Analysis of Semantic Labeling of Point Clouds

***Abstract*—One central problem when transforming a pointcloud into a CAD model is to segment points into semantic groups and labeling these groups. Almost every modern approach tries to solve this problem using Deep Learning. Most approaches are following the well-established pattern of splitting a certain dataset into a train- and test set and only exposing the test set to the model in a validation phase. Still, these test sets originate from the same dataset, thus may have very similar structures compared to the training sets. In this analysis, we are investigating the loss of accuracy when applying a model trained on one dataset to the test set of a whole different dataset. On top of that, we investigate the impact of sampling methods (isometric vs concentric) on the resulting accuracy.**

I. Introduction

Renovating a building as a whole requires "as-is" plans for the architect to work on. Sadly, often times the original construction plans are either faulty or not available anymore. Also, most commonly these construction plans are analogue and 2D ("blueprints"). Therefore, the original construction plans are typically a bad basis to work on. This problem can be solved by scanning the building in its current state. Then the resulting pointclouds have to be transformed into a CAD model, which is a non-trivial problem. One essential step in this transformation is to segment the pointlcoud into semantically meaningful and labeled parts (e.g. this set of points represents a table.).

The groundbreaking success of Convolutional (Deep) Neural Networks in classifying, segmenting and labeling images [LeCun et al., 1989; LeCun et al., 1998; Krizhevsky et al., 2012] lead to the hope of a similar success for the same tasks performed on 3D pointclouds. A general training/testing approach, which has shown to work very well, is to obtain a dataset which is as general as possible, split it into two subsets, train on one of them and only expose the test set to the model for validation. To the best knowledge of the authors, all of the current approaches in 3D pointcloud (deep) learning are using this strategy. Unfortunately, most available scanned datasets are typically recorded in one facility (e.g. a hallways, lecture rooms or offices), which typically uses the same type of furniture for most of its rooms or has other similarities/biases, making each individual dataset a bad representative for reality in general. Also, available datasets are often times resampled and contain a lot of outliers. In contrast to these very "nice" circumstances, real world scans often times suffer from sampling artifacts — e.g. concentric rings for lidar scans — and a lot of outliers (created by reflecting surfaces).

In this analysis we aim to investigate the following two things:

1) The generality of current datasets, respectively the generality of resulting models.
2) The effect of different sampling/scanning techniques as well as the impact of real world scanning artifacts.

II. Related Work

A. Non deep learning methods

Approaches using classical methods instead of Deep Learning are quite rare nowadays. Still, we want to highlight one approach from 2014, since the central idea somehow reappears in current methods again.

1) Object detection and classification from large-scale cluttered indoor scans [Mattausch et al., 2014]: The work by Mattausch et al. mainly targets the segmentation of a given pointcloud and doesn't apply labels to these segments. In contrast to most modern approaches, they don't use Deep Learning to segment the pointcloud into semantically meaningful parts. Their main observaiton is, that in most environments (e.g. an office) furniture and objects are repeating, respectively different objects can be considered an instance of the same "class". A good example is a chair, which typically is the same for every desk in an office.

Given a set of pointclouds, their pipeline is the following [Mattausch et al., 2014, Section 3]:

1) **Preprocessing:** In this step, the pointcloud gets converted into as planar as possible patches.
2) **Patch embedding:** The patches are going to be embedded into a high dimensional euclidian space in such a way similar patches are close to each other in this space.
3) **Clustering:** The last step is to cluster close patches into a group, yielding the final segmentation of the whole pointcloud.

The key step here is the patch embedding, since it already locates similar objects close to each other, allowing also "simple" clustering techniques to yield good results in the third step. Also, this technique is variable in its input (number of points) and output (number of segment groups) size. In contrast, Deep Learning approaches are typically bound to a fixed input and output format.

Even though Mattausch et al. don't use Deep Learning, their method can be used to segment pointclouds, making it a possible preprocessing step to a Deep Learning approach working on single items instead of full pointclouds. Additionally, this approach might be used to create "proxy objects", which could be created using the union of each item within one group. That way, possible holes within one instance of an object can be filled, increasing the chance a Deep Learning approach correctly labels the object.

B. Voxel/Image based methods

The first approaches to solve 3D classification tasks tried to map 2D techniques into the 3D world.

1) Multi-view convolutional neural networks for 3d shape recognition [Su et al., 2015]: Inspired by the great results of convolutional neural networks in image segmentation and classification, Su et al. created a way to use exactly these techniques to work on 3D objects. In contrast to the other approaches discussed in this section, this one doesn't use pointclouds as inputs, but meshed data. Converting pointclouds into a mesh represenation is a problem for itself, but one which is quite well researched, as long as the scan has a resonable quality and very few shadows or holes (e.g. by using [Lorensen and Cline, 1987]).

The main idea is to render the given mesh from different perspectives and let each image be processed by a (2D) convolutional neural network (CNN). The results of these CNNs are then pooled and fed into another convolutional neural networks. The last CNN combines the features given by the previous CNNs and makes a decision, which label is the most likely one. The authors are reporting accuracies up to 90% (working on the ModelNet [Wu et al., 2015] dataset with 40 labels), which was way better than any 3D classifier could do back then. One drawback of this approach is that it needs a lot of

views/renders to yield these good results (for the 90% the authors report to have used 80 views [Su et al., 2015, Table 1]). Creating these renders may consume a lot of resources.

This approach can also be combined with the one introduced by Mattausch et al., using a meshed version of the above mentioned "proxy objects".

2) Voxnet: A 3d convolutional neural network for real-time object recognition [Maturana and Scherer, 2015]: Pixel based images got some very nice properties, one of which is the equidistance between each pixel. This property allows for easy convoltion methods, since no expensive range lookups have to be performed. Voxels share this property with pixels, making it convenient to use 3D CNNs voxels. VoxNet was one of the first Deep (Convolutional) Neural Net working on voxels. Its architecture aims for real-time applications, like robots equipped with range sensors. Therefore, it doesn't fit a completely the architectural problem described above, but one could still use VoxNet e.g. by simulating a movement through the scanned building.

Maturana and Scherer decided to transform a given segment of a pointlcoud into an occupancy grid first. The segment can be fairly simple, like a box cut out of a big pointcloud. The occupancy grid gets then created by calculating the probability a certain voxel within this (3D) grid is occupied. In their implementation, they differ between occupied, free and unkown space. This third option can be a critical information for path-planning, where "unkown" should be treated with care, while for the architectural case it may be sufficient to fill the unknown spaces with estimates. After creating the occupancy grid — they used 32^3 voxels for their experiments [Maturana and Scherer, 2015, Section III B.]) — the data was fed into a CNN. Their resulting CNN is fairly small with less than 1 million parameters to optimize [Maturana and Scherer, 2015, Section III E.]. As a comparison, modern CNNs working on images got 10-145 million parameters [Zoph et al., 2018, Table 2]. Nevertheless, Maturana and Scherer report prediction accuracies up to 92% (working on the ModelNet dataset with 10 labels.), outperforming state-of-the-art models of that time [Maturana and Scherer, 2015, Table III].

C. Point based methods

While the two above mentioned approaches were trying to convert a given pointcloud into some other format, which better suits known techniques, the following approaches introduced new techniques fitting raw pointclouds as an input. In contrast to voxelized scenes, pointclouds are irregular and unordered. Futhermore, the central tool in signal processing (fast fourier transformation) is not available as it is for images or voxeles.

1) PointNet/PointNet++[Qi et al., 2016][Qi et al., 2017]: "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" is known to be the pioneer work in Deep Learning on pointclouds. In their work the authors are describing different problems and how they solve them [Qi et al., 2016, Section 4].

1) **Unordered**: A pointset has no intrisic order. The authors tackle this problem by applying a symmetric function to transformed points. The transformation is being approximated by a mini-network. As a symmetric function, they chose a max function, whichs output is independent of its input order.
2) **Interaction among points**: A single point within a pointcloud only has a "meaning" if you also take its surrounding neighbors into account. Qi et al. are therefore using both, local and global information to classify a point and are also able to calculate features like normals on a per point basis.
3) **Invariance under transformations**: The authors point out, rotating or translating an object should not affect its segmentation or classification. By predicting an affine transformation via a mini-network and applying it, the current set of points gets aligned into a canonical space.

During training, the authors chose to subsample the pointcloud into 1024 points, scale them into a unit sphere, randomly rotating the points along the up-axis and jitter the position of each point with gaussian noise ($\sigma = 0.02$) [Qi et al., 2016, Section 5.1]. The authors report a classification accuracy of 89.2% (working on the ModelNet dataset with 40 labels), which is beyond VoxNets performance (83% on the ModelNet dataset with 40 labels).
In contrast to the approaches presented before PointNet is able to classify, but also to segment pointclouds. Every of the previous approaches was able to do either of the two, but not both.

"PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space" is an extension to PointNet, which addresses some shortcomings in the original design. The main difference between PointNet and PointNet++ is the hierarchical approach being used instead of a global max pooling, enabling the authors to extract larger and larger abstractions, instead of one single local to global one [Qi et al., 2017, Section 3.2]. This change has increased the classifcation accuracy to 91.9% (again, working on the ModelNet dataset using 40 labels).

2) Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs [Landrieu and Simonovsky, 2017]: Landrieu and Simonovsky developed an algorithm to segment and classify pointclouds at a time. In contrast to PointNet(++), which was already able to perform both tasks, this approach does both at the same time. The core idea is similar to the approach presented by Mattausch et al. and is split into three parts as well [Landrieu and Simonovsky, 2017, Section 3]:

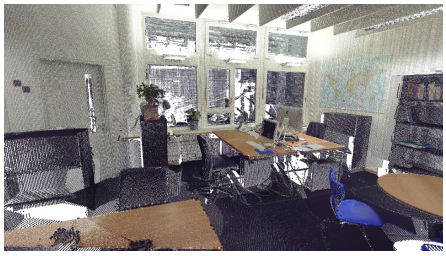
1) **Geometrically homogeneous partition:** The first step is to partition the pointcloud into superpoints. The idea of superpoints is the 3D point analogy to superpixels. Note that in comparison to the work done by Mattausch et al., these superpoints don't have to be planar.
2) **Superpoint embedding:** Based on these superpoints and the symmetric voronoi adjacency graph, the "superpoint graph" is constructed [Landrieu and Simonovsky, 2017, Section 3.2].
3) **Contextual segmentation:** Applying Deep Learning graph convolution algorithms will then solve the classification problem.

The authors are making use of the S3DIS dataset (see Section III-A), using a cross validation technique to train and test their models. Since the S3DIS dataset is split into 6 parts, the authors are training 6 models using one area as a testing set and the other five areas as training sets. Working on the S3DIS dataset, the authors report an overall accuracy of 82.9% (in compariosn: PointNet achieved an overall accuracy of 78.5% on the same dataset) [Landrieu and Simonovsky, 2017, Table 3].
The precision of segmentation is usually measured in "intersection over union" (given in %), also called the Jaccard Index [Real and Vargas, 1996]. The superpoint graph approach has a mean intersection over union precision (mIoU) of 54.06%, while PointNet was only able to achieve 47.6% (again, working on the S3DIS dataset) [Landrieu and Simonovsky, 2017, Table 3].

3) PointCNN: Convolution on X-transformed points [Li et al., 2018]: One very recent publication by Li et al. aims to bring classical convolution applied on regular grids (e.g. an image) to irregular sampled inputs (like pointclouds). Similar to PointNet(++), PointCNN moves each current point into a local coordinate system, making the whole system translation invariant. In contrast to PointNet(++), which can be interpreted as point convolution as well, PointCNN doesn't make use of a symmetric function. Instead, the authors decided to

(a) The S3DIS (pointcloud) dataset. Please note the holes underneath most chairs, which are due to the scanning technique.

(b) The ScanNet (mesh) dataset. Please note the hole in the middle of the room and the missing ceiling, which are due to the scanning technique.



(c) The IFI (pointcloud) dataset. Please note the reflections in the window, which are typical lidar scanning artifacts.

Figure 1: Example screenshots of the different datasets being used.

explicitely weight each point and then permute it [Li et al., 2018, Section 3.2]. All these operations, like the weighting and permutation as well as lifting the points into a high dimensional feature space, is learned by multi-layer perceptrons (MLPs).

The authors report an accuracy of 92.2% (working on the ModelNet dataset with 40 labels), outperforming previous methods like PointNet++ (91.9%) by a small margine [Li et al., 2018, Table 1].

III. Datasets

Generally, the amount of labeled 3D data is very small, compared to the amount of images designed to be used for Deep Learning. In this analysis, we make use of three datasets.

A. S3DIS [Armeni et al., 2016]

The S3DIS dataset[1] is provided by the stanford university. They used a Matterport[2] scanner, scanning offices, hallways, learning areas and other indoor locations of their university[Armeni et al., 2016, Section 5.1]. These scans were split into 6 areas, which are themselves again dvided into different rooms/sections, each represented by a single labeled pointcloud. The pointclouds are evenly sampled and don't show any sampling artifacts, except shadows. The S3DIS dataset has 13 labels. See Figure (1a) for a visualization.

B. ScanNet [Dai et al., 2017]

The ScanNet dataset[3] was created by crowd sourced scanning (20 workers) and labeling (more than 500 workers), and has more than 1500 scans [Dai et al., 2017, Section 4]. It contains different representations of the same scenes. For example, RGB-D scans, which could be used to simulate a live input for SLAM techniques. In this work we are using the high resolution labeled meshes. Just like the S3DIS dataset, ScanNet doesn't suffer from artifacts, except shadows/holes. To transform the mesh into a pointcloud, we scanned each mesh virtually, using two different methods. One method aimed to yield an isometrically sampled pointcloud (comparable to the S3DIS ones), while the other one simulated a lidar scanner, introducing concentric sampling artifacts. The isometric scan used each vertex of the mesh as well as the barycentric midpoints of each triangle as a scanned point. The virtual lidar scanner got placed systematically into the room, scanning each mesh from 3-5 different locations. The virtual scanner didn't introduce any artifacts besides the concentric sampling. It's still an interesting future task to investigate the effect of other scanning artifacts. We will refer to the isometrically sampled version of this dataset as Iso and to the lidar-like sampled one as Lidar. The ScanNet dataset has 1357 labels (many of them are semantically similar) and also labelmaps to the most common label sets (e.g. NYU40, MPCAT40 and others). See Figure (1b) for a visualization.

C. IFI

The IFI dataset[4] has been recorded in the University of Zurich, constisting of several offices and one lab. As a recording device a 3D lidar[5] scanner was used, introducing a lot of typical artifacts like concentric sampling, reflections, shadows/holes, outliers and depth-noise. This dataset has no labels. Therefore, the quality of estimated labels has to be evaluated on a visual level instead of a statistical one. See Figure (1c) for a visualization.

IV. Analysis

In this analysis we focus on the superpoint graph approach presented by Landrieu and Simonovsky. We chose this one, because it's able to segment and label at the same time, which — to our knowledge — none of the other approaches can do that at the same time. We are working on three different datasets, while one of them is represented twice (isometrically sampled and lidar-like sampled). To better differentiate between trained model names and dataset names, we will refer to the datasets by their name and to the models trained on a dataset with an M_ followed by the name of the dataset (e.g. M_S3DIS).

A. Preparations

To investigate the effect of different sampling techniques, we decided to train two different models, one on an isometrically sampled dataset and one on a lidar-like sampled one. We chose these datasets to be the S3DIS dataset as well as the Lidar one. This training happened in the classical fashion, but since we want to apply the trained models to the respective other datasets as well and still be able to compare the results, we had to change the datasets labelsets to be identical. Since we didn't want to make semantical decisions (like "Is a stool also a chair?"), we decided to only keep those labels which are present in both labelsets and put the rest into one bin, which we named "others". This way we ended up having 12 labels, including "others". Please refer to Table II (in the Appendix) for a full list of labels.

B. Training

We performed the training on the S3DIS dataset exactly as described in the superpoint graph paper [Landrieu and Simonovsky,

[1] http://buildingparser.stanford.edu/dataset.html
[2] https://matterport.com/
[3] http://www.scan-net.org/
[4] https://www.ifi.uzh.ch/en/vmml/research/datasets.html
[5] Faro Focus 3D laser range scanner

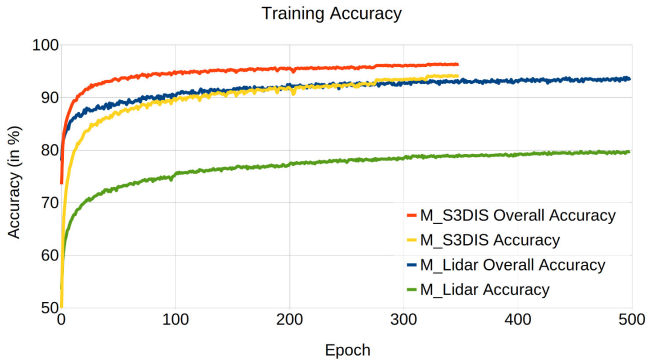


Figure 2: Training accuracy and overall accuracy for the M_S3DIS and M_Lidar models.

			S3DIS	Lidar	Iso
with others	M_S3DIS	acc	87.686	48.655	48.412
		oacc	91.709	69.869	67.428
		aIoU	70.284	27.383	26.343
	M_Lidar	acc	68.268	81.509	80.449
		oacc	79.716	90.702	88.223
		aIoU	40.357	53.692	50.119
without others	M_S3DIS	acc	87.373	47.861	47.447
		oacc	91.050	69.918	67.578
		aIoU	69.404	23.309	22.644
	M_Lidar	acc	66.946	81.44	80.334
		oacc	78.640	90.512	88.105
		aIoU	36.576	49.399	46.013

Table I: Test results (given in %) of applying every model onto every dataset. The IFI dataset has no ground truth labels, which is why it's not listed here. The "with others" and "without others" rows are indicating if the "others" label has been included into the statistics or not.

2017, Section 4.2], using cross validation to split the dataset into a train and test set. Since this results in six models, we decided to perform each test on each model and average the results.

The training on the Lidar dataset was performed as similar as possible to the S3DIS training. Since the dataset is a bit larger, we chose to use 500 training epochs instead of 350, and also to adjust the regularization to the new number of epochs. In contrast to the S3DIS training, we split the dataset into two parts (train/test), resulting in one model instead of six. Figure (2) shows the training accuracy for the M_S3DIS model as well as the M_Lidar one.

C. Testing

As tests we applied each model to its own test set as well as the test sets of the three other dataset instances. That way we can observe the difference in sampling (applying M_Lidar to Lidar vs Iso) as well as the difference in completely new environments (applying M_S3DIS to Iso). As mentioned above, the IFI dataset has no labels. Therefore, we can't provide any precise numbers, but only a qualitative analysis.

Since the S3DIS dataset is split into six parts, of which each part will serve as a testing set once, we decided to average the values for these tests.

Each test has been performed using 10 multisamples. As resulting measures we were interested in the (average) test accuracy, the overall test accuracy as well as the average intersection over union precisions. You can find the per-class intersection over union values for each test in Table III. When calculating the average, we excluded classes which have not been seen/predicted. Since the label "others" might add some undesired bias, we calculated the precisions once with the label included and once without.

V. Results

Table I shows the accuracy, overall accuracy and average intersection over unit for each model applied to each dataset with and without the "others" label. Besides the table, Figures (3) and (4) (in the Appendix) visualize the outcome as well. Surprisingly, the M_S3DIS model performed slightly better on the IFI dataset than the M_Lidar model. Still, both classifications are very unprecise — which was expectable.

A. Conclusion

The results allow for several conclusions

- The "others" label has a small effect on the accuracy. Thus, it's unlikely the network has learned to only label things as "others", which could have marked a local minimum in the learning process.
- The M_Lidar model performs better on the S3DIS dataset than the M_S3DIS model on the Lidar dataset. This might be an indicator for the ScanNet dataset to be more divers/general than the S3DIS one.
- The sampling method (isometric or lidar-like) has an impact of a few percent. This is still a lot and confirms the need to treat this problem, as current publications (e.g. [Hermosilla et al., 2018]) already do.
- The preprocessing sometimes fails to segment difficult transitions (e.g. a white door next to a white wall), giving the classifier no chance to label the resulting superpoint correctly (since it has more than one label in it). This effect is shown in Figure (5c) (in the Appendix).
- Applying any model to the IFI dataset yields bad segmentations and classifications. Only the floor and the ceiling have mostly been labeled correctly (while huge parts of the room have falsely been labeled as "floor" too). Detecting ceiling, walls and floor are already covered quite wel by classical approaches (e.g. [Mura et al., 2016]), yielding better results than the ones shown here.

VI. Future Work

This analysis was bound to one method. It would still be interesting to investigate other methods, to see if the relations found here are general, or specific to this approach. Furthermore, it would be very interesting to see how the different precisions behave when adding more sampling errors to the virtual lidar scanner (like depth noise, outliers or reflections). Possible future work could try to make use of non Deep Learning methods like the one presented by Mattausch et al. to extract segments of a pointcloud to perform a labeling approach just on this segment.

References

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., and Ropinski, T. (2018). Monte carlo convolution for learning on non-uniformly sampled point clouds. In *SIGGRAPH Asia 2018 Technical Papers*, page 235. ACM.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Landrieu, L. and Simonovsky, M. (2017). Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings ACM SIGGRAPH*, pages 163–169.

Mattausch, O., Panozzo, D., Mura, C., Sorkine-Hornung, O., and Pajarola, R. (2014). Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum*, 33(2):11–21.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE.

Mura, C., Mattausch, O., and Pajarola, R. (2016). Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Computer Graphics Forum*, 35(7):179–188.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.

Real, R. and Vargas, J. M. (1996). The probabilistic basis of jaccard's index of similarity. *Systematic biology*, 45(3):380–385.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.

VII. Appendix

Class number	Label name	label color name	RGB-triple
0	others	black	(0, 0, 0)
1	ceiling	light blue	(88, 181, 225)
2	floor	magenta	(180, 18, 112)
3	wall	light green	(45, 212, 96)
4	column	rose	(235, 112, 213)
5	beam	dark green	(151, 200, 76)
6	window	pruple	(171, 57, 249)
7	door	grass green	(89, 131, 34)
8	table	dark purple	(114, 68, 185)
9	chair	orange	(251, 144, 70)
10	stairs	dark blue	(73, 64, 110)
11	sofa	turquoise	(46, 206, 183)

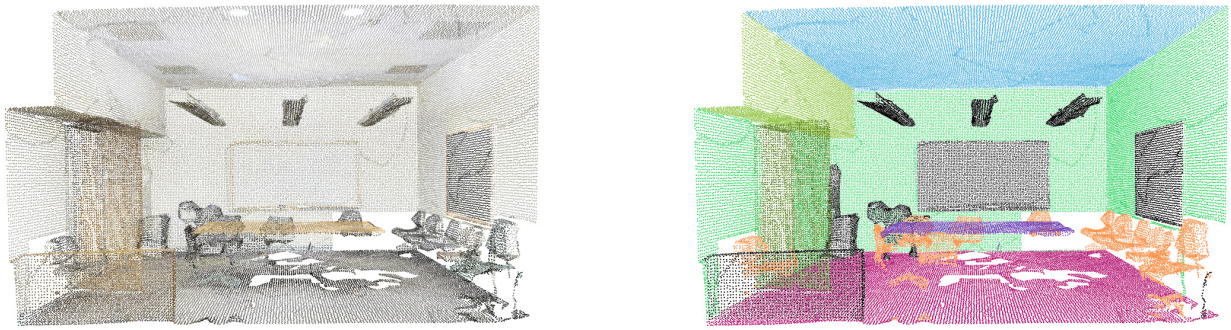
Table II: The used list of labels for all datasets (except the IFI one).

	M_S3DIS			M_Lidar		
	S3DIS	Lidar	Iso	S3DIS	Lidar	Iso
others	93.811	70.54	66.028	83.118	96.971	91.925
ceiling	95.197	79.363	77.479	94.546	98.098	95.712
floor	86.387	63.681	60.696	71.083	83.546	78.613
wall	47.222	5.247	3.919	0.0	10.642	20.358
column	65.275	0.0	0.0	0.0	0.0	0.0
beam	54.242	2.832	10.711	18.306	36.584	29.19
window	73.332	20.703	17.237	37.593	39.739	30.615
door	80.732	40.302	32.37	49.645	84.877	74.128
table	80.065	11.2	13.515	45.82	72.169	75.235
chair	59.563	2.912	1.656	8.274	8.671	2.3
stairs	58.525	11.223	10.812	30.375	59.665	53.976
sofa	0.0	0.0	0.0	0.0	0.0	0.0

Table III: Per class intersection over union precisions (in %).

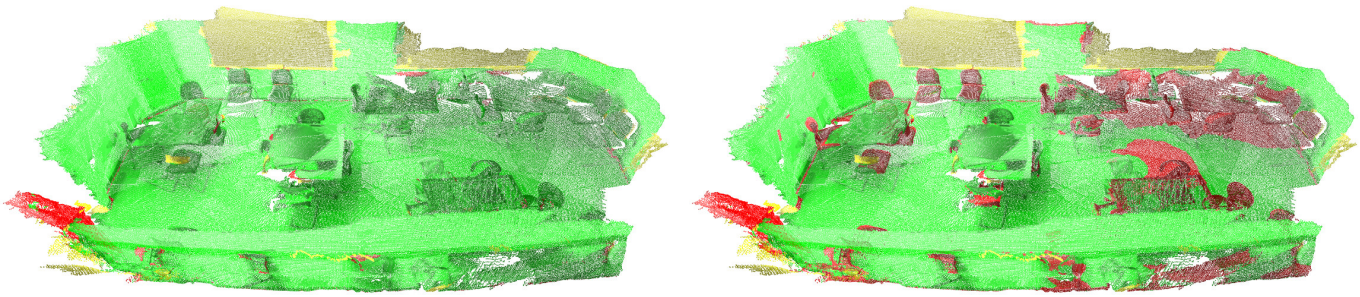
(a) An example of the Lidar dataset.

(b) An example of the Iso dataset.

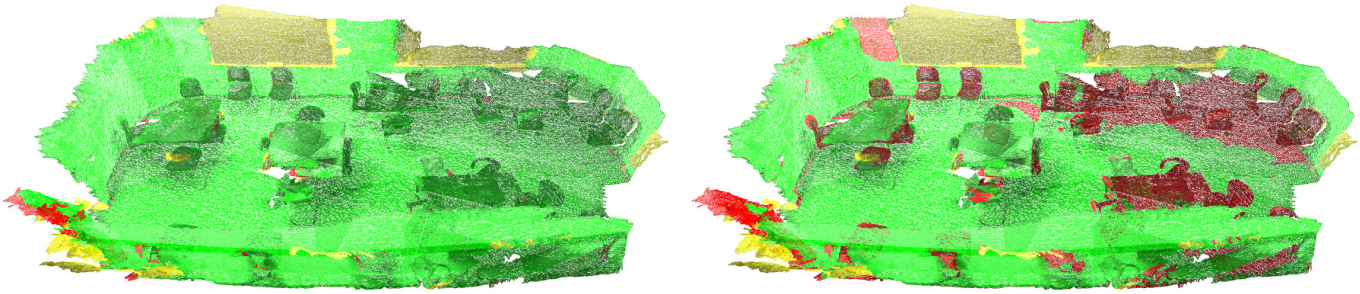


(c) An example of the S3DIS dataset.

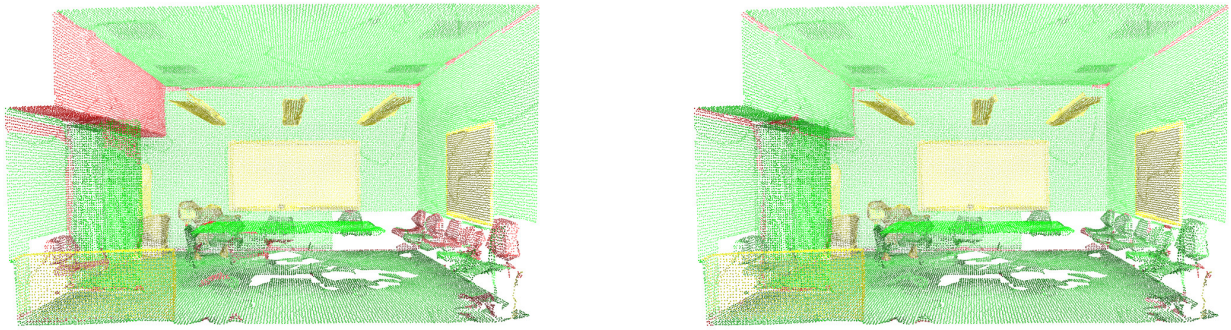
Figure 3: Another view on three of the four datasets. Left: RGB pointcloud; Right: Ground truth labels colorcoded. See Table II for the label-color map.



(a) Results for the Lidar dataset.

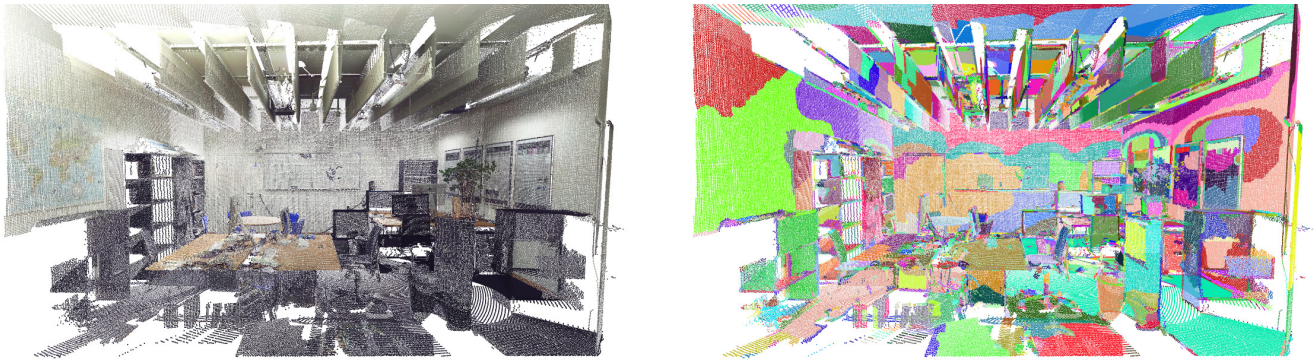


(b) Results for the Iso dataset.

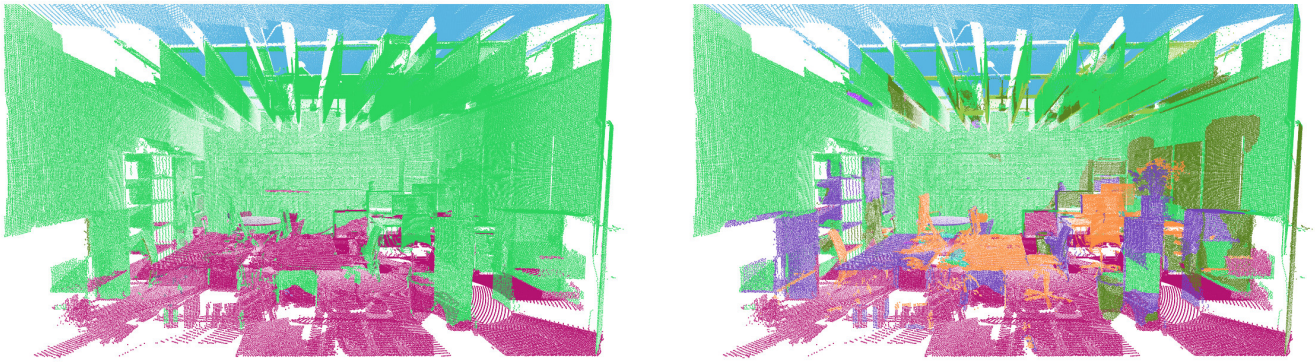


(c) Results for the S3DIS dataset.

Figure 4: Error image of applying the learned models to different datasets. Left: M_Lidar; Right: M_S3DIS. The error colors are green (correctly labeled), yellow (original label was “others”) and red (wrong labeled). If the original label was “others”, we don’t evaluate the predicted label in these images at all. In the statistical analysis we do.



(a) An overview. Left: RGB pointcloud; Right: Partitioning result. The partitioning colors are randomly chosen and unrelated to the label colors.



(b) The predicted labels by the two models. Left: M_Lidar; Right: M_S3DIS. Interestingly, the M_S3DIS model performs a little bit better on this dataset.

(c) A specific problem yielded by the partitioning/preprocessing. The door, wall and board are all part of a huge cluster (dark purple), making it impossible for the classifier to label this patch correctly.

Figure 5: Results of applying both models to the IFI dataset.