

# FAST SEMANTIC SEGMENTATION OF 3D POINT CLOUDS WITH STRONGLY VARYING DENSITY

Timo Hackel, Jan D. Wegner, Konrad Schindler

Photogrammetry and Remote Sensing, ETH Zürich – [firstname.lastname@geod.baug.ethz.ch](mailto:firstname.lastname@geod.baug.ethz.ch)

Commission III, WG III/2

**KEY WORDS:** Semantic Classification, Scene Understanding, Point Clouds, LIDAR, Features, Multiscale

## ABSTRACT:

We describe an effective and efficient method for point-wise semantic classification of 3D point clouds. The method can handle unstructured and inhomogeneous point clouds such as those derived from static terrestrial LiDAR or photogrammetric reconstruction; and it is computationally efficient, making it possible to process point clouds with many millions of points in a matter of minutes. The key issue, both to cope with strong variations in point density and to bring down computation time, turns out to be careful handling of neighborhood relations. By choosing appropriate definitions of a point's (multi-scale) neighborhood, we obtain a feature set that is both expressive and fast to compute. We evaluate our classification method both on benchmark data from a mobile mapping platform and on a variety of large, terrestrial laser scans with greatly varying point density. The proposed feature set outperforms the state of the art with respect to per-point classification accuracy, while at the same time being much faster to compute.

## 1. INTRODUCTION

Dense 3D acquisition technologies like laser scanning and automated photogrammetric reconstruction have reached a high level of maturity, and routinely deliver point clouds with many millions of points. Due to their lack of structure, raw point clouds are of limited use for downstream applications beyond visualization and simple distance measurements, therefore much effort has gone into developing automated approaches for point cloud interpretation. The two main tasks are on the one hand to derive semantic information about the scanned objects, and on the other hand to convert point clouds into higher-level, CAD-like geometric representations. Both tasks have turned out to be surprisingly difficult to automate.

In this work we are concerned with the first problem. Much like in semantic image segmentation (a.k.a. pixel labelling), we attempt to attach a semantic label to every single 3D point. Point-wise labels are *per se* also not a final result. But like several other authors, e.g. (Golovinskiy et al., 2009, Lafarge and Mallet, 2012), we argue that it makes sense to extract that information at an early stage, in order to support later processing steps. In particular, semantic labels (respectively, soft class probabilities) allow one to utilize class-specific a-priori models for subsequent tasks such as surface fitting, object extraction, etc.

Early work on semantic segmentation of airborne LiDAR data aimed to classify buildings, trees and the ground surface, and sometimes also to reconstruct the buildings. In many cases the point cloud is first converted to a heightfield, so that standard image processing methods can be applied, e.g. (Hug and Wehr, 1997, Haala et al., 1998). More recent work in aerial imaging has followed the more general approach to directly label 3D points, using geometric information derived from a set of neighbors in the point cloud, e.g. (Chehata et al., 2009, Yao et al., 2011). 3D point labelling is also the standard method for terrestrial mobile mapping data, for which it is not obvious how to generate a consistent 2.5-dimensional height (respectively, range) map, e.g. (Golovinskiy et al., 2009, Weinmann et al., 2013). Like many other recent works we follow the standard pipeline of discriminative classification: First, extract a rich, expressive set of features

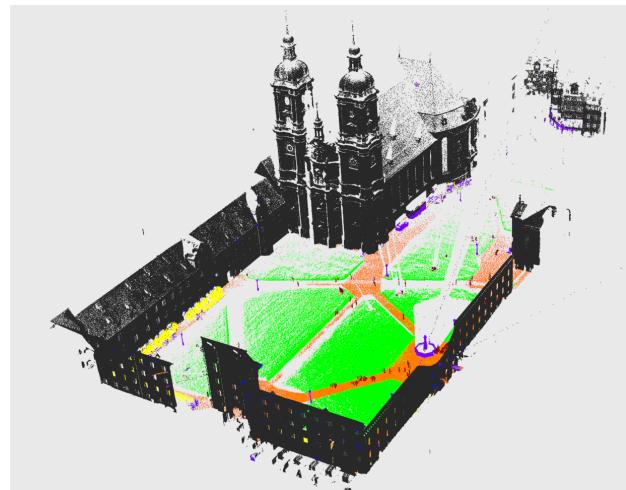


Figure 1. The task: given an unstructured, inhomogeneous point cloud, assign point-wise semantic labels.

that capture the geometric properties of a point's neighborhood. Then, train an off-the-shelf classifier to predict class-conditional probabilities from those features.

Compared to existing work, we aim to handle all kinds of scan data, including point clouds from static terrestrial laser scanning (TLS) or close-range photogrammetry. Due to the measurement principle (polar recording from a limited number of instrument locations), this type of data exhibits huge variations in point density. First of all, point density naturally decreases quadratically with distance. Second, the laser intensity (respectively, for image-based reconstruction the amount of accurately localisable surface detail) also decreases with distance, so that more points do not generate a return at all. Third, the setup with few viewpoints from close range makes it all but impossible to avoid large occlusions.

A second goal of this work is computational efficiency. Real point clouds are big – a single laser scan nowadays has tens of millions of points. Therefore a practically useful method should

not involve expensive computations: if one uses 1 millisecond per point, it already takes approximately 3 hours to classify 10 million points. The contribution of the present paper is a feature set that addresses both variations in point density and computational efficiency. We propose variants and approximations of a number of existing features to capture the geometry in a point's local neighborhood. Central to the improved feature definitions is the concept of *neighborhood*. By appropriately choosing which nearby points form the basis for computing features at different scales, the proposed method (*i*) is robust against strong variations of the local point density, respectively point-to-point distance; and (*ii*) greatly speeds up the pointwise feature extraction that constitutes the computational bottleneck.

We present experimental evaluations on several different datasets from both mobile mapping and static terrestrial LiDAR, see Figure 1. For *Paris-Rue-Cassette* and *Paris-Rue-Madame* the proposed method reaches overall classification accuracies of 95–98% at a mean class recall 93–99%, to our knowledge the best results reported to date. The end-to-end computation time of our system on these datasets is < 4 minutes per 10 million points, compared to several hours reported in previous work.

## 2. RELATED WORK

Semantic segmentation of point clouds has mostly been investigated for laser scanner data captured from airplanes, mobile mapping systems, and autonomous robots. Some of the earliest work on point cloud classification dealt with airborne LiDAR data, with a focus on separating buildings and trees from the ground surface, and on reconstructing the buildings. Often the point cloud is converted to a regular raster heightfield, in order to apply well-known image processing algorithms like edge and texture filters for semantic segmentation (Hug and Wehr, 1997), usually in combination with maximum likelihood classification (Maas, 1999) or iterative bottom-up classification rules (Haala et al., 1998, Rottensteiner and Briese, 2002). Given the focus on buildings, another natural strategy is to model them with a limited number of geometric 2D (Schnabel et al., 2007, Pu and Vosselman, 2009) or 3D (Li et al., 2011, Xiao and Furukawa, 2014) primitives, which are fitted to the points – see also (Vosselman et al., 2004) for an overview of early work. A limitation of such methods is that for most realistic scenes a fixed shape library is insufficient. It has thus been proposed to fill the remaining gaps with unstructured triangle meshes, generated by triangulating the raw points (Lafarge and Mallet, 2012). Recent work in aerial imaging has followed the more general strategy also used in this paper, namely to first attach semantic meaning to individual points through supervised classification, and then continue high-level modelling from there (Charaniya et al., 2004, Chehata et al., 2009, Niemeyer et al., 2011, Yao et al., 2011). Also related to that strategy are methods for classifying tree species in forestry, e.g. (Orka et al., 2012, Dalponte et al., 2012).

A second main application area is the semantic classification of point clouds acquired with mobile mapping systems, sometimes in combination with airborne LiDAR (Kim and Medioni, 2011); to extract for example roads (Boyko and Funkhouser, 2011), buildings (Pu et al., 2011), street furniture (Golovinskiy et al., 2009) or trees (Monnier et al., 2012). Several authors solve for all relevant object classes at once, like we do here, by attaching a label to every single 3D point (Weinmann et al., 2013, Dohan et al., 2015). Related work in robotics uses point classification to separate the ground from trees and other obstacles (Lalonde et al., 2006).

At the heart of point-wise semantic labelling is the task of designing appropriate features (descriptors), which capture the geometric properties of a point's neighborhood such as roughness,

surface orientation, height over ground, etc. A large variety of geometric 3D point descriptors exist. Among the most popular ones are spin images (Johnson and Hebert, 1999), fast point feature histograms (FPFH) (Rusu et al., 2009) and signatures of histograms (SHOT) (Tombari et al., 2010). These methods were originally developed as descriptors for sparse keypoints, and while they do work well also as dense features for point labelling, they tend to be expensive to compute and thus do not scale well to large point clouds. Features designed for dense point cloud description are often based on the structure tensor of a point's neighborhood (Demantke et al., 2011), custom descriptors for vertical cylindrical shapes (Monnier et al., 2012) and the height distribution in a vertical column around the point (Weinmann et al., 2013). Some authors have tried to avoid the analysis of the structure tensor to describe shape, e.g. by characterising the shape with a randomised set of histograms (Blomley et al., 2014). In our experience this does not improve the results, but has significantly higher runtime. For speech and image understanding, end-to-end feature learning with convolutional neural networks (CNNs) has in the last few years become a de-facto standard. To our knowledge, it has only been used – with some success – for 3D voxel grids (Lai et al., 2014, Wu et al., 2015), but not yet for generic point clouds; Presumably because the convolutional strategy is not efficient in the absence of a regular neighborhood structure.

We found that features derived from the structure tensor and the vertical point distribution work well over a wide range of scales, and are particularly fast when combined with an efficient approximation of a point's multi-scale neighborhood (Pauly et al., 2003). Our proposed feature extraction can thus be seen as a fast multi-scale extension of (Weinmann et al., 2013). A common strategy to account for correlations between class labels (e.g. smoothness, or frequent neighborhood patterns) is to use the point-wise class probabilities as unary term in some sort of random field model. This has also been tried for point cloud classification. For example, (Shapovalov et al., 2010) propose a non-associative Markov random field for semantic 3D point cloud labelling, after a pre-segmentation of the point cloud into homogeneous 3D segments. (Najafi et al., 2014) follow the same idea and add higher-order cliques to represent long-range correlations in aerial and mobile mapping scans of cities. (Rusu et al., 2009) use FPFH together with a Conditional random field (CRF) to label small indoor LiDAR point clouds. (Niemeyer et al., 2011, Schmidt et al., 2014) use a CRF with Random Forest unaries to classify urban objects, respectively land-cover in shallow coastal areas. In the present work we do not model correlations between points explicitly. However, since our method outputs class-conditional probabilities for each point, it is straight-forward to combine it with a suitable random field prior. We note, however, that at least local Potts-type interactions would probably have at most a small effect, because smoothness is implicitly accounted for by the high proportion of common neighbors shared between nearby points. On the other hand, inference is expensive even in relatively simple random field models. To remain practical for real point clouds one might have to resort to local tiling schemes, or approximate inference with fast local filtering methods (He et al., 2010).

## 3. METHOD

Our goal is efficient point cloud labelling in terms of both runtime and memory, such that the algorithm is applicable to point clouds of realistic size. The main bottleneck, which significantly slows down processing, are the large number of 3D nearest-neighbor queries. For each single point, sets of nearest neighbors must be found at multiple scales, as a basis for computing geometric features. Although efficient search structures exist for nearest-neighbor search, they are still too slow to retrieve large sets of

neighbors for many millions of points. The trick is to resort to an approximate procedure. Instead of computing (almost) exact neighborhoods for each point, we down-sample the entire point cloud to generate a multi-scale pyramid with decreasing point density, and compute a separate search structure per scale level (Section 3.1). The search then returns only a representative subset of all nearest neighbors within some radius, but that subset turns out to be sufficient for feature computation. The approximation makes the construction of multi-scale neighbourhoods so fast that we no longer need to limit the number of scale levels. Empirically, feature extraction for ten million points over nine scale levels takes about 3 minutes on a single PC, see Section 4. I.e., rather than representing the neighborhood at a given scale by all points within a certain distance, we only sample a subset of those points; but in return gain so much speed that we can exhaustively cover all relevant scale levels, without having to select a single “best” scale for a point, e.g. (Weinmann et al., 2013).

### 3.1 Neighborhood approximation

Generally speaking, there are two different ways to determine a point’s neighborhood: geometrical search, for instance radius search, as well as  $k$ -nearest-neighbor search. From a purely conceptual viewpoint radius search is the correct procedure, at least for point clouds with reasonably uniform point density, because a fixed radius (respectively, volume) corresponds to a fixed geometric scale in object space. On the other hand, radius search quickly becomes impractical if the point density exhibits strong variations within the data set, as in terrestrial scans. A small radius will enclose too few neighbors in low-density regions, whereas a large radius will enclose too many points in high-density regions. We thus prefer  $k$ -nearest neighbor search, which in fact can be interpreted as an approximation to a density-adaptive search radius.

Brute force nearest neighbor search has computational complexity  $\mathcal{O}(n)$ , linear in the number of points  $n$ . A classical data structure to speed up the search, especially in low dimensions, are  $k$ D-trees (Friedman et al., 1977). By recursive, binary splitting they reduce the average complexity to  $\mathcal{O}(\log n)$ , while the memory consumption is  $\mathcal{O}(d \cdot n)$ , with  $d$  the dimension of the data. Beyond the basic  $k$ D-tree, a whole family of approximate search techniques exists that are significantly faster, including approximate nearest neighbor search with the best-bin-first heuristic, and randomized search with multiple bootstrapped  $k$ D-trees. We use the method of (Muja and Lowe, 2009) to automatically select and configure the most suitable  $k$ D-tree ensemble for our problem.

However, finding all neighbors at a large scale will require a large number  $k$ , respectively a large search radius, such that  $k$ D-trees become computationally demanding, too. To further accelerate processing it has been proposed to approximate larger neighborhoods, by first downsampling the point cloud and then picking a proportionally smaller number  $k$  of nearest neighbors as representatives (Brodi and Lague, 2012). Embedded in our multi-scale setting this leads to the following simple algorithm (Figure 2): (i) generate a scale pyramid by repeatedly downsampling the point cloud with a voxel-grid filter; and (ii) compute features at each scale from a fixed, small number  $k$  of nearest neighbors. The widely used voxel-grid filter proceeds by dividing the bounding volume into evenly spaced cubes (“voxels”) of a given size, and replacing the points inside a voxel by their centroid. The method is computationally efficient: the memory footprint is relatively small, because the data is sparse and only voxels containing at least one point must be stored; and parallelization over different voxels is straight-forward. Moreover, the filtering gradually evens out the point density by skipping points only in high-density regions, such that a constant  $k$  will at high scale levels

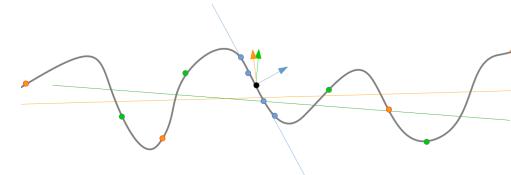


Figure 2. Sampling a point cloud at different scales (denoted by colors) reveals different properties of the underlying surface – here different surface normals (colored arrows). Multi-scale feature extraction therefore improves classification.

approximate a fixed search radius. As a side effect, the reduced variation in point density also makes radius search feasible. We will exploit this to generate height features (see Table 1), because  $k$ NN is not reliable for cylindrical neighborhoods. In our implementation we choose  $k = 10$ , which proved to be a suitable value (Weinmann et al., 2015), set the smallest voxel spacing to 2.5 cm, and use 9 rounds of standard multiplicative downsampling with a factor of 2 from one level to the next. These settings ensure that the memory footprint is at most twice as large as that of the highest resolution, and reach a maximum voxel size (point spacing) of 6.4 m. In terms of runtime, the benefit of approximating the multi-scale neighborhood is twofold. First, a small  $k$  is sufficient to capture even long-range geometric context, while the approximation of the true neighborhood with a small, but well-distributed subsample is acceptable for coarse long-range information. Second, repeated downsampling greatly reduces the total number  $n$  of points, hence the neighborhood search itself gets faster at coarse scales. We point out that, alternatively, one could also generate a similar multi-scale representation with octrees (Elseberg et al., 2013).

### 3.2 Feature extraction

In our application we face a purely geometric classification problem. We avoid using color and/or intensity values, which, depending on the recording technology, are not always available; and often also unreliable due to lighting effects. Given a point  $\mathbf{p}$  and its neighborhood  $\mathcal{P}$ , we follow (Weinmann et al., 2013) and use 3D features based on eigenvalues<sup>1</sup>  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$  and corresponding eigenvectors  $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$  of the covariance tensor  $C = \frac{1}{k} \sum_{\mathbf{i} \in \mathcal{P}} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^\top$ , where  $\mathcal{P}$  is the set of  $k$  nearest neighbors and  $\bar{\mathbf{p}} = \text{med}_{\mathbf{i} \in \mathcal{P}}(\mathbf{p}_i)$  is its medoid. Eigenvalues are normalised to sum up to 1, so as to increase robustness against changes in point density. We augment the original feature set with four additional features that help to identify crease edges and occlusion boundaries, namely the first and second order moments of the point neighborhood around the eigenvectors  $\mathbf{e}_1$  and  $\mathbf{e}_2$ . Finally, in order to better describe vertical and/or thin objects like lamp posts, traffic signs or tree trunks, we also compute three further features from the height values ( $z$ -coordinates) in upright, cylindrical neighborhoods. Table 1 gives a summary and formal definition of all features.

The local structure tensor is computed from  $k = 10$  points (current point plus 9 neighbors) over 9 scale levels ranging from voxel size  $0.025 \times 0.025 \times 0.025 \text{ m}^3$  to  $6.4 \times 6.4 \times 6.4 \text{ m}^3$ . In total this amounts to 144 feature dimensions. Note, the downsampling only affects which neighbors are found, via the  $k$ D-trees. Every point in the original data is classified, but in regions of extremely high point density multiple points may fall into the same voxel already at 2.5 cm resolution, and thus have the same neighbors.

<sup>1</sup>We note that it is computationally more efficient to analytically solve for eigenvectors and eigenvalues of a  $3 \times 3$  matrix, rather than use a numerical solver from a linear algebra library.

	Sum	$\lambda_1 + \lambda_2 + \lambda_3$
	Omnivariance	$(\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}}$
	Eigenentropy	$-\sum_{i=1}^3 \lambda_i \cdot \ln(\lambda_i)$
	Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
	Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
	Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
	Surface Variation	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
	Sphericity	$\lambda_3/\lambda_1$
	Verticality	$1 -  \langle [0 \ 0 \ 1], \mathbf{e}_3 \rangle $
moment	1 <sup>st</sup> order, 1 <sup>st</sup> axis	$\sum_{i \in \mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1 \rangle$
	1 <sup>st</sup> order, 2 <sup>nd</sup> axis	$\sum_{i \in \mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2 \rangle$
	2 <sup>nd</sup> order, 1 <sup>st</sup> axis	$\sum_{i \in \mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1 \rangle^2$
	2 <sup>nd</sup> order, 2 <sup>nd</sup> axis	$\sum_{i \in \mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2 \rangle^2$
height	Vertical range	$z_{\max} - z_{\min}$
	Height below	$z - z_{\min}$
	Height above	$z_{\max} - z$

Table 1. Our basic feature set consists of geometric features based on eigenvalues of the local structure tensor, moments around the corresponding eigenvectors, as well as features computed in vertical columns around the point.

**3.2.1 Approximate 3D shape context.** The above covariance-based features describe surface properties. However, for more complex objects they may be insufficient, especially near contour edges. We found that explicit contour features in some situations improve classification. To encode contour information, if needed, we have also developed efficient variants of *Shape Context 3D (SC3D)* (Frome et al., 2004) and *Signature of Histogram of Orientations (SHOT)* (Tombari et al., 2010). SC3D and SHOT are histogram-based descriptors similar in spirit to the original shape context for images (Belongie et al., 2002). Both divide the surroundings of a point into multiple smaller (radial) bins, see Figure 3, and therefore require large neighborhoods to be effective. The main difference is that SC3D fills a single histogram with vectors from the center to the neighbors, whereas SHOT consists of a set of histograms over the point-wise normals, taken separately per bin. We have tried both descriptors in their original versions, and found that computing them densely for a single laser scan of moderate size ( $\approx 10^6$  points) takes multiple days on a good workstation. Moreover, it turned out that, except near contour edges, the original SHOT and SC3D did not add much extra information to our feature set, and were almost completely ignored during classifier training. We therefore first learn a binary classifier that predicts from the features of Table 1 which points lie on (or near) a contour (Hackel et al., 2016). Only points with high contour likelihood are then used to fill the descriptors. This turns out to make them more informative, and at the same time speeds up the computation.

Again, our scale pyramid (Section 3.1) greatly accelerates the feature extraction. We do *not* compute all bins with the same (highest) point density. Instead, only histogram counts inside the smallest radial shell are computed at the highest resolution, whereas counts for larger shells are computed from progressively coarser scales. As long as there are more than a handful of points in each bin, the approximation roughly corresponds to rescaling the bin counts of the original histograms as a function of the radius, and still characterizes the point distributions well. We dub our fast variants approximate SHOT (A-SHOT) and approximate SC3D (A-SC3D). Even with the proposed approximation, computing either of these descriptors densely takes  $\approx 30$  minutes for  $3 \cdot 10^7$  points. As will be seen in Section 4., they only slightly improve the results, hence we recommend the additional descriptors only for applications that require maximum accuracy.

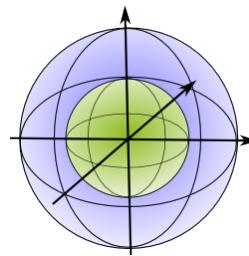


Figure 3. Histogram bins (shells are different colored spheres).

### 3.3 Classification and training

Given feature vectors  $x$ , we learn a supervised classifier that predicts conditional probabilities  $P(y|x)$  of different class labels  $y$ . We use a Random Forest classifier, because it is straight-forward to parallelize, is directly applicable to multi-class problems, by construction delivers probabilities, and has been shown to yield good results in reasonable time on large point clouds (Chehata et al., 2009, Weinmann et al., 2015). Random Forest parameters are found by minimizing the generalization error via 5-fold cross validation. We find 50 trees, Gini index as splitting criterion, and tree depths  $\approx 30$  to be optimal parameters for our application. For our particular problem, the class frequencies in a certain scan do not necessarily represent the prior distribution over the class labels, because objects closer to the scanner have quadratically more samples per unit of area than those further away. This effect is a function of the scene layout and the scanner viewpoint. It can lead to heavily biased class frequencies and thus has to be accounted for. We resolve the issue pragmatically, by down-sampling the training set to approximately uniform resolution. Besides better approximating the true class distributions, down-sampling has the side effect that labelled data from a larger area fits into memory, which improves generalization capability of the classifier.

For maximum efficiency at test time we also traverse the random forest after training and build a list of the used features. Features that are not used are subsequently not extracted from test data.<sup>2</sup> In practice all covariance features are used, only if one adds also the A-SC3D or A-SHOT descriptors some histogram bins are not selected. Note that reducing the feature dimension for testing only influences computation time. Memory usage is not affected, because at test time we only need to store point coordinates and the precomputed  $k$ -D-tree structure. Features are computed on the fly and not stored. For very large test sets it is straight-forward to additionally split the point cloud into smaller (volumetric) slices or tiles with only the minimum overlap required for feature computation (Weinmann et al., 2015).

## 4. EXPERIMENTS

For our evaluation, data from different recording technologies was classified. To compare to prior art, we use well-known databases from mobile mapping devices, namely *Paris-Rue-Cassette* and *Paris-Rue-Madame*. These point clouds were recorded in Paris using laser scanners mounted on cars (Serna et al., 2014, Vallet et al., 2015). Both datasets are processed with the same training and test sets as in (Weinmann et al., 2015), which consist of 1000 training samples per class and  $\approx 1.2 \cdot 10^7$ , respectively  $\approx 2 \cdot 10^7$  test samples. Furthermore, we test our method on multiple terrestrial laser scans from locations throughout Europe

<sup>2</sup>Note that this type of feature selection does not influence the classification result at all.

<b>Paris-rue-Madame</b>	Our Method			(Weinmann et al., 2015)		
	Recall	Precision	$F_1$	Recall	Precision	$F_1$
Façade	0.9799	0.9902	0.9851	0.9527	0.9620	0.9573
Ground	0.9692	0.9934	0.9811	0.8650	0.9782	0.9182
Cars	0.9786	0.9086	0.9423	0.6476	0.7948	0.7137
Motorcycles	0.9796	0.4792	0.6435	0.7198	0.0980	0.1725
Traffic signs	0.9939	0.3403	0.5070	0.9485	0.0491	0.0934
Pedestrians	0.9987	0.2414	0.3888	0.8780	0.0163	0.0320
Overall accuracy	0.9755			0.8882		
Mean class recall	0.9833			0.8353		
Mean $F_1$ -score	0.7413			0.4812		

<b>Paris-rue-Cassette</b>	Our Method			(Weinmann et al., 2015)		
	Recall	Precision	$F_1$	Recall	Precision	$F_1$
Façade	0.9421	0.9964	0.9685	0.8721	0.9928	0.9285
Ground	0.9822	0.9871	0.9847	0.9646	0.9924	0.9783
Cars	0.9307	0.8608	0.8943	0.6112	0.6767	0.6423
Motorcycles	0.9758	0.5199	0.6784	0.8285	0.1774	0.2923
Traffic signs	0.8963	0.1899	0.3134	0.7657	0.1495	0.2501
Pedestrians	0.9687	0.2488	0.3960	0.8225	0.0924	0.1661
Vegetation	0.8478	0.5662	0.6790	0.8602	0.2566	0.3953
Overall accuracy	0.9543			0.8960		
Mean class recall	0.9348			0.8178		
Mean $F_1$ -score	0.7020			0.5218		

Table 2. Quantitative results for iQmulus / TerraMobilita and Paris-Rue-Madame databases.

and Asia. Each scan contains  $\approx 3 \cdot 10^7$  points and has been acquired from a single viewpoint, such that the data exhibits strong variations in point density. We learn the classifier from 8 scans (sub-sampled for training, see Section 3.3) and test on the other 10 scans. On these more challenging examples we also evaluate the influence of the additional A-SHOT and A-SC3D descriptors. As performance metrics for quantitative evaluation, we use *precision*, *recall* and  $F_1$ -score, individually for each class. Moreover, we show overall accuracy and mean class recall over the whole dataset, to evaluate the overall performance.

#### 4.1 Implementation Details

All software is implemented in C++, using the *Point Cloud Library* ([pointclouds.org](http://pointclouds.org)), *FLANN* ([github.com/mariusmuja/flann](https://github.com/mariusmuja/flann)) for nearest-neighbor search and the *ETH Random Forest Template Library* ([prs.igp.ethz.ch/research/Source\\_code\\_and\\_datasets.html](https://prs.igp.ethz.ch/research/Source_code_and_datasets.html)). All experiments are run on a standard desktop PC with Intel Xeon E5-1650 CPU (hexa-core, 3.5 GHz) and 64 GB of RAM. The large amount of RAM is only needed for training, whereas classification of a typical point cloud requires  $< 16$  GB. Point-wise classification is parallelized with *OpenMP* ([openmp.org/wp](http://openmp.org/wp)) across the available CPU cores. Intuitively, porting the classification part to the GPU should be straight-forward and could further reduce runtime.

Our classification routine consists of two computational stages. First, for each scale level the  $k$ D-tree structure is pre-computed. Second, the feature computation is run for each individual point, at each scale level (Section 3.2). During training, this is done for all points, so that all features are available for learning the classifier. The Random Forest uses Gini impurity as splitting criterion, its hyper-parameters (number and depth of trees) are found with grid search and 5-fold cross-validation. Typically, 50 trees of maximum depth  $\approx 30$  are required. Having found the best parameters, the classifier is relearned on the entire training set. At test time, the feature vector for each point is needed only once, hence features are not precomputed, but rather evaluated on the fly to save memory.

<b>Paris-rue-Cassette</b>	(Weinmann et al., 2015)	this paper
features	23,000 s	191 s
training	2 s	16 s
classification	90 s	60 s
total	23'092 s	267 s

Table 3. Computation times for processing the *Paris-Rue-Cassette* database on a single PC. Our feature computation is much faster. The comparison is indicative, implementation details and hardware setup may differ, see text.

#### 4.2 Results on Mobile Mapping Data

On the MLS datasets we can compare directly to previous work. The best results we are aware of are those of (Weinmann et al., 2015). Quantitative classification results with the base features (covariance, moment and height) are shown in Table 2. Our mean class recall is 98.3% for Paris-Rue-Madame and 93.5% for Paris-Rue-Cassette, while our overall accuracy is 97.6%, respectively 95.4%. Most precision and recall values, as well as all  $F_1$ -scores, are higher than the previously reported numbers. On average our  $F_1$ -scores are more than 20% higher. We attribute the gains mainly to the fact that we need not restrict the number of scales, or select a single one, but instead supply the classifier with the full multi-scale pyramid from 0.025 to 6.4 m. The fact that practically all features are used by the Random Forest supports this assertion. Moreover, the additional moment features, which are almost for free once the eigen-decomposition of the structure tensor has been done, seem to improve the performance near occlusion boundaries. Example views of the classified point clouds are depicted in Figure 4. A feature relevance test was performed by removing each feature in turn and re-training the classifier. The feature with the strongest individual influence is  $z - z_{\min}$  with a performance drop by 5 percent points. Leaving out one of the remaining features changes the result by less than 1 percent point. Due to the small amount of training data neither A-SHOT nor A-SC3D were used for the mobile mapping databases. We also tested our system on further parts of the data, for which there is

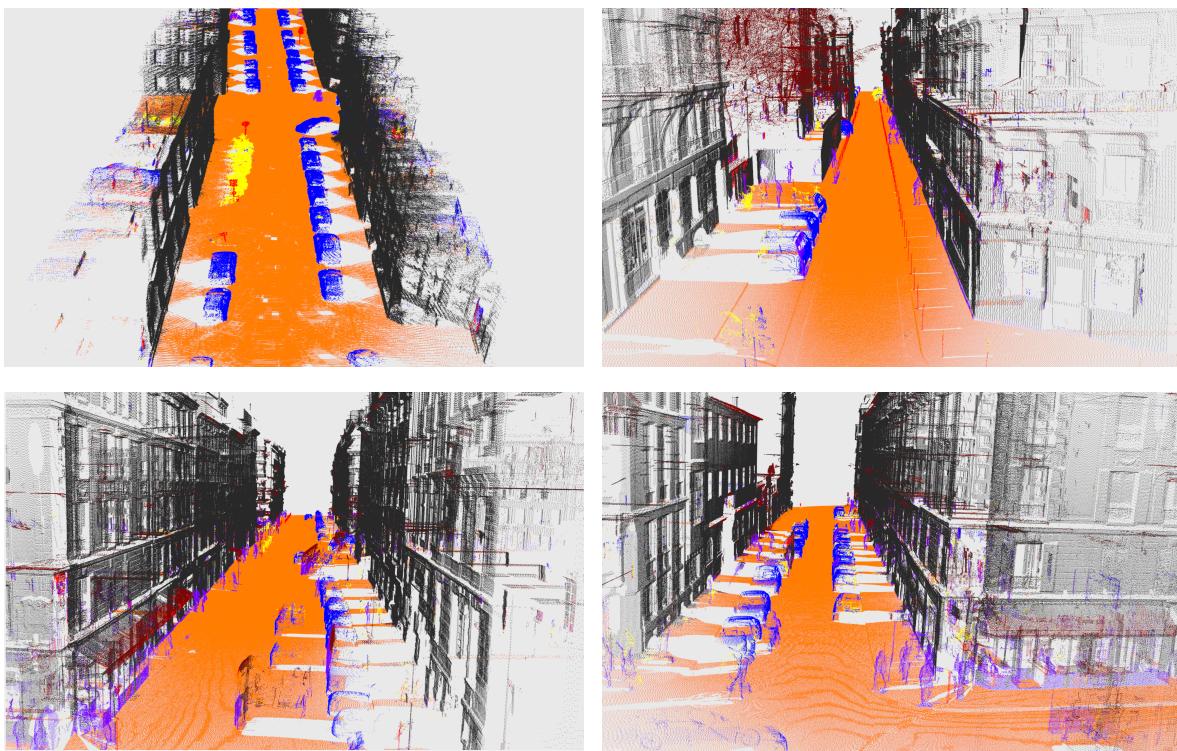


Figure 4. Results from iQmulus / TerraMobilita dataset. *Top row:* Paris-Rue-Madame (left) and Paris-Rue-Cassette (right) with classes: façade, ground, cars, motorcycles, traffic signs, pedestrians and vegetation. *Bottom row:* Further results without ground truth.

no ground truth, see Figure 4. Visually the method yields good results even though we do not use more training data, but stick with only 1000 training samples per class. Besides better accuracy, our algorithm also greatly outperforms the baseline in terms of runtime, in spite of the higher-dimensional feature representation. Our system is two orders of magnitude faster, see Table 3. We point out that the comparison should be seen as indicative: the timings for the baseline are those reported by the authors. Their software is also written in C++ (called from Matlab). But it is single-threaded, and it was run on a different, albeit comparable, machine (64 GB RAM, Intel Core i7-3820 quad-core, 3.6 GHz).

#### 4.3 Results on Terrestrial Laser Scans

The terrestrial laser scans are more challenging, for several reasons. The data was captured in both urban and rural environments, so that there is more variety, and a greater degree of occlusion. Also, the training and testing datasets are from different physical locations, and constitute a stricter test of the classifier's ability to generalize. Finally, the surveying-grade scanners used for recording have better angular resolution and higher range, leading to massive variations in point density between nearby and distant objects. There is also more data. The training data has  $\approx 500,000$  points, sampled from a total of  $2 \cdot 10^8$  points in the training scans. The test set has more than  $3 \cdot 10^8$  points. Table 4 shows that our proposed framework is able to cope rather well with the TLS data. Overall accuracy achieved with only base features is 90.3%, at a mean class recall of 79.7%. This results in 74.4%  $F_1$ -score. Recall and precision values for the larger and easier classes are – as expected – lower than for the Paris, on the other hand the precision of smaller classes is significantly better, leading to a higher average  $F_1$ -score. The computation time for the complete test set is 90 minutes, including disk I/O.

These results can be further improved with additional histogram descriptors (Section 3.2.1), especially for the *low vegetation* class,

where the gains in  $F_1$ -score reach 8-9 percent points. Overall, A-SC3D works a bit better than A-SHOT and is to be preferred. With the latter, classification of scanning artifacts on moving objects is less reliable ( $-4$  percent points). However, the gains are rather small, at least for our class nomenclature, and they come at considerable computational cost. The following configuration gave the best trade-off between quality and speed: four shells with radii  $\{1, 2, 4, 16\}$  m, 12 bins per shell, and 4 bins per local histogram of A-SHOT. With these settings the results improve for 6 out of 7 classes, only for *scanning artifacts* the additional descriptors potentially hurt performance. Mean  $F_1$ -score goes up by 1.7 percent points. On the other hand, the feature computation takes  $\approx 3 \times$  longer than for the base features, so adding A-SC3D or A-SHOT quadruples the processing time. Still, even large datasets with tens of millions of points are processed in less than one hour, which we consider acceptable for many applications. Based on our results we do not generally recommend histogram-based descriptors, but note that they could have a significant impact for certain tasks and object classes.

#### 5. CONCLUSION

We have described an efficient pipeline for point-wise classification of 3D point cloud data. The core of our method is an efficient strategy to construct approximate multi-scale neighborhoods in 3D point data. That scheme makes it possible to extract a rich feature representation in very little time, even in the presence of wildly varying point density. As a consequence, the proposed classification system outperforms the state of the art in terms of classification accuracy, while at the same time being fast enough for operational use on a single desktop machine.

There are several directions we would like to explore in future work. One idea is to use larger-scale context or even global CRF-type models to further improve the classification. This may be challenging in terms of computational cost. Alternatively,

Terrestrial Laser Scans	Original Features			Original Features + A-SHOT			Original Features + A-SC3D		
	Recall	Precision	$F_1$	Recall	Precision	$F_1$	Recall	Precision	$F_1$
Man made terrain	0.8758	0.9433	0.9083	0.8757	0.9627	0.9171	0.8788	0.9651	0.9200
Natural terrain	0.8809	0.8921	0.8864	0.9064	0.8774	0.8916	0.9127	0.8747	0.8933
High vegetation	0.9129	0.8102	0.8585	0.8086	0.9254	0.8631	0.8655	0.8785	0.8720
Low vegetation	0.6496	0.4021	0.4967	0.6356	0.5364	0.5818	0.6685	0.5052	0.5755
Buildings	0.9592	0.9878	0.9733	0.9726	0.9813	0.9769	0.9676	0.9859	0.9766
Remaining hard scape	0.7879	0.4878	0.6025	0.8419	0.4743	0.6068	0.8256	0.4824	0.6090
Scanning artefacts	0.5127	0.4595	0.4847	0.3995	0.5039	0.4457	0.4197	0.5684	0.4829
Overall accuracy	0.9028			0.9097			0.9114		
Mean class recall	0.7970			0.7772			0.7912		
Mean $F_1$ -score	0.7443			0.7547			0.7613		

Table 4. Quantitative results for terrestrial laser scans.

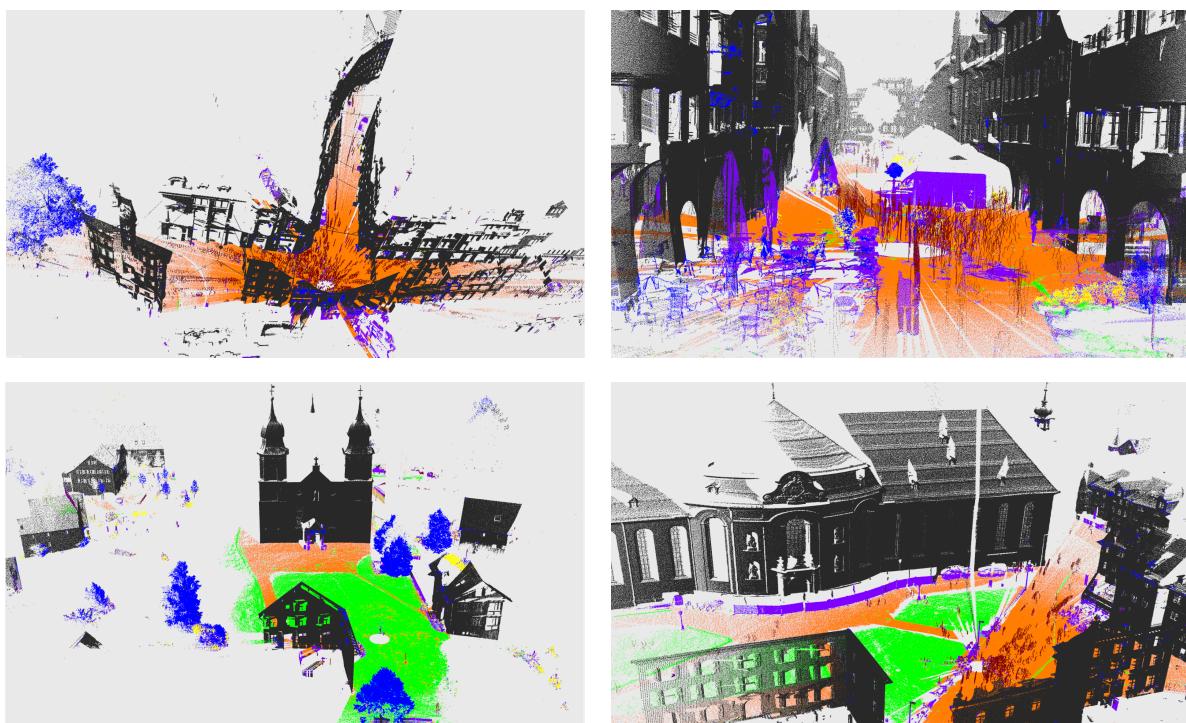


Figure 5. Results for terrestrial laser scans. *Top row:* urban street in St. Gallen (left), market square in Feldkirch (right). *Bottom row:* church in Bildstein (left), cathedral in St. Gallen (right) with classes: **man-made terrain**, **natural terrain**, **high vegetation**, **low vegetation**, **buildings**, **remaining hard scape** and **scanning artefacts**.

one could try to explicitly detect and reconstruct line features to help in delineating the classes. Another possible direction is to sidestep feature design altogether and adapt the recently very successful deep neural networks to point clouds. The challenge here will be to remain efficient in the absence of a regular grid, although the voxel-grid neighborhoods built into our pipeline may provide a good starting point. Finally, on the technical level it would be interesting to port the method to the GPU to further improve speed.

## REFERENCES

- Belongie, S., Malik, J. and Puzicha, J., 2002. Shape matching and object recognition using shape contexts. *IEEE TPAMI* 24(4), pp. 509–522.
- Blomley, R., Weinmann, M., Leitloff, J. and Jutzi, B., 2014. Shape distribution features for point cloud analysis—a geometric histogram approach on multiple scales. *ISPRS Annals*.
- Boyko, A. and Funkhouser, T., 2011. Extracting roads from dense point clouds in large scale urban environment. *ISPRS J Photogrammetry & Remote Sensing* 66(6), pp. 2–12.
- Brodu, N. and Lague, D., 2012. 3d terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS J Photogrammetry & Remote Sensing* 68, pp. 121–134.
- Charaniya, A. P., Manduchi, R. and Lodha, S. K., 2004. Supervised parametric classification of aerial LiDAR data. *CVPR Workshops*.
- Chehata, N., Guo, L. and Mallet, C., 2009. Airborne LiDAR feature selection for urban classification using random forests. *ISPRS Archives*.
- Dalponte, M., Bruzzone, L. and Gianelle, D., 2012. Tree species classification in the Southern Alps based on the fusion of very high geometrical resolution multispectral/hyperspectral images and LiDAR data. *Remote Sensing of Environment* 123, pp. 258–270.

- Demantké, J., Mallet, C., David, N. and Vallet, B., 2011. Dimensionality based scale selection in 3d lidar point clouds. ISPRS Archives.
- Dohan, D., Matejek, B. and Funkhouser, T., 2015. Learning Hierarchical Semantic Segmentation of LIDAR Data. International Conference on 3D Vision.
- Elseberg, J., Borrmann, D. and Nüchter, A., 2013. One billion points in the cloud—an octree for efficient processing of 3d laser scans. ISPRS J Photogrammetry & Remote Sensing 76, pp. 76–88.
- Friedman, J. H., Bentley, J. L. and Finkel, R. A., 1977. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software 3(3), pp. 209–226.
- Frome, A., Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. Recognizing objects in range data using regional point descriptors. ECCV.
- Golovinskiy, A., Kim, V. G. and Funkhouser, T., 2009. Shape-based Recognition of 3D Point Clouds in Urban Environments. ICCV.
- Haala, N., Brenner, C. and Anders, K.-H., 1998. 3d urban GIS from laser altimeter and 2d map data. ISPRS Archives.
- Hackel, T., Wegner, J. D. and Schindler, K., 2016. Contour detection in unstructured 3d point clouds. CVPR.
- He, K., Sun, J. and Tang, X., 2010. Guided image filtering. ECCV.
- Hug, C. and Wehr, A., 1997. Detecting and identifying topographic objects in imaging laser altimetry data. ISPRS Archives.
- Johnson, A. and Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3d scenes. IEEE TPAMI 21(5), pp. 433–449.
- Kim, E. and Medioni, G., 2011. Urban scene understanding from aerial and ground LIDAR data. Machine Vision and Applications 22, pp. 691–703.
- Lafarge, F. and Mallet, C., 2012. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. IJCV 99(1), pp. 69–85.
- Lai, K., Bo, L. and Fox, D., 2014. Unsupervised feature learning for 3d scene labeling. ICRA.
- Lalonde, J.-F., Vandapel, N., Huber, D. and Hebert, M., 2006. Natural terrain classification using three-dimensional ladar data for ground robot mobility. Journal of Field Robotics 23(1), pp. 839–861.
- Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D. and Mitra, N. J., 2011. Globfit: Consistently fitting primitives by discovering global relations. ACM Transactions on Graphics 30(4), pp. 52:1–52:12.
- Maas, H.-G., 1999. The potential of height texture measures for the segmentation of airborne laserscanner data. International Airborne Remote Sensing Conference and Exhibition / Canadian Symposium on Remote Sensing.
- Monnier, F., Vallet, B. and Soheilian, B., 2012. Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system. ISPRS Annals.
- Muja, M. and Lowe, D. G., 2009. Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP.
- Najafi, M., Namin, S. T., Salzmann, M. and Petersson, L., 2014. Non-associative Higher-Order Markov Networks for Point Cloud Classification. ECCV.
- Niemeyer, J., Wegner, J. D., Mallet, C., Rottensteiner, F. and Soergel, U., 2011. Conditional Random Fields for Urban Scene Classification with Full Waveform LiDAR Data. Photogrammetric Image Analysis.
- Orka, H. O., Naesset, E. and Bollandsas, O. M., 2012. Classifying species of individual trees by intensity and structure features derived from airborne laser scanner data. Remote Sensing of Environment 123, pp. 258–270.
- Pauly, M., Keiser, R. and Gross, M., 2003. Multi-scale feature extraction on point-sampled surfaces. Computer Graphics Forum 22(3), pp. 281–289.
- Pu, S. and Vosselman, G., 2009. Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS J Photogrammetry & Remote Sensing 64, pp. 575–584.
- Pu, S., Rutzinger, M., Vosselman, G. and Oude Elberink, S., 2011. Recognizing basic structures from mobile laser scanning data for road inventory studies. ISPRS J Photogrammetry & Remote Sensing 66, pp. 28–39.
- Rottensteiner, F. and Briese, C., 2002. A new method for building extraction in urban areas from high-resolution LIDAR data. ISPRS Archives.
- Rusu, R., Holzbach, A., Blodow, N. and Beetz, M., 2009. Fast geometric point labeling using conditional random fields. IROS.
- Schmidt, A., Niemeyer, J., Rottensteiner, F. and Soergel, U., 2014. Contextual Classification of Full Waveform Lidar Data in the Wadden Sea. IEEE Geoscience and Remote Sensing Letters 11(9), pp. 1614–1618.
- Schnabel, R., Wahl, R. and Klein, R., 2007. Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum 26(2), pp. 214–226.
- Serna, A., Marcotegui, B., Goulette, F. and Deschaud, J.-E., 2014. Paris-rue-madame database: a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. ICPRAM.
- Shapovalov, R., Velizhev, A. and Barinova, O., 2010. Non-associative markov networks for 3D point cloud classification. ISPRS Archives.
- Tombari, F., Salti, S. and Di Stefano, L., 2010. Unique signatures of histograms for local surface description. ECCV.
- Vallet, B., Brédif, M., Serna, A., Marcotegui, B. and Paparoditis, N., 2015. Terramobilita/iquimulus urban point cloud analysis benchmark. Computers & Graphics 49, pp. 126–133.
- Vosselman, G., Gorte, B., Sithole, G. and Rabbani, T., 2004. Recognizing structure in laser scanner point clouds. ISPRS Archives.
- Weinmann, M., Jutzi, B. and Mallet, C., 2013. Feature relevance assessment for the semantic interpretation of 3d point cloud data. ISPRS Annals.
- Weinmann, M., Urban, S., Hinz, S., Jutzi, B. and Mallet, C., 2015. Distinctive 2d and 3d features for automated large-scale scene analysis in urban areas. Computers & Graphics 49, pp. 47–57.
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J., 2015. 3D ShapeNets: A deep representation for volumetric shape modeling. CVPR.
- Xiao, J. and Furukawa, Y., 2014. Reconstructing the world's museums. IJCV 110(3), pp. 243–258.
- Yao, W., Hinz, S. and Stilla, U., 2011. Extraction and motion estimation of vehicles in single-pass airborne LiDAR data towards urban traffic analysis. ISPRS J Photogrammetry & Remote Sensing 66, pp. 260–271.