Supplementary Material
SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences

Jens Behley* Martin Garbade* Andres Milioto Jan Quenzel
Sven Behnke Cyrill Stachniss Juergen Gall
University of Bonn, Germany

A. Consistent Labels for LiDAR Sequences

In this section, we explain the implementation of our point cloud labeling tool in more detail and the rationale behind our decision to subdivide the sequences spatially, but not temporally, for getting consistently labeled point cloud sequences. The labeling tool itself was critical to provide the amount of scans with such fine-grained labels.

In summary, we developed an OpenGL-based labeling tool, which exploits parallelization on the GPU. The main challenge is the visualization of vast amounts of point data, but also processing these at the same time, while reaching responsiveness that allows the annotator to label interactively the aggregated point clouds. Figure 1 shows our point cloud annotation program visualizing an aggregated point cloud of over 20 million points. We provide a wide range of tools for annotation, like a brush, a polygon tool, and different filtering methods to hide selected labels. Even with that many points, we are still able to maintain interactive labeling capabilities. Changes to the label of the points inside the aggregated point cloud are reflected in the individual scans, which enables high consistency of the labels over time.

Since we are labeling each point, we are able to annotate objects, even with complex occlusions, more precisely than just using bounding volumes [11]. For instance, we ensured that ground points below a car are labeled accordingly, which was enabled by our filtering capabilities of the annotation tool.

To accelerate the search for points that must be labeled, we used a projective approach to assign labels. To this end, we determine for each point the two-dimensional projection on the screen and then determine for the projection if the point is near to the clicked position (in case of the brush) or inside the selected polygon. Therefore, annotators had to ensure that they did not choose a view that essentially destroyed previously assigned points.

Usually, an annotator performed the following cycle to annotate points: (1) mark points with a specific label and (2) filter points with that label. Due to the filtering of already labeled points, one can resolve occlusions and furthermore ensure that the aforementioned projective labeling does not destroy already labeled points.

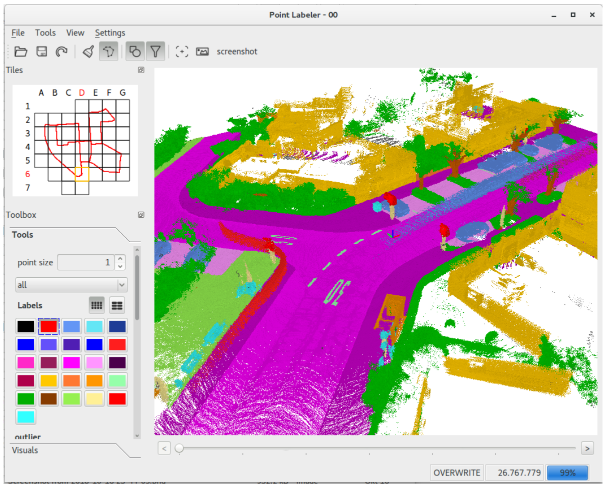

Figure 1. Point cloud labeling tool. In the upper left corner the user sees the tile and the sensor's path indicated by the red trajectory.

Tile-Based Labeling. An important detail is the aforementioned spatial subdivision of the complete aggregated point cloud into tiles (also shown in the left upper part of Figure 1). Initially, we simply rendered all scans in a range of timestamps, say $100 - 150$, and then moved on the next part, say $150-200$. However, this leads quickly to inconsistencies in the labels, since scans from such parts still overlap and therefore must be relabeled to match labels from before. Since we, furthermore, encounter loop closures with a considerable temporal distance, this overlap can even happen between parts of the sequences that are not temporally close, which even more complicated the task.

Thus, it quickly became apparent that such an additional effort to ensure consistent labels would lead to an unreasonable complicated annotations process and consequently to insufficient results. Therefore, we decided to subdivide the sequence spatially into tiles, where each tile contains all points from scans overlapping with this tile. Consistency at the boundaries between tiles was achieved by having a small overlap between the tiles, which enabled to consistently continue the labels from one tile into another neighboring tile.

Moving Objects. We annotated all moving objects, i.e., car, truck, person, bicyclist, and motorcyclist, and each moving object is represented by a different class to distinguish it from its corresponding non-moving class. In our case, we assigned an object the corresponding moving class when it moved at some point in time while observing it with the sensor.

Since moving objects will appear at different places when aggregating scans captured from different sensor locations, we had to take special care to annotate moving objects. This is especially challenging, when multiple types of vehicles move on the same lane, like in most of the encountered highway scenes. We annotated moving objects either by filtering ground points or by labeling each scan individually, which was often necessary to label points of tires of a car and bicycles or the feet of persons. But scan-by-scan labeling was also necessary in aforementioned cases where multiple vehicles of different type drive on the same lane. The labeling of moving objects often was the first step when annotating a tile, since this allowed the annotator to filter all moving points and then concentrate on the static parts of the environment.

B. Basis of the Dataset

The basis of our dataset is data from the KITTI Vision Benchmark [3], which is still the largest collection of data also used in autonomous driving at the time of writing. The KITTI dataset is the basis of many experimental evaluations in different contexts and was extended by novel tasks or additional data over time. Thus, we decided to build upon this legacy and also enable synergies between our annotations and other parts and tasks of the KITTI Vision Benchmark.

We particularly decided to use the Odometry Benchmark to enable usage of the annotation data with this task. We expect that exploiting semantic information in the odometry estimation is an interesting avenue for future research. However, also other tasks of the KITTI Vision Benchmark might profit from our annotations and the pre-trained models we will publish on the dataset website.

Nevertheless, we hope that our effort and the availability of the point labeling tool will enable others to replicate our work on future publicly available datasets from an automotive LiDAR.

C. Class Definition

In the process of labeling such large amounts of data, we had to decide which classes we want to be annotated at some point in time. In general, we followed the class definitions and selection of the *Mapillary Vistas* dataset [5] and *Cityscapes* [2] dataset, but did some simplifications and adjustments for the data source used.

First, we do not explicitly consider a *rider* class for persons riding a motorcycle or a bicycle, since the available point clouds do not provide the density for a single scan to distinguish the person riding a vehicle. Furthermore, we get for such classes only moving examples and therefore cannot easily aggregate the point clouds to increase the fidelity of the point cloud and make it easier to distinguish the rider of a vehicle and the vehicle.

The classes *other-structure*, *other-vehicle*, and *other-object* are fallback classes of their respective root category in unclear cases or missing classes, since this simplified the labeling process and might be used to distinguish these categories further in future.

Annotators often annotated some object or part of the scene and then hide the labeled points to avoid overwriting or removing the labels. Thus, assigning the fallback class in ambiguous cases or cases where a specific class was missing made it possible to simply hide that class to avoid overwriting it. If we had instructed the annotators to label such parts as unlabeled, it would have caused problems to consistently label the point clouds.

We furthermore distinguished between moving and non-moving vehicles and humans, *i.e.*, a vehicle or human gets the ‘moving’ tag if it moved in some consecutive scans while being observed by the LiDAR sensor.

In summary, we annotated 28 classes and all annotated classes with their respective definitions are listed in Table 1 on the next page.

D. Baseline Setup

We modified the available implementations such that the methods could be trained and evaluated on our large-scale dataset with very sparse point clouds due to the LiDAR sensor. Note that most of these approaches have so far only been evaluated on small RGB-D indoor datasets.

We restricted the number of points within a single scan due to memory limitations on some approaches [6, 7] to 50 000 via random sampling.

For SPLATNet, we used the SPLATNet$_{3D}$[1] architecture from [8]. The input consisted per point of the 3D position and its normal. The normals were previously estimated given 30 closest neighbors.

With TangentConv[2] we used the existing configuration

[1] https://github.com/NVlabs/splatnet

[2] https://github.com/tatarchm/tangent_conv

cat.	class	definition
Ground-related	road	Drivable areas where cars are allowed to drive on including service lanes, bike lanes, crossed areas on the street. Only the road surface is labeled excluding the curb.
	sidewalk	Areas used mainly by pedestrians, bicycles, but not meant for driving with a car. This includes curbs and spaces where you are not allowed to drive faster than $5\,\mathrm{km}\,/\,\mathrm{h}$. Private driveways are also labeled as sidewalk. Here cars should also not drive with regular speeds (such as 30 or $50\,\mathrm{km}\,/\,\mathrm{h}$).
	parking	Areas meant explicitly for parking and that are clearly separated from sidewalk and road by means of a small curb. If unclear then *other-ground* or *sidewalk* can be selected. Garages are labeled as *building* and not as parking.
	other-ground	This label is chosen whenever a distinction between sidewalk and terrain is unclear. It includes (paved/plastered) traffic islands which are not meant for walking. Also the paved parts of a gas station are not meant for parking.
structures	building	The whole building including building walls, doors, windows, stairs, etc. Garages count as building.
	other-structure	This includes other vertical structures, like tunnel walls, bridge posts, scaffolding on a building from a construction site or bus stops with a roof.
vehicle	car	Cars, jeeps, SUVs, vans with a continuous body shape (i.e. the driver cabin and cargo compartment are one) are included.
	truck	Trucks, vans with a body that is separate from the driver cabin, pickup trucks, as well as their attached trailers.
	bicycle	Bicycles without the cyclist or possibly other passengers. If the bicycle is driven by a person or a person stands nearby the vehicle, we label it as bicyclist.
	motorcycle	Motorcycles, mopeds without the driver or other passengers. Includes also motorcycles covered by a cover. If the motorcycle is driven by a person or a person stands nearby the vehicle, we label it as motorcyclist.
	other-vehicle	Caravans, Trailers and fallback category for vehicles not explicitly defined otherwise in the meta category *vehicle*. Included are buses intended for 9+ persons for public or long-distance transport. This further includes all vehicles moving on rails, e.g., trams, trains.
nature	vegetation	Vegetation are all bushes, shrubs, foliage, and other clearly identifiable vegetation.
	trunk	The tree trunk is labeled as *trunk* separately from the treetop which gets the label *vegetation*.
	terrain	Grass and all other types of horizontal spreading vegetation, including soil.
human	person	Humans moving by their own legs, sitting, or any unusual pose, but not meant to drive a vehicle.
	bicyclist	Humans driving a bicycle or standing in close range to a bicycle (within *arm reach*). We do not distinguish between riders and bicyclist.
	motorcyclist	Humans driving a motorcycle or standing in close range to a motorcycle (within *arm reach*).
object	fence	Separators, like fences, small walls and crash barriers.
	pole	Lamp posts and the poles of traffic signs.
	traffic sign	Traffic sign excluding its mounting. Spurious points in a layer in front and behind the traffic sign are also labeled as traffic sign and not as outlier.
	other-object	Fallback category that includes advertising columns.
outlier	outlier	Outlier are caused by reflections or inaccuracies in the deskewing of scans, where it is unclear where the points came from.

Table 1. Class definitions.

Approach	road	sidewalk	parking	other-ground	building	car	car (moving)	truck	truck (moving)	bicycle	motorcycle	other-vehicle	other-vehicle (moving)	vegetation	trunk	terrain	person	person (moving)	bicyclist	bicyclist (moving)	motorcyclist	motorcyclist (moving)	fence	pole	traffic-sign	mIoU
TangentConv	83.9	64.0	38.3	15.3	85.8	84.9	40.3	21.1	42.2	2.0	18.2	18.5	30.1	79.5	43.2	56.7	1.6	6.4	0.0	1.1	0.0	1.9	49.1	36.4	31.2	34.1
DarkNet53Seg	91.6	75.3	64.9	27.5	85.2	84.1	61.5	20.0	37.8	30.4	32.9	20.7	28.9	78.4	50.7	64.8	7.5	15.2	0.0	14.1	0.0	0.2	56.5	38.1	53.3	41.6

Table 2. IoU results using a sequence of multiple past scans (in %).

	Scene Completion			Semantic Scene Completion																			
Approach	precision	recall	IoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mIoU
SSCNet	31.71	83.40	29.83	27.55	16.99	15.60	6.04	20.88	10.35	1.79	0	0	0.11	25.77	11.88	18.16	0	0	0	14.40	7.90	3.67	9.53
TS3D	31.58	84.18	29.81	28.00	16.98	15.65	4.86	23.19	10.72	2.39	0	0	0.19	24.73	12.46	18.32	0.03	0.05	0	13.23	6.98	3.52	9.54
TS3D + DarkNet53Seg	25.85	88.25	24.99	27.53	18.51	18.89	6.58	22.05	8.04	2.19	0.08	0.02	3.96	19.48	12.85	20.22	2.33	0.61	0.01	15.79	7.57	6.99	10.19
TS3D + DarkNet53Seg + SATNet	80.52	57.65	50.60	62.20	31.57	23.29	6.46	34.12	30.70	4.85	0	0	0.07	40.12	21.88	33.09	0	0	0	24.05	16.89	6.94	17.70

Table 3. Results for scene completion and class-wise results for semantic scene completion (in %).

	Approach	scan size	projected	learning rate	epochs trained	converged
single scan	PointNet	50 000	-	$3 \cdot 10^{-4} \times 0.9^{\text{epoch}}$	33	✓
	PointNet++	45 000	-	$3 \cdot 10^{-3} \times 0.9^{\text{epoch}}$	25	✓
	TangentConv	120 000	-	$1 \cdot 10^{-4}$	10	✓
	SPLATNet	50 000	-	$1 \cdot 10^{-3}$	20	-
	SqueezeSeg	64×2048	✓	$1 \cdot 10^{-2} \times 0.99^{\text{epoch}}$	200	✓
	DarkNet21Seg	64×2048	✓	$1 \cdot 10^{-3} \times 0.99^{\text{epoch}}$	40	✓
	DarkNet53Seg	64×2048	✓	$1 \cdot 10^{-3} \times 0.99^{\text{epoch}}$	120	✓
multi scan	TangentConv	500 000	-		5*	✓
	DarkNet53Seg64	64×2048	✓	$1 \cdot 10^{-3} \times 0.99^{\text{epoch}}$	40*	✓

Table 4. Approach statistics. $*$ in number of epochs means that it was started from the pretrained weights of the single scan version.

for Semantic3D. We sped up the training and validation procedures by precomputing scan batches and added asynchronous data loading. Complete single scans were provided during training. In the multi scan experiment we fixed the number of points per batch to 500 000 due to memory constraints and started training from the single scan weights.

For SqueezeSeg [10] and its Darknet backbone equivalents, we used a spherical projection of the scans in the same way as the original SqueezeSeg approach. The projection contains 64 lines in height corresponding with the separate beams of the sensor, and extrapolating the configuration of SqueezeSeg which only uses the front 90° and a horizontal resolution of 512, we use 2048 for the entire scan. Because some points are duplicated in this sampling process, we always keep the closest range value, and in inference of each scan we iterate over the entire point list and check it's semantic value in the output grid.

An overview of the used parameters is given in Table 4. We furthermore provide the number of trained epochs and if we could get a results which seems to be converged in the given amount of time.

E. Results using Multiple Scans

The full per class IoU results for the multiple scans experiment are listed in Table 2. As already mentioned in the main text, we generally observe that the IoU of static classes is mostly unaffected by the availability of multiple past scans. To some extent, the IoU for some classes increases slightly. The drop in performance in terms of mIoU is mainly caused by the additional challenge to correctly separate moving and non-moving classes.

F. Semantic Scene Completion

Table 3 shows the class-wise results for semantic scene completion as well as precision and recall for scene completion. One can see that TS3D + DarkNet53Seg performs slightly better than SSCNet and TS3D. Note that DarkNet53Seg has been pretrained on the exact same classes as required for semantic scene completion. TS3D on the other hand uses DeepLab v2 (ResNet-101) [1] pretrained on the Cityscapes [2] dataset, which does not differentiate between classes such as other-ground, parking or trunk for example. Another reason might be that 2D semantic labels projected back onto the point cloud is not very accurate especially at object boundaries, where labels often bleed onto

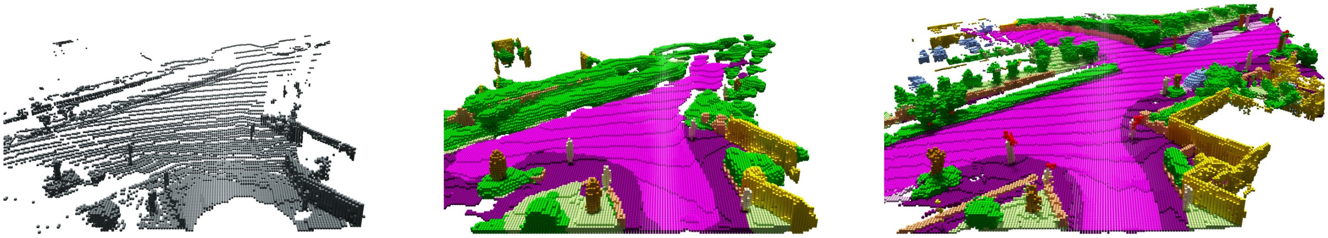

Figure 2. Qualitative results for the semantic scene completion approach TS3D + DarkNet53Seg + SATNet. Left: Input volume. Middle: Network prediction. Right: Ground truth. Due to memory limitations the inference has to be done in six steps on overlapping subvolumes. The subvolumes are consequently fused to obtain the final result.

distant objects. This is because in the 2D projection, they are close to each other, a problem that is inherent to the projection method. The best approach (TS3D + DarkNet53Seg + SATNet) outperforms the other approaches significantly (+20.77% IoU on scene completion and +7.51% mIoU on semantic scene completion). As mentioned above, it is the only approach capable of producing high resolution outputs. This approach however suffers from huge memory consumption. Therefore, during training the input volume is randomly cropped to volumes of grid size $64 \times 64 \times 32$ while during inference, each volume gets divided into 6 overlapping blocks of size $90 \times 138 \times 32$ for which the inference is performed individually. The individual blocks are subsequently fused to obtain the final result. Figure 2 shows an example result of this approach.

Rare classes like bicycle, motorcycle, motorcyclist, and person are not or almost not recognized. This suggests that these classes are potentially hard to recognize, as they represent a small and rare signal in the SemanticKITTI data.

G. Qualitative Results

Figure 3 shows qualitative results for the evaluated baseline approaches on a scan from the validation data. Here we show the spherical projections of the results to enable an easier comparison of the results.

With increasing performance in terms of mean IoU (top to bottom), see also Table 2 of the paper, we see that ground points get better separated into the classes sidewalk, road, and parking. In particular, parking areas need a lot of contextual information and also information from neighboring points, since often a small curb distinguishes the parking area from the road.

In general, one can see definitely an increased accuracy for smaller objects like the poles on the right side of the image, which indicates that the extra parameters of the models with the largest capacity (25 million as in the case of DarkNet21Seg and 50 million as in the case of Darknet53Seg) are needed to distinguish smaller classes and class with few examples.

H. Dataset and Baseline Access API

Along with the annotations and the labeling tool, we also provide a public API implemented in Python.

In contrast to our labeling tool, which is intended for allowing users to easily extend this dataset, and generate others for other purposes, this API is intended to be used to easily access the data, calculate statistics, evaluate metrics, and access several implementations of different state-of-the-art semantic segmentation approaches. We hope that this API will serve as a baseline to implement new point cloud semantic segmentation approaches, and will provide a common framework to evaluate them, and compare them more transparently with other methods. The choice of Python as the underlying language for the API is that it is the current language of choice for the front end for deep learning framework developers, and therefore, for deep learning practitioners.

Figure 4 gives an overview of the labeled sequences showing the estimated trajectories and the aggregated point cloud over the whole sequence.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation withDeep Convolutional Nets, Atrous Convolution,and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2018. 4

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 2

[4] Loic Landrieu and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In

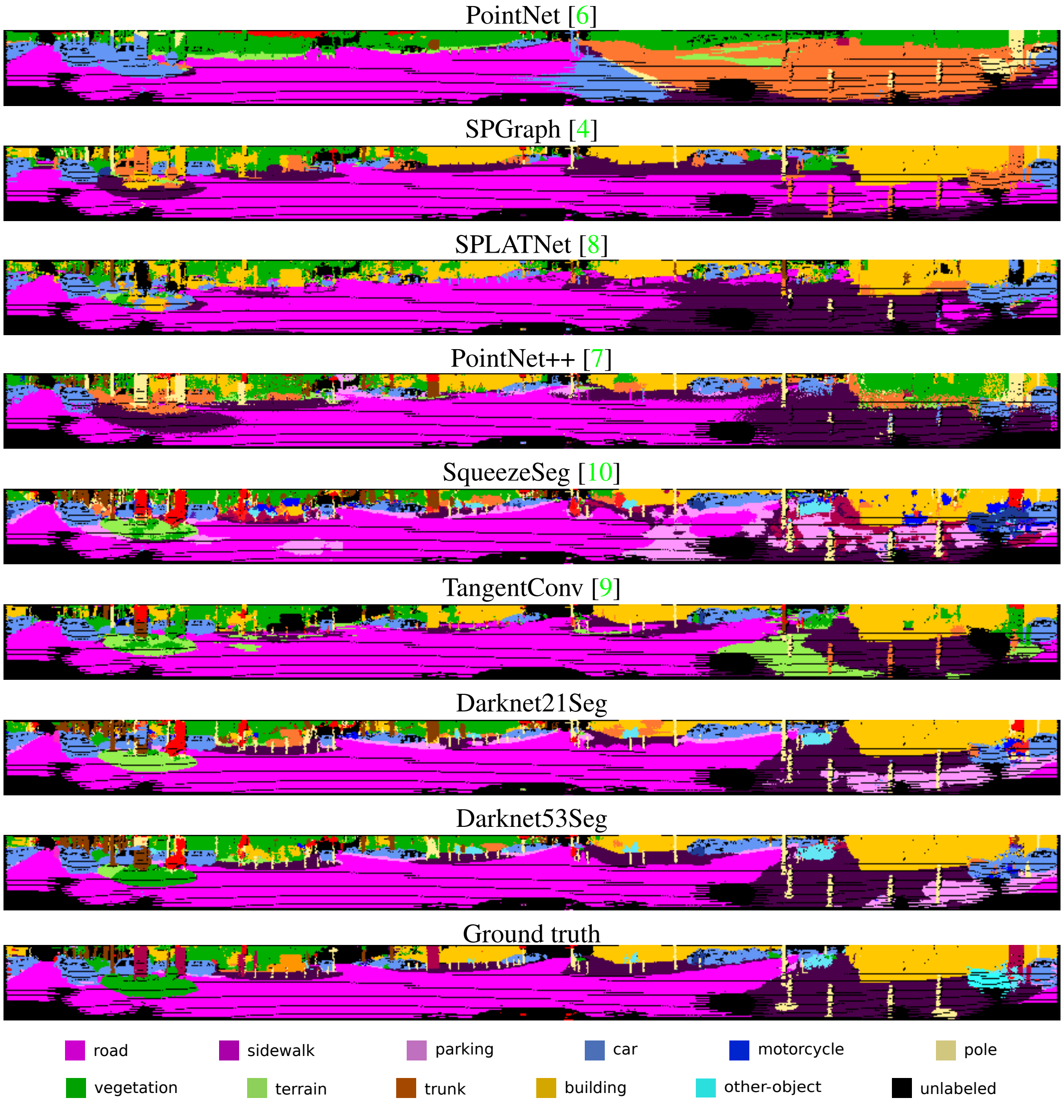

Figure 3. Examples of inference for all methods. The point clouds were projected to 2D using a spherical projection to make the comparison easier.

Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2018. 6

[5] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017. 2

[6] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 6

[7] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2017. 2, 6

[8] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6

[9] Maxim Tatarchenko, Jaesik Park, Vladen Koltun, and Qian-Yi Zhou. Tangent Convolutions for Dense Prediction in 3D. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[10] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018. 4, 6

[11] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

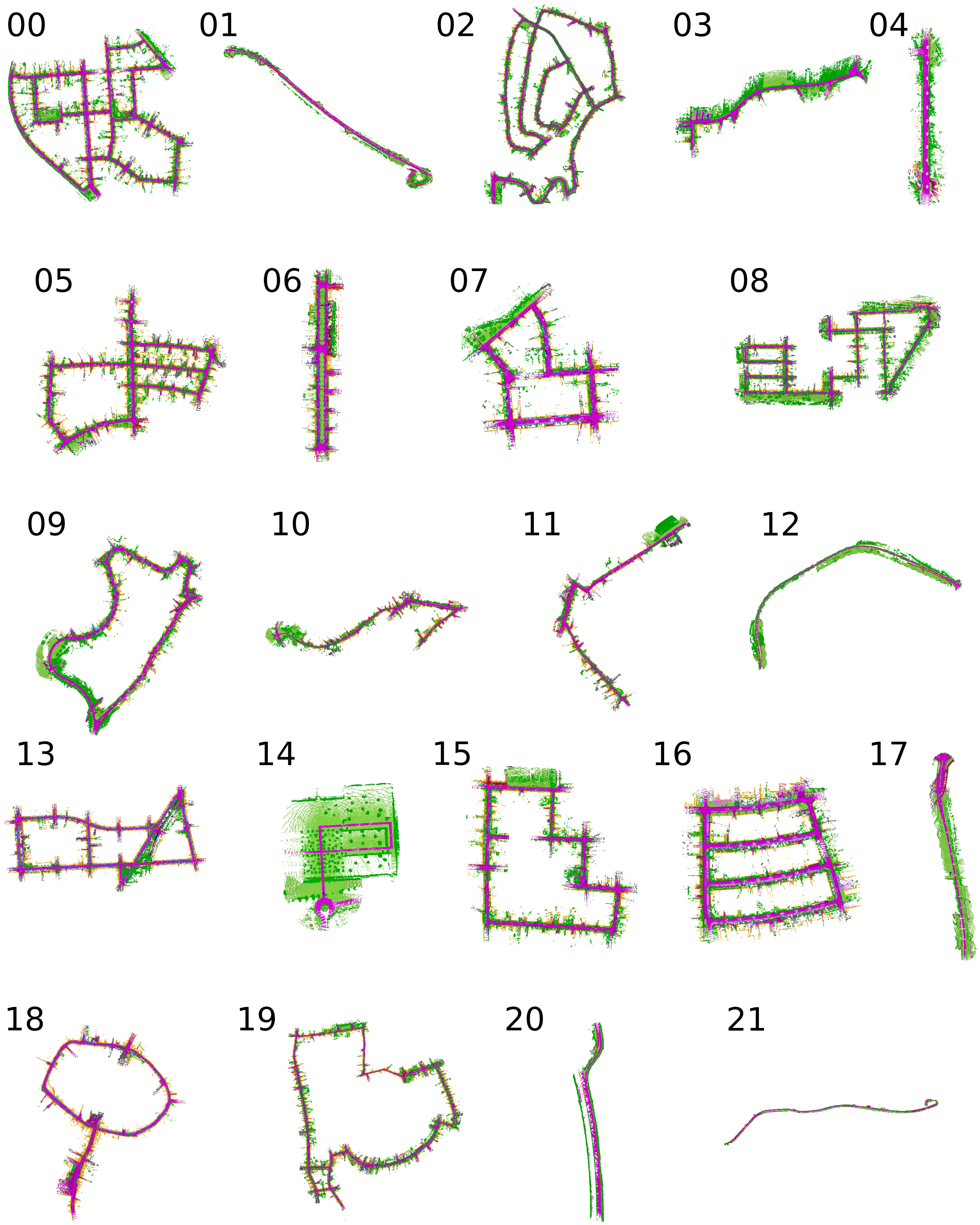

Figure 4. Qualitative overview of labeled sequences and trajectories.