeled the same way. We interpret this as a consequence of the fact that our training set contains only ground laser acquisitions so only sloping roofs are present at training time. When confronted to roofs with small inclination, the network is misled to a ground class. Finally, high vegetation labels appear on destroyed parts which are still standing. This is mainly due to the high noise estimation (red channel of the depth composite image) which is incompatible with the building class.

Second, we annotated the point cloud using two classes: rubble or non rubble ; and split it as training and testing set. The figure 11b shows the two sets (training in green) and test in blue. Rubble is represented in bright color, while non-rubble is lighter. We finetune the RGB and composite networks trained on *semantic-8* where we replaced the last classification layer (8 classes) by a new one (2 classes). As the input sizes of the original fusion module do not corresponds anymore, we train from scratch a new fusion module. The results are presented on figure 11c. The bright green points are the detected rubble and match the destroyed church walls and the surrounding wreckage. No false positives appear far from the damaged area. Interestingly, this class is quite complex, with vertical but also chaotic elements, and of the same color as most intact buildings. These results show that transfer learning was possible even in this difficult case.



(a) RGB colors. (b) Depth composite texture. (c) Predictions.

Figure 10: Semantic labeling of photogrammetric data.



(a) RGB colors (b) Training set (green), test set (blue). (c) Rubble prediction.

Figure 11: Finetuning for destroyed building detection.

8.3 RGB-D camera point clouds

Finally, we tested our approach on point clouds captured by low-cost RGB-D cameras. We use the SUN RGB-D dataset [41] made of previous smaller RGB-D datasets [42, 43, 44]. It contains more than 10k images from 4 various RGB-D sensors, completed with several types of 2D and 3D annotations over the whole dataset. We focus on the semantic segmentation task of the SUN RGB-D dataset, using the 13 classes defined in [45].

Architecture and parameter choice

Pre-processing. In the previous cases, we made the implicit assumption that point clouds where unstructured. Actually, with RGB-D data, the original point cloud is noisy and incomplete. If we apply the same decimation and meshing process, the resulting point cloud would be so downgraded that classification would be impossible. Preferably, we chose to exploit the data structure inherent to Kinect-like acquisitions. First, we apply an enhancement procedure in the image space: inpainting, using [46], to fill the missing parts of the acquisition, and denoising from [47]. The resulting point cloud is smoother and more consistent.

The meshing process is presented on figure 12. A dense mesh is computed in the image space where each vertex corresponds to a pixel. At this stage the objects at different depths are not separated, so we then remove the elongated faces. For this purpose, we discard triangles with an aspect ratio (AR) greater than 10:

$$AR = \frac{a\ b\ c}{8\,(s-a)\,(s-b)\,(s-c)} \quad \text{where} \quad s = \frac{a+b+c}{2} \tag{1}$$

View generation. The small field of view of the RGB-D sensors produces very partial and oriented scenes. But there are lots of them due to the easy acquisition process. So for the sake of computational feasibility, we restrain the number of snapshots per scene (or RGB-D pair). In the meantime, we use the a priori about the sensor orientation to select better virtual views than randomly. We choose a deterministic strategy, illustrated on figure 13 that insures views are different enough while minimizing irrelevant views.

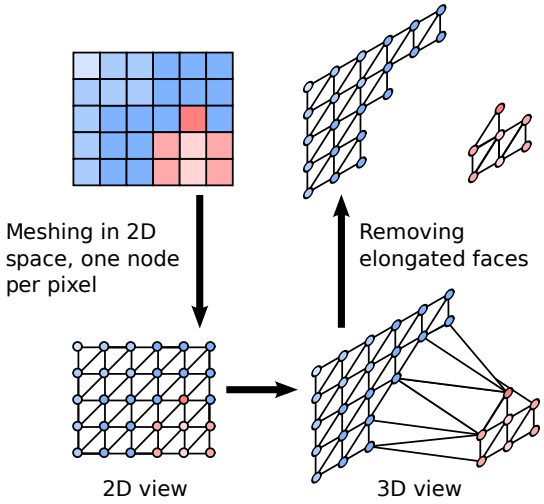


Figure 12: Meshing process for structured RGB-D data.

We set a virtual rotation point at 6 meters from the origin (camera center of the original sensor). Then we generate views at 7 and 8 meters from the virtual point. We place the camera in front, 20^{deg} on the left and on the right and reproduce the process for three elevation angle values (0^{deg}, 15^{deg} and 30^{deg}), for a total of 18 views per scene.

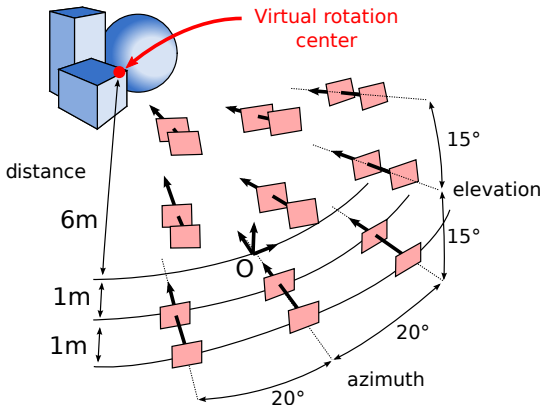


Figure 13: View generation strategy for RGB-D data.

Training. In order to face the class imbalance problem of SUN RGBD dataset we had to weight each class participation over the loss function as in [48].

Post-processing. Due to discarded mesh triangles and re-projection artifact we had to fill prediction map holes by nearest neighbor propagation.

8.3.1 Results and analysis

Table 2 shows per class accuracy rates for the 13-class segmentation task on SUNRGBD along state-of-the-art results from [45] proposing the DHA encoding for depth map (Depth-Height-Angle). Our method leads to a 14.6 points improvement on the mean accuracy. This demonstrates the capacity of our method to adapt itself to various kind of 3D data. In particular, our method ranks first for 11 furnitures or objects such as chairs, tables or beds. For the "wall" and "floor" classes however, [45]'s method perform better, though it might be a consequence of over-fitting. Indeed, these classes are more present in the dataset than other and can mislead the training. Fig. 14 shows that 3D analysis is relevant for our own understanding. Even with incomplete data, a 3D rendering allows to distinguish better planes and separated objects than the 2D image (even completed by a depth map). Thus, intricated objects are better distinguished in the 3D point cloud, which yields in better 2D segmentation maps.

Figure 14: 3D rendering of Sun RGB-D test scenes. From left to right, original RGB image, point clouds with RGB colors, ground truth and predicted labels respectively, and segmented image. (White is not labeled, green is floor, beige is wall, orange is chairs, brown for sofas, yellow for bed, blue for table, light blue for furniture.)

8.4 Limitations and perspectives

Even though the proposed approach obtains the best performances on the semantic-8 leader board there are still issues to overcome. First, a non-exhaustive training set influences the results: for example missing architectural elements or samples may explain the relatively low scores on hardscape and scanning

methods	Bed	Books	Ceiling	Chair	Floor	Furniture	Objects	Picture	Sofa	Table	TV	Wall	Window	mean
SnapNet (our)	**86.7%**	**48.3%**	**77.3%**	**74.7%**	81.4%	**64.0%**	**42.8%**	**72.3%**	**62.7%**	**83.7%**	**49.8%**	79.2%	**52.9%**	**67.4%**
SceneNet-DHA	33.2%	2.5%	40.6%	54.0%	71.1%	26.2%	22.1%	9.5%	15.0%	29.2%	0.0%	*89.2%*	0.0%	30.2%
SceneNet-DO-DHA	46.1%	5.2%	43.6%	54.8%	63.1%	37.4%	23.2%	10.7%	12.2%	29.8%	0.0%	83.6%	1.0%	31.6%
SUNRGBD-DHA	70.4%	11.2%	64.7%	69.2%	**94.0%**	48.4%	35.3%	13.7%	48.2%	63.0%	3.5%	**89.7%**	27.9%	49.2%
SUNRGBD-DO-DHA	73.6%	16.6%	*71.6%*	70.1%	93.5%	47.9%	*38.7%*	*17.2%*	*58.5%*	61.8%	6.8%	88.7%	33.9%	52.2%
SceneNet-FT-SUNRGBD-DHA	69.0%	*20.0%*	70.3%	70.7%	93.7%	49.7%	35.5%	15.7%	57.8%	*65.9%*	*14.1%*	89.0%	33.8%	52.7%
SceneNet-FT-SUNRGBD-DO-DHA	*75.6%*	13.5%	69.2%	*73.6%*	*93.8%*	*52.0%*	37.1%	16.8%	57.2%	62.7%	9.5%	88.8%	*36.5%*	*52.8%*

Best values are emphasized in bold, second best values in italics.

Table 2: Semantic segmentation results for 13 classes SUNRGBD. Results with methods used for comparison were retrieved from [45].

artifacts. For a more generic pipeline, one should use a more diversified training set. A second field of future investigation is post-processing the results to remove the outliers by regularization. For example, we could enforce the volumetric consistency of labels in a neighborhood or impose constraints on points belonging to a common extracted shape. Concerning the segmentation task over SUN RGBD dataset, our approach presents very good quantitative results in spite of a very different spatial structure of the data. This shows its genericity and robustness to sparsity and noise. Finally, a promising line of investigation is to perform data augmentation by using data from other sources. For example, synthetically generated images could be added in the training set, or scenes could be augmented with 3D models of small objects like cars. In addition to modifying the proportion of given classes, it increases the variability of the scenes (more configurations) and consequently avoids overfitting which leads to a more generic framework.

9 Conclusion

We have presented an new and efficient framework for semantic labeling of 3D point clouds using deep segmentation neural networks. We first generate RGB and geometric composite images of the scene. These pairs are the inputs of our network architectures for semantic segmentation. Several strategies for data fusion were investigated, and among them segmentation network with residual correction proved to perform the best. Finally, image segmentation were aggregated on the 3D model to give to each point a label. We experimented on both laser scans, photogrammetric reconstructions and RGB-D data. The method was evaluated against the semantic3D reduced and full datasets and obtained the best performances in the leader board concerning the global measurements and several individual classes. We also got encouraging results of transferring networks trained on laser acquisition to photogrammetric data. Finally, for single view RGB-D data, we also establish state-of-the-art performances on the segmentation task of the SUNRGBD (13 classes) dataset. Although we obtain good performances, several fields of investigation remain such as data augmentation or images generation strategies to improve the scores on small and rare classes.

Implementation details

The manipulation of point clouds, i.e the preprocessing, the image creation and the back projection was implemented using Python and C++; with PCL and the 3D viewer of `http://www.pyqtgraph.org`. The neural networks were implemented using Tensorflow and PyTorch.

Acknowledgments

The research of A. Boulch and B. Le Saux was partly supported by the ONERA project Delta and the European Commission under INACHUS, a collaborative project part of the FP7 for research, technological development and demonstration (Grant Agreement NO 607522). The authors would like to thank all partners within INACHUS for their cooperation and valuable contribution. The authors are grateful to Aibotix Italy which acquired the Mirabello images. N. Audebert's work is funded by ONERA-TOTAL research project Naomi.

References

[1] A. Boulch, M. De La Gorce, and R. Marlet, "Piecewise-planar 3D reconstruction with edge and corner regularization," in *Computer Graphics Forum*, vol. 33, pp. 55–64, 2014.

[2] T. Hackel, J. D. Wegner, and K. Schindler, "Fast semantic segmentation of 3D point clouds with strongly varying density," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 3, pp. 177–184, 2016.

[3] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3D point clouds in urban environments," *ICCV*, Sept. 2009.

[4] F. Lafarge and C. Mallet, "Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation," *Int. journal of computer vision*, vol. 99, no. 1, pp. 69–85, 2012.

[5] R. B. Rusu, A. Holzbach, N. Blodow, and M. Beetz, "Fast geometric point labeling using conditional random fields," in *IROS*, pp. 7–12, IEEE, 2009.

[6] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *ECCV*, (Hersonissos, Crete), pp. 356–369, Springer, 2010.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *CVPR*, pp. 3431–3440, 2015.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, pp. 3354–3361, IEEE, 2012.

[10] N. Audebert, B. Le Saux, and S. Lefèvre, "Semantic Segmentation of Earth Observation Data Using Multimodal and Multi-scale Deep Networks," in *ACCV*, (Taipei, Taiwan), Nov. 2016.

[11] A. Boulch, B. Le Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Eurographics/3DOR*, (Lyon, France), April 2017.

[12] M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo System," *IEEE PAMI*, vol. 15(4), pp. 353–363, 1993.

[13] C. Hug and A. Wehr, "Detecting and identifying topographic objects in imaging laser altimeter data," *International archives Photogramm. Remote Sens.*, vol. 32, no. 3 SECT 4W2, pp. 19–26, 1997.

[14] H.-G. Maas, "The potential of height texture measures for the segmentation of airborne laserscanner data," in *21st Canadian symp. on remote sensing*, pp. 154–161, 1999.

[15] E. Bughin, A. Almansa, R. G. von Gioi, and Y. Tendero, "Fast plane detection in disparity maps," in *ICIP*, pp. 2961–2964, IEEE, 2010.

[16] N. Haala, C. Brenner, and K.-H. Anders, "3D urban GIS from laser altimeter and 2D map data," *International Archives Photogramm. Remote Sens.*, vol. 32, pp. 339–346, 1998.

[17] F. Rottensteiner and C. Briese, "A new method for building extraction in urban areas from high-resolution Lidar data," *Int. Archives Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 34, no. 3/A, pp. 295–301, 2002.

[18] G. Vosselman, B. G. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanner point clouds," *Int. Archives Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 46, no. 8, pp. 33–38, 2004.

[19] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer graphics forum*, vol. 26, pp. 214–226, Wiley Online Library, 2007.

[20] F. Lafarge, R. Keriven, M. Brédif, and V. H. Hiep, "Hybrid multi-view reconstruction by jump-diffusion," in *CVPR*, pp. 350–357, IEEE, 2010.

[21] N. Chehata, L. Guo, and C. Mallet, "Airborne Lidar feature selection for urban classification using random forests," *Int. Archives Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 38, no. Part 3, p. W8, 2009.

[22] A. P. Charaniya, R. Manduchi, and S. K. Lodha, "Supervised parametric classification of aerial Lidar data," in *CVPR/W*, pp. 30–30, IEEE, 2004.

[23] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered

3D scenes," *IEEE PAMI*, vol. 21, no. 5, pp. 433–449, 1999.

[24] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *ICRA*, pp. 3050–3057, IEEE, 2014.

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *CVPR*, (Boston, USA), pp. 1912–1920, 2015.

[26] D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *IROS*, (Hamburg, Germany), pp. 922–928, 2015.

[27] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d.net: A new large-scale point cloud classification benchmark," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, vol. IV-1/W1, 2017.

[28] H. Su, , C. Qi, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, (Honolulu, Hawai, USA), July 2017.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, (Long Beach, Cal., USA), December 2017.

[30] I. Lim, A. Gehre, and L. Kobbelt, "Identifying style of 3D shapes using deep metric learning," *Computer Graphics Forum*, vol. 35, no. 5, pp. 207–215, 2016.

[31] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *ICCV*, pp. 945–953, 2015.

[32] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval," in *Eurographics Workshop on 3D Object Retrieval*, (Lyon, France), 2017.

[33] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *ICRA*, (Kobe, Japan), May 12-17 2009.

[34] A. Boulch and R. Marlet, "Deep Learning for Robust Normal Estimation in Unstructured Point Clouds," *Computer Graphics Forum*, 2016.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, (Munich), pp. 234–241, 2015.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[38] J. A. Montoya, J. D. Wegner, L. Ladickỳ, and K. Schindler, "Mind the gap: modeling local and global context in (road) networks," in *GCPR*, pp. 212–223, Springer, 2014.

[39] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3d semantic segmentation," 2017.

[40] T. Hackel, N. Savinov, L. Ladicky, J.-D. Wegner, K. Schindler, and M. Pollefeys, "Large-scale point cloud classification benchmark," in *CVPR/ Large Scale 3D Data Workshop*, 2016.

[41] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *CVPR*, (Boston, USA), pp. 567–576, 2015.

[42] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, (Florence, Italy), 2012.

[43] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work," in *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, 2013.

[44] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *2013 IEEE International Conference on Computer Vision*, (Sidney, Australia), pp. 1625–1632, 2013.

[45] A. Handa, V. Pătrăucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Understanding real world indoor scenes with synthetic data," in *CVPR*, (Las Vegas, NV, USA), 2016.

[46] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[47] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical imaging and vision*, vol. 20, no. 1, pp. 89–97, 2004.

[48] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Asian Conference on Computer Vision*, pp. 213–228, Springer, 2016.