KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D

Yiyi Liao Jun Xie Andreas Geiger

Abstract—For the last few decades, several major subfields of artificial intelligence including computer vision, graphics, and robotics have progressed largely independently from each other. Recently, however, the community has realized that progress towards robust intelligent systems such as self-driving cars requires a concerted effort across the different fields. This motivated us to develop KITTI-360, successor of the popular KITTI dataset. KITTI-360 is a suburban driving dataset which comprises richer input modalities, comprehensive semantic instance annotations and accurate localization to facilitate research at the intersection of vision, graphics and robotics. For efficient annotation, we created a tool to label 3D scenes with bounding primitives and developed a model that transfers this information into the 2D image domain, resulting in over 150k images and 1B 3D points with coherent semantic instance annotations across 2D and 3D. Moreover, we established benchmarks and baselines for several tasks relevant to mobile perception, encompassing problems from computer vision, graphics, and robotics on the same dataset, e.g., semantic scene understanding, novel view synthesis and semantic SLAM. KITTI-360 will enable progress at the intersection of these research areas and thus contribute towards solving one of today's grand challenges: the development of fully autonomous self-driving systems.

Index Terms—Point Cloud Labeling, Semantic Label Transfer, Scene Understanding, Self-Driving, Datasets, Performance Evaluation

1 Introduction

One of the pioneering works in *computer vision* can be traced back to Larry Roberts' "Blocks World" in the 1960s [85], which aimed at identifying individual objects and inferring the 3D structure of simple shapes from 2D images. With the goal of understanding a scene from visual cues, computer vision was viewed as a comparably easy first step towards solving higher-level reasoning tasks in *robotics* at that time (e.g., the MIT copy demo [104]). Albeit being seemingly easy for humans, robustly perceiving geometry and semantics from images proved hard for machines due to the high complexity of real-world environments. Thus, in the 1980s, computer vision and robotics evolved into their own, largely independent research fields. Only recently, the communities have realized that it is impossible to solve one without the other, e.g., in the context of self-driving [47]. Similarly, computer vision's interaction with *computer graphics* emerged in the 1990s [92] and has gained traction over the last decade, in particular in areas such as neural and image-based rendering [64]. These advances can in turn benefit robotics as simulation will be crucial for training and validating the next generation of robotic systems.

The converging trend of vision, graphics, and robotics motivates us to create a new dataset, KITTI-360, that addresses tasks at the intersection of these fields with a focus on autonomous driving. While the KITTI dataset [34] has pushed the state-of-the-art in computer vision algorithms forward, it does not contain dense and complete semantic labels. Thus, many interesting interdisciplinary tasks, e.g., synthesizing novel view images jointly with semantics or reconstructing large-scale semantic maps, cannot be evaluated on KITTI. Moreover, the captured perspective front images provide only a partial view of the scene and the 3D information provided by the LiDAR sensor is very sparse. The GPS localization of KITTI is reliable but does not reach sub-pixel accuracy when fusing multiple frames. With KITTI-360 we address these shortcomings by providing a new dataset with more comprehensive semantic/instance labels in 2D and 3D, richer 360° sensory information (fisheye images and pushbroom laser scans), very accurate and geo-localized vehicle and camera poses, and a series of new challenging benchmarks, see Fig. 1 for an overview.

A key challenge towards building such a dataset is to obtain coherent dense and comprehensive semantics in 2D and 3D. Many existing datasets are annotated in the 2D image domain where pixel-wise labeling requires up to 60 minutes per image for a human annotator [6]. Other datasets [8], [96], [119] are annotated in 3D space while ignoring information in the 2D image domain. A few datasets [18] offer labels in both 2D and 3D. However, annotation is conducted independently, thus duplicating the labeling effort.

In this paper, we propose an alternative approach that leverages coarse 3D annotations to significantly simplify the dense annotation task and establish coherent labels in both 2D and 3D space. Moreover, this yields a unique *instance* index for each object in the scene across all 2D video frames. Specifically, we build a WebGL-based annotation tool that allows for annotating both *static* and *dynamic* scene elements directly in 3D using simple primitives. This approach has several advantages over labeling in 2D: First, objects often project into several video frames, thus lowering annotation efforts considerably. Further, the obtained 2D instance annotations are temporally coherent as they are associated with a single physical 3D object. Finally, our 3D annotations

• *Y. Liao is with the Autonomous Vision Group, University of Tübingen and Max Planck Institute for Intelligent Systems, Tübingen, Germany, and Zhejiang University. J. Xie is with Google Research. A. Geiger is with the Autonomous Vision Group, University of Tübingen and Max Planck Institute for Intelligent Systems, Tübingen, Germany. E-mails: yiyi.liao@tue.mpg.de, junx@google.com, a.geiger@uni-tuebingen.de.*

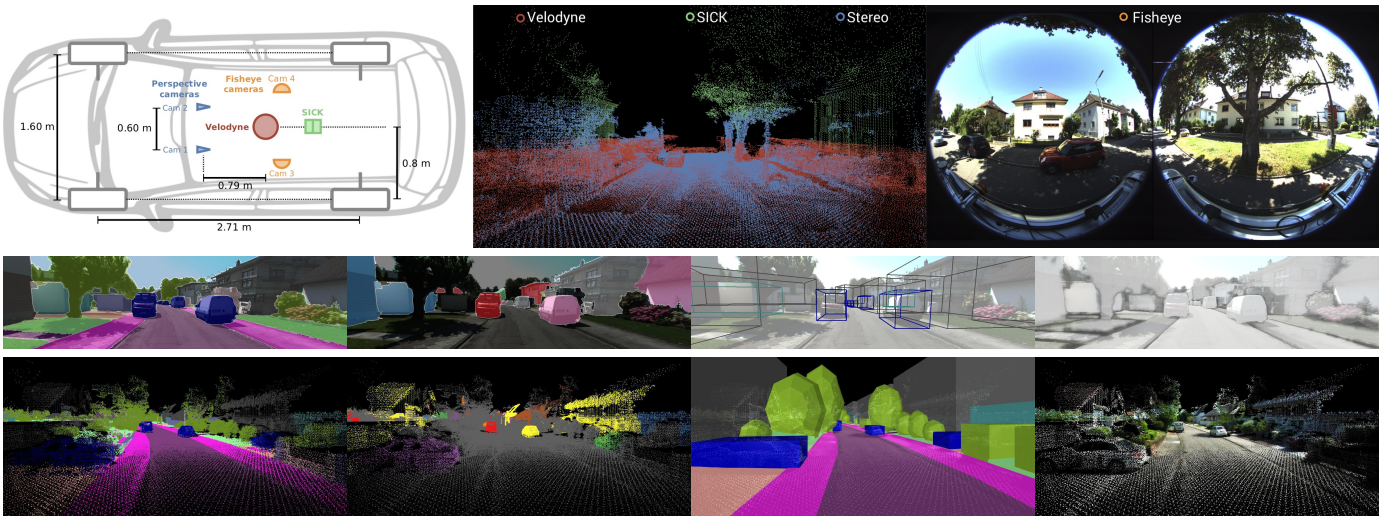


Fig. 1: **KITTI-360.** Our dataset contains rich sensor modalities, including a perspective stereo camera, a pair of fisheye cameras, a Velodyne and a SICK laser scanning unit which together enable 360° scene perception. We release comprehensive annotations including consistent semantic and instance labels for every 2D image pixel and 3D point.

covering the full 3D scene are useful on their own, e.g., for reasoning in 3D [35], [114] or to enrich 2D annotations with approximate 3D geometry.

However, obtaining dense and accurate pixel-wise 2D labels and point-wise 3D labels from sparse, noisy point clouds and coarse 3D annotations is a challenging task. Towards solving this problem, we propose a non-local multi-field CRF model which reasons jointly about semantic and instance labels of all 3D points and 2D pixels. Our approach also leverages learning-based methods to provide dense semantic and instance priors in the 2D image domain. As evidenced by our experiments, our method outperforms label propagation methods operating purely in 2D as well as pure learning-based approaches. Furthermore, the probabilistic nature of our model allows for estimating label uncertainties which can be used to increase label accuracy when only a subset of the pixels require a label.

From the annotated dataset, we derive several benchmarks and baselines with novel and challenging tasks at the intersection of vision, graphics and robotics which we believe are crucial for making progress towards the grand challenge of fully autonomous driving. Our *semantic scene understanding* benchmark includes tasks for 2D/3D recognition and semantic scene completion. The former requires predicting a semantic/instance label for the visible part of the scene, while the latter aims for joint geometric completion and semantic perception that can benefit higher-level reasoning, e.g., control and planning. In our *novel view synthesis* benchmark, we establish a challenging task that requires synthesis of both RGB appearance and semantic labels at a given novel viewpoint, aiming to foster research on building fully labeled simulation environments from real-world images. Lastly, our *semantic SLAM* benchmark evaluates vehicle localization as well as geometric and semantic 3D reconstruction over long sequences.

We summarize the contributions of this paper as follows:

- We present a novel georegistered dataset of suburban scenes recorded by a moving platform. The dataset comprises over 300k images and 80k laser scans.
- We create and release a WebGL-based annotation tool that allows for labeling street scenes in 3D space. Exploiting our annotation tool, we obtain 3D annotations for all static and dynamic scene elements.
- We propose a method which transfers these labels from 3D into 2D, yielding pixel-wise semantic instance annotations. We validate our approach in ablation studies and demonstrate its potential with respect to several 2D and 3D baselines.
- Enabled by our dense and coherent semantic instance annotations in both 2D and 3D as well as accurate vehicle and camera poses, we establish an online benchmark with novel and challenging tasks at the intersection of computer vision, graphics and robotics. We believe that our dataset and benchmarks will complement existing datasets and foster novel research towards solving the grand goal of full autonomy.

This journal paper is an extension of a conference paper published at CVPR 2016 [107]. In comparison to [107], we 1) extend our annotation tool and update our inference algorithm to support the annotation of dynamic objects; 2) provide a detailed description of the annotation tool and process; 3) establish new online benchmarks with held-out test data on a set of challenging tasks; 4) propose and evaluate several baselines to bootstrap the leaderboards and assess the difficulties of the tasks. We make our dataset[1], utility scripts[2] and annotation tool[3] publicly available.

2 Related Work

In this section, we first discuss existing datasets in the context of autonomous driving, followed by a review of current methods for efficient (semi-automatic) label annotation.

1. http://www.cvlibs.net/datasets/kitti-360
2. https://github.com/autonomousvision/kitti360scripts
3. https://github.com/autonomousvision/kitti360labeltool

Dataset	2D Annotations			3D Annotations					Coherency		Test
	#Smt. Img.	#Ins. Img.	Dense	#Smt. Pts.	#Ins. Pts.	FoV Azm.	FoV Plr.	#3D Bbox	Temporal	3D-2D	Server
CamVid [13]	631	–	✓	–	–	–	–	–	✓	–	–
DUS [1]	1k	–	✓	–	–	–	–	–	✓	–	–
CityScape (fine) [25]	5k	5k	✓	–	–	–	–	–	–	–	✓
CityScape (coarse) [25]	20k	20k	–	–	–	–	–	–	–	–	✓
Mapillary Vistas [70]	25k	25k	✓	–	–	–	–	–	–	–	–
CityScape-VPS [52]	3k	3k	✓	–	–	–	–	–	✓	–	–
KITTI-STEP [103]	19k	19k	✓	–	–	–	–	–	✓	–	✓
WoodScape [111]	10k	10k	✓	–	–	–	–	–	✓	–	–
Toronto-3D [96]	–	–	–	78.3M	–	**360°**	40°	–	–	–	–
Paris-Lille-3D [88]	–	–	–	143.1M	–	**360°**	40°	–	–	–	✓
DublinCity [119]	–	–	–	260M	–	-	-	–	–	–	–
Semantic3D.net [39]	–	–	–	4.0B	–	**360°**	**180°**	–	–	–	✓
SemanticKITTI [8]	–	–	–	**4.5B**	–	**360°**	26.8°	–	–	–	✓
Argoverse [21]	–	–	–	–	–	–	–	993k	–	–	–
Lyft [50]	–	–	–	–	–	–	–	**1.3M**	–	–	–
Waymo [94]	–	–	–	–	–	–	–	12M	–	–	✓
A*3D [75]	–	–	–	–	–	–	–	230k	–	–	–
KITTI [34]	200	200	✓	–	–	–	–	200k	–	–	✓
ApolloScape [45]	144k	90k	✓	–	–	–	–	70k	–	–	✓
nuScenes [18]	93k	93k	–	1.2B	78.9M	**360°**	40°	1.2M	–	–	✓
A2D2 [36]	41k	41k	✓	387.1M	23.8M	60°	30°	43k	–	✓	–
SemKITTI-DVPS [80]	23k	23k	–	**4.5B**	**400M**	**360°**	26.8°	–	✓	✓	✓
KITTI-360 (Ours)	**2× 78k**	**2× 78k**	✓	1.0B	172.4M	**360°**	120°	68k	✓	✓	✓

TABLE 1: **Overview of Publicly Available Datasets.** For pixel-level 2D annotations, we compare the number of semantic maps (#Smt. Img.), the number of instance maps (#Ins. Img.) and the density of the 2D semantic maps (Density). Note that the proposed KITTI-360 dataset includes images from both left and right view of the stereo camera. For 3D annotation, we show the number of semantically labeled points (#Smt. Pts.), the number of points with instance labels (#Ins. Pts.), the field of view with 3D semantic annotations at both azimuthal and polar directions (FoV Azm. and FoV Plr.), and the number of 3D bounding boxes (#3D Bbox). We further compare temporal consistency and 3D-to-2D consistency of the instance labels. The last row indicates whether the dataset hosts an online evaluation benchmark with held-out ground truth.

2.1 Datasets

Indoor Video Datasets: Several datasets provide annotations for video sequences captured in indoor scenes [20], [26], [93], [105]. The SUN RGB-D dataset [93] provides labeled 2D polygons as well as 3D cuboids for 10k indoor RGB-D images. In a closely related work, ScanNet [26] is annotated in 3D with its 2D labels directly obtained from 3D-to-2D projection based on the dense depth from RGB-D sensors. In this work, we focus on outdoor street scenes where 3D observations are much more sparse, posing a challenging task for 3D-to-2D label transfer.

Outdoor Datasets: A number of outdoor datasets of driving scenes have been released in the literature [8], [9], [13], [18], [25], [34], [36], [45], [52], [56], [62], [65], [70], [80], [84], [96], [99], [103], [111], [119]. We summarize the most related ones in Table 1, categorized by whether they offer labels in the 2D image domain or in 3D space.

For datasets focusing on 2D labels, CamVid [13] is the first dataset for semantic segmentation in the context of self-driving. However, CamVid does not provide instance labels and only a very limited number of frames. Both Cityscapes [25] and Mapillary Vistas [70] release thousands of manually annotated 2D images. However, they do not offer temporally coherent semantic instance annotations. Recently, Cityscape-VPS [52] extends Cityscapes by providing semantic instance labels for every 5 frames. Furthermore, KITTI-STEP [103] offers spatially and temporally dense semantic instance annotations for the KITTI tracking dataset [34]. While aforementioned works focus on perspective images, WoodScape [111] releases semantic instance annotations of fisheye images. Our dataset differs from the above in that we provide not only temporally coherent semantic instance annotations for perspective images, but also omnidirectional imagery, 3D laser scans, and 3D annotations which are useful for 3D reasoning. While [25] focuses on inner-city scenes, our dataset comprises mainly suburban areas, thus both datasets complement each other (we use the same label definition to facilitate research).

Another line of works provides labels in 3D space. Toronto-3D [96], Paris-Lille-3D [88] and DublinCity [119] offer annotated point clouds collected from urban environments. Semantic3D.net [39] presents a large-scale dataset with 4 billion points, labeled with 8 semantic categories. SemanticKITTI [8] provides semantic labels for raw laser scans in KITTI, resulting in 4.5 billion labeled 3D points in 28 classes. Instead of focusing on point cloud semantic classification, Argoverse [21], Lyft [50], Waymo [94], and A*3D [75] offer 2D/3D bounding boxes and establish benchmarks for 2D/3D detection and tracking[4]. In contrast to KITTI-360, the aforementioned datasets either lack dense annotations in images or they do not have per-point 3D annotations of stuff classes.

Our dataset provides labels for both 2D images and corresponding 3D points. Within this category, KITTI [34] provides dense semantic information on 200 images and 200k 3D bounding boxes. However, KITTI does not provide dense (per-point) 3D labels on the point cloud. Closely related to our work, ApolloScape [45] annotates static scene elements in the 3D space[5] and projects them to the 2D image space, followed by manual annotation of dynamic objects in

4. We refer to 2D annotations as pixel-level annotations in Table 1. 2D bounding boxes are not included.

5. The 3D annotation has not been released yet.

images. In this work, we annotate both static and dynamic objects in 3D, providing coherent annotations for dynamic objects both in 2D and 3D. More recently, nuScenes [18] and A2D2 [36] released labels in both 2D and 3D. However, the labels of nuScenes are manually and independently annotated in 2D and 3D, and not every pixel is labeled in 2D. In contrast, we propose to leverage labels in the 3D space to infer dense labels in the image domain, thus providing consistent labels across 2D and 3D space. A2D2 [36] labels 2D images and maps 2D labels to 3D to obtain per-point 3D labels. Thus, the 3D labels are limited to a small FoV of the cameras in the azimuthal direction. While A2D2 also provides 3D bounding boxes, all of them are within the FoV of the forward-facing camera. We instead offer per-point 3D labels and 3D bounding boxes within an azimuthal FoV of 360°. A concurrent work, SemKITTI-DVPS [80] provides labels in both 2D and 3D by projecting the 3D labels of SemanticKITTI to images. Compared to the projected sparse 2D labels of SemKITTI-DVPS, KITTI-360 offers dense pixel-wise labels and additionally provides 3D bounding boxes.

There also exist several synthetic urban datasets [17], [32], [82], [86]. However, there still exists a significant perceptual gap between the virtual and real domains [43], [97], making synthetic-to-real generalization difficult.

Benchmarks: Recently, evaluation benchmarks have been widely recognized by the community. Some of the previously mentioned datasets also provide online evaluation benchmarks and held-out test data for different tasks. For instance, Cityscapes [25] offers a benchmark suite for pixel and instance-level semantic segmentation as well as 3D vehicle detection. SemanticKITTI [4], [8] hosts lidar segmentation challenges to predict the category of every point. For datasets including both 2D and 3D annotations, KITTI [33], nuScenes [18], and ApolloScape [45] provide benchmarks on a set of vision tasks including detection, stereo, localization, multi-object tracking, and segmentation in both 2D and 3D, etc. Moving beyond the established tasks, KITTI-360 provides novel benchmarks and will hold new challenges, e.g., on novel view semantic synthesis and semantic SLAM, to foster new progress towards full autonomy.

2.2 Methods

Efficient Annotation: Many works have attempted to reduce the per pixel annotation time of individual images, including classical methods [38], [60] and learning based methods [2], [3], [19], [59]. While all of these methods focus on annotating images individually, we are interested in annotating 2D video sequences as well as 3D scenes. There is also a growing interest in autolabeling 3D shapes or 3D bounding boxes [79], [112]. These methods are only applicable to a specific class, e.g., vehicles. We instead annotate the full 3D scene and aim to obtain coherent per-pixel 2D annotations and per-point 3D annotations.

2D Label Propagation: Compared to annotating individual images, video sequences offer the advantage of temporal coherence between adjacent frames. Label propagation techniques exploit this fact by transferring labels from a sparse set of annotated keyframes to all unlabeled frames based on color and motion information. While in some works a single foreground object is assumed [46], here we focus on methods that can handle multiple object categories. Towards this goal, [6] and [15] propose a coupled Bayesian network based on video epitomes and semantic regions to propagate label information between two annotated keyframes. [118] proposes a joint propagation strategy with synthesized training samples. To better account for errors in label propagation, [68] proposes a hierarchy of local classifiers for this task and [5] leverages a mixture-of-tree model for temporal association. The work of [16] leverages label propagation as a data augmentation scheme and demonstrate improved performance on semantic segmentation. Optical flow is also commonly used for semantic video label transfer. [31] uses optical flow of adjacent frames to warp network representations across time and thus propagates labels from previous frames to the current one. [117] proposes to run a convolutional sub-network only on sparse keyframes and propagate the deep feature maps to other frames via flow fields. In the indoor scenario where dense geometry is available, [81] proposes a method on RGB-D video propagating labels on super-pixel.

In contrast to the aforementioned methods which propagate labels in 2D, in this paper we propose to annotate both semantic and instance labels directly in 3D and then project these annotations into the 2D domain. While this approach requires a source of 3D information (e.g., SfM, stereo, laser), it is able to produce more accurate semantic and temporally consistent instance annotations for tracking purposes. Further, our experiments indicate that annotation in 3D is more time-efficient than labeling in 2D as scene elements can be separated more easily and often project into many images of the input video sequence while being only annotated once.

3D-to-2D Label Propagation: There are a few existing works on 3D-to-2D label transfer. Chen et al. [22] leverage annotations from KITTI [33] as well as 3D car models to infer separate figure-ground segmentation for all vehicles in the image. In comparison, our approach reasons jointly about all objects in the scene and also handles categories for which CAD models or 3D point measurements are unavailable (e.g., "Tree", "Sky"). Huang et al. [45] also applies 3D to 2D label transfer for generating the ApolloScape dataset. In this work, labels are transferred from 3D point clouds to images with simple splatting and projection. However, 3D points are too sparse compared with image pixels, thus, setting the splatting range is not trivial. Similarly, in [26], semantic labels annotated in the reconstructed scene are projected into each frame but not all 2D pixels are covered due to missing geometry. In addition, the two aforementioned works are limited to static scenes.

In the context of street view image segmentation, [14], [63], [65], [67], [69], [106] exploit the interaction between image pixels and 3D points to improve classification performance or efficiency. In comparison, our goal is to transfer ambiguous 3D primitive labels to every pixel in the image.

3 Annotation

In this section, we describe our data collection efforts, data preprocessing, the annotation tool, and annotation details.

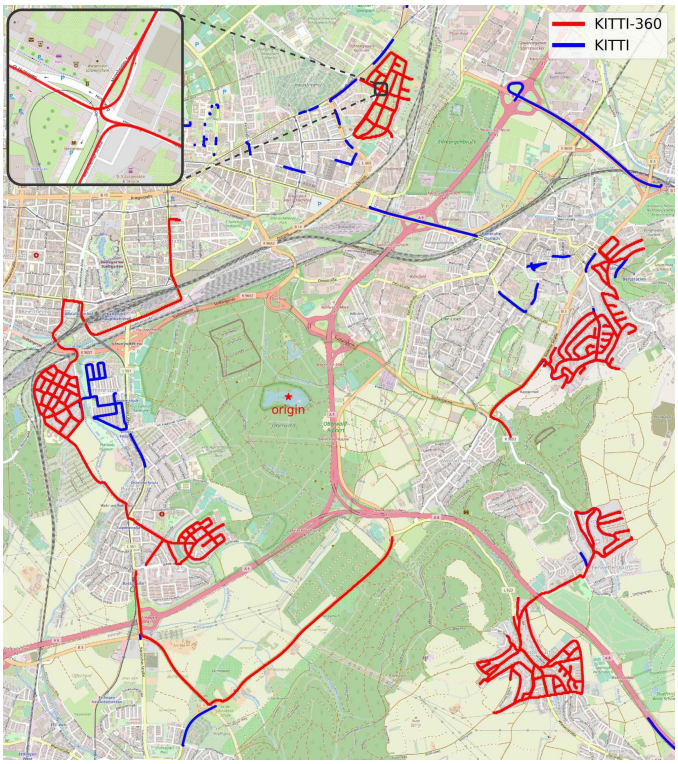


Fig. 2: **Georegistered poses** overlaid on OpenStreetMap.

3.1 Data Collection

For data collection, we equipped a station wagon with one 180° fisheye camera to each side and a 90° perspective stereo camera (baseline 60 cm) to the front. Furthermore, we mounted a Velodyne HDL-64E and a SICK LMS 200 laser scanning unit in pushbroom configuration on top of the roof. This setup is similar to the one used in KITTI [33], [34], except that we gain a wider field of view with the additional fisheye cameras and the pushbroom laser scanner while KITTI only provides perspective images and Velodyne laser scans with a 26.8° vertical field of view. Compared to omnidirectional camera systems [90], [91], our setup benefits from increased resolution of the 3D reconstruction. Localization is provided by IMU and GPS which we fuse with visual features. Fig. 1 (top left) illustrates our setup.

Using this setup, we recorded several suburbs of a mid-size city corresponding to over 300k images and 80k laser scans, covering a driving distance of 73.7km. We estimate all vehicle and camera poses using structure-from-motion [41]. More specifically, we minimize 3D reprojection errors based on all feature matches while regularizing against the GPS location. We further add loop-closures detected from LiDAR scans as regularization to complement image feature matching (which might fail on opposite-facing frames). This results in accurate georegistered camera poses. Fig. 2 illustrates the camera poses overlaid on OpenStreetMap[6]. We also plot the camera poses of the KITTI dataset [33], [34] for reference. KITTI-360 follows KITTI's forward facing camera configuration, but has minimal overlap with KITTI in terms of trajectories. This allows us to split training and test data without conflicting with the KITTI dataset, e.g., avoiding the situation where a region is used for training in KITTI but testing in KITTI-360. Following KITTI, we use Mercator projection [73] to convert geographic coordinates to a local Euclidean coordinate frame in order to facilitate usage of the dataset. The origin of the coordinate frame is chosen as the center of the map as illustrated in Fig. 2.

6. http://www.openstreetmap.org/

3.2 Annotation Interface

To facilitate 3D annotation, we developed an online annotation tool based on WebGL. We release our annotation tool (see Fig. 3) as part of this project. It consists of three main components: a scene viewer (including 2D images and 3D scene), a semantic label selection panel, and controllers. Annotators are asked to insert 3D primitives with adjustable shapes and semantic labels into the 3D scene.

3.2.1 Scene

To annotate the data while limiting transfer bandwidth, we split the collected data into batches according to the accumulated driving distances. Specifically, a single batch contains observations within a driving distance of about 200 meters (240 frames on average) and there is an overlap of 10 meters between two consecutive batches. Within one batch, we accumulate 3D points observed from the Velodyne and SICK laser scanning unit as well as the stereo camera.

During annotation, the accumulated point clouds are downsampled to reduce data loading traffic and memory. However, downsampling makes it hard to precisely perceive dynamic objects whose 3D observations are distributed along a moving trajectory. To allow for accurate labeling of dynamic objects, we apply a simple heuristic to detect dynamic objects, see Appendix A.2. We then load all detected dynamic points at each frame into the annotation tool without down-sampling. To help the annotators efficiently identifying dynamic objects, we highlight dynamic objects using white color as illustrated in Fig. 3.

As auxiliary visualization to the 3D point clouds, we provide fisheye and perspective images (see "Side View" and "Front View" in Fig. 3) in order to allow annotators to select and perceive the scene from different camera views. We also visualize the pose of each camera, enabling annotators to quickly select informative viewpoints.

3.2.2 Semantic Label Panel and Controllers

We show semantic labels with different colors in the label panel for users to choose from. To better assist annotators in placing the primitives accurately, we also offer easy-to-use controllers to interact with the 3D scene, including zoom, pan, rotation of the point cloud, switching data sources or camera views, and toggling annotations. We provide more details about the annotation interface in Appendix B.

3.3 Annotation Details

We ask the annotators to annotate the 3D point clouds in the form of bounding primitives, i.e., place cuboids and ellipsoids to enclose objects in 3D and assign a semantic label to each of them. The 3D scene is annotated with 37 label classes, including 24 "instance" classes and 13 "stuff" classes. Labels are defined in accordance with the Cityscapes

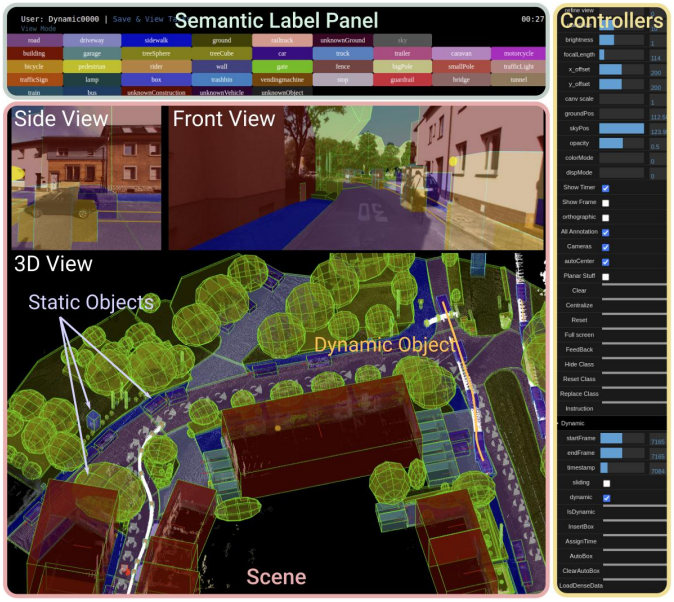


Fig. 3: **Annotation Interface.** Our interface consists of three main components: scene view (perspective views and 3D view), semantic label panel, and controllers.

dataset [25] label definition. More details about the label definition can be found in Appendix C. The annotations are categorized into static and dynamic objects, which are treated differently by our annotation tool.

3.3.1 *Static Objects Annotation*

Static labels can be further classified into two categories: "stuff" and "instance". For *instance* classes, each object is constrained to be associated with only one cuboid primitive, representing both semantic and instance labels of this object. We ask the annotators to tightly enclose the point clouds with the bounding primitives. For *stuff* classes, which usually have irregular shapes, annotators are allowed to use multiple cuboids or ellipsoids to roughly enclose the 3D points of the target objects.

We also provide a "planar" annotation option for stuff categories on the ground such as "Road" and "Sidewalk". Using this option, we allow annotators to draw a 2D polygon representing the ground object's boundary in bird's eye view. The interface then automatically estimates the height of the polygon based on the surrounding 3D geometry and extrudes the 2D polygon into 3D along the vertical direction to enclose corresponding 3D ground points. We provide more details in Appendix B.3.

3.3.2 *Dynamic Objects Annotation*

Dynamic objects mainly comprise moving vehicle and pedestrian instances. In contrast to ApolloScape [45] which annotates static objects in 3D and dynamic objects in 2D respectively, we annotate both static and dynamic objects in 3D space. However, compared to static objects, annotating dynamic objects in 3D outdoor scenes is more challenging as individual dynamic objects in the 3D reconstruction are hard to perceive and distinguish. Moreover, we need to label not only where the moving instance is, but also "when" the instance appears, requiring the annotation of moving 3D bounding boxes over time. A naïve solution is to place a 3D bounding box in every frame where the dynamic object is present. However, such an annotation process would be intractably slow. Thus, we instead implement a semi-automatic annotation scheme to reduce label time. Specifically, we minimize the effort required by annotators by making two assumptions: the size of the dynamic object is fixed over time and its trajectory is smooth. Under these assumptions, the required annotation is reduced to the size of a single 3D primitive and the pose of this primitive at several keyframes. Our annotation tool then automatically places the remaining primitives along the trajectory, see Appendix B.4 for more details.

3.4 Annotation Procedure

We annotated 379 batches in total, assigning one batch to one annotator. To control the annotation quality, we train and evaluate the annotators based on multiple pilot tasks until they have proven qualified for the full task. We also regularly verify their annotation quality and ask them for correction if necessary. We further identify a few annotators who consistently produced high-quality labels and ask them to cross-check other annotators' quality. Our annotation interface simplifies the detection and correction of annotation errors compared to annotating image sequences, which requires corrections across multiple frames. Fig. 3 shows parts of an annotated batch via our web interface.

3.5 Annotation Time

On average, annotating one full batch ($\sim$ 240 frames) in 3D required about 3 hours. Thus, our annotators spend only 3 $\times$ 60 / 240 = 0.75 minutes for "annotating" one image. In comparison, 7 minutes are required for coarse annotation of semantic instance labels in the image domain, and 1.5 hours for pixel-accurate annotations as discussed by the creators of the Cityscapes dataset [25].

4 Label Transfer Method

In this section, we first provide an overview of our method for transferring the 3D annotation to semantic instance annotations in 2D. Next, we formally introduce the model and discuss parameter learning and inference.

4.1 Overview

Given 3D annotations, we are interested in generating dense semantic instance annotations for all images and all 3D points. To incorporate inductive biases about image formation and label smoothness, we explore a Conditional Random Field (CRF) model which reasons jointly about the labels of the *3D points* and *all pixels* in the image. In practice, we apply the CRF at every timestamp independently to keep inference tractable. Despite independent inference, we are able to obtain consistent results over multiple frames thanks to the shared 3D annotations. We also experimented with inference over multiple adjacent frames but did not observe measurable improvements.

Let $\mathcal{B}_t = \{\{b^n\}, \{b_t^m\}\}$ denote all 3D annotations available at timestamp t. Here, b^n and b_t^m correspond to 3D

bounding primitives of static and dynamic objects respectively, with n and m indexing each primitive. Note that a static primitive b^n is used at all timestamps (if visible) whereas a dynamic primitive b_t^m is only included in $\mathcal{B}_t$ when it is labeled to appear at timestamp t. Fig. 4a illustrates static and dynamic bounding primitives as well as their projection into the 2D image domain. With this design, our framework allows for annotating the same object using a unique instance ID across the entire sequence as well as across 2D and 3D.

Let $\mathcal{P}_t$ denote the set of image pixels at timestamp t and $\mathcal{L}_t$ denote the visible 3D points at the same timestamp. The CRF model is defined over all elements in $\mathcal{P}_t$ and $\mathcal{L}_t$. To obtain more complete 3D information, $\mathcal{L}_t$ fuses stereo and laser scans over multiple frames. We first fuse points covering static parts of the scene, and then accumulate points of each dynamic object according to its bounding primitives and insert them into the static scene depending on the location of b_t^m. We provide more details regarding the accumulation of static and dynamic 3D points in Appendix D.1.

4.2 Model

We now formalize the CRF model applied at every frame as illustrated in Fig. 4b. Note that our 3D annotations are sparse and noisy, i.e., 3D points can carry none, one or multiple labels due to overlapping bounding primitives in 3D. The algorithm described in this section is designed to resolve these situations and infers marginal estimates for all 3D points and pixels in the image.

As the CRF model is applied at every frame independently, we drop the dependency on timestamp t of $\mathcal{B}, \mathcal{L}$ and $\mathcal{P}$ for simplicity. For each pixel $i \in \mathcal{P}$ and each 3D point $l \in \mathcal{L}$, we specify random variables s_i and s_l taking values from the set of semantic (or instance) labels $\{1, \dots, S\}$, where S denotes the number of classes. For instance inference, we assign a unique ID to each object which projects into the image. Thus, semantic and instance inference can be treated equally under our model and we will refer to both as "semantic labels" in the following. Note that there is no need to distinguish static or dynamic objects in the single frame-based CRF model. Still, we are able to retrieve whether a pixel or a 3D point belongs to a dynamic object or not according to its instance ID.

Let $\mathbf{s} = \{s_i | i \in \mathcal{P}\} \cup \{s_l | l \in \mathcal{L}\}$ denote the set of semantic labels. Dropping all dependencies on the image and point cloud for clarity we specify our CRF in terms of the following Gibbs energy function:

$$E(\mathbf{s}) = \sum_{i \in \mathcal{P}} \varphi_i^{\mathcal{P}}(s_i) + \sum_{l \in \mathcal{L}} \varphi_l^{\mathcal{L}}(s_l) + \sum_{i,j \in \mathcal{P}} \psi_{ij}^{\mathcal{P},\mathcal{P}}(s_i, s_j) + \sum_{l,k \in \mathcal{L}} \psi_{lk}^{\mathcal{L},\mathcal{L}}(s_l, s_k) + \sum_{i \in \mathcal{P}, l \in \mathcal{L}} \psi_{il}^{\mathcal{P},\mathcal{L}}(s_i, s_l) \tag{1}$$

with unary potentials $\varphi(\cdot)$ and pairwise potentials $\psi(\cdot)$. For notational clarity, we omit all conditional dependencies on the input images, 3D points and 3D annotations.

Pixel Unary Potentials: The pixel unary potentials $\varphi_i^{\mathcal{P}}(s_i)$ encode the likelihood of pixel i taking label s_i

$$\varphi_i^{\mathcal{P}}(s_i) = w_1^{\mathcal{P}}(s_i)\, \xi_i^{\mathcal{P}}(s_i) - w_2^{\mathcal{P}}(s_i) \log p_i^{\mathcal{P}}(s_i) \tag{2}$$

where $w_1^{\mathcal{P}}$ and $w_2^{\mathcal{P}}$ denote learned feature weights. Our first constraint $\xi_i^{\mathcal{P}}(s_i)$ determines the set of admissible labels and is obtained by projecting all 3D bounding primitives $\mathcal{B}$ (which are an upper bound on the objects' extent) into the image. We formulate the constraint via a binary feature $\xi_i^{\mathcal{P}}(s_i) \in \{0, 1\}$ which takes 0 for pixel i if its ray passes through a primitive of class s_i, and 1 otherwise.

In addition, we exploit a data-driven approach in order to obtain a per-pixel probability distribution over semantic labels $p_i^{\mathcal{P}}(s_i)$. Specifically, we project all non-occluded and uniquely labeled sparse 3D points into the image plane, and use these sparse projections as supervision to train a semantic segmentation network (PSPNet [116]) on the entire dataset. The output of the network's last layer is taken as the probability distribution. We also augment the training dataset using Cityscape images and labels [25] to enable the model to learn accurate object boundaries which is difficult to learn based on the projection of sparse and noisy LiDAR point clouds. As the semantic segmentation model does not distinguish instances, we further adopt a state-of-the-art instance segmentation method [108] to obtain instance hypotheses for "car", "truck", and "pedestrian". Thus, we effectively exploit the inductive biases of modern neural network architectures and co-training on related labeled datasets. As demonstrated in Appendix D.4, this leads to a significant improvement at object boundaries.

3D Point Unary Potentials: The 3D point unary potentials $\varphi_l^{\mathcal{L}}(s_l)$ encode the likelihood of 3D point l taking label s_l:

$$\varphi_l^{\mathcal{L}}(s_l) = -w^{\mathcal{L}}(s_l)\, \xi_l^{\mathcal{L}}(s_l) \tag{3}$$

where $\xi_l^{\mathcal{L}}(s_l)$ denotes a feature which takes 0 if the 3D point l lies within a 3D primitive of class s_l within $\mathcal{B}$, and 1 otherwise. As the "sky" class can't be modeled with primitives, we set $\xi_l^{\mathcal{L}}(s_l)$ to 0 if s_l takes the label "sky". Additionally, we create "virtual sky points" at infinity for all pixels whose ray doesn't intersect any 3D primitive. Note that these pixels must correspond to sky regions as we assume that the scene is densely annotated, hence each object is contained in one or several bounding 3D primitive(s).

Pixel Pairwise Potentials: Our dense pairwise term encourages semantic label coherence and connects all pixels in the image via Gaussian edge kernels following [55]

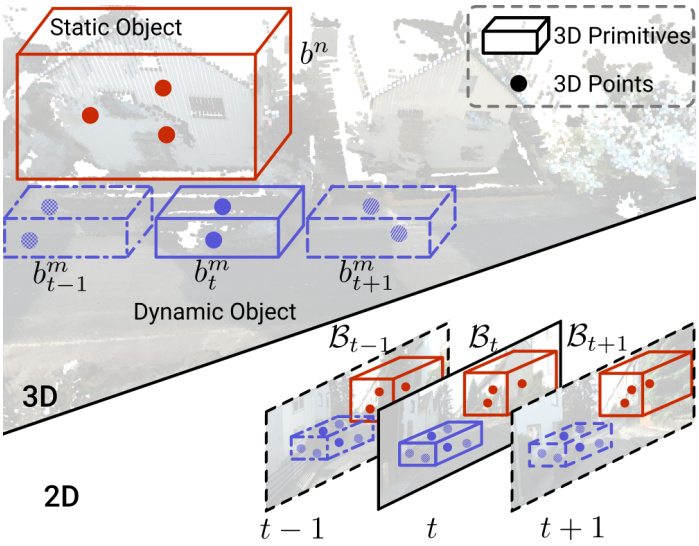
$$\psi_{ij}^{\mathcal{P},\mathcal{P}}(s_i, s_j) = w_1^{\mathcal{P},\mathcal{P}}(s_i, s_j) \exp\left\{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\,\theta_1^{\mathcal{P},\mathcal{P}}}\right\} + w_2^{\mathcal{P},\mathcal{P}}(s_i, s_j) \exp\left\{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\,\theta_2^{\mathcal{P},\mathcal{P}}} - \frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{2\,\theta_3^{\mathcal{P},\mathcal{P}}}\right\} \tag{4}$$

where $\mathbf{p}_i$ is the 2D location of pixel i and $\mathbf{c}_i$ denotes its color value. Further, $w_1^{\mathcal{P},\mathcal{P}}$ and $w_2^{\mathcal{P},\mathcal{P}}$ are learned pairwise feature weights and $\theta^{\mathcal{P},\mathcal{P}}$ parameterizes the kernel width.

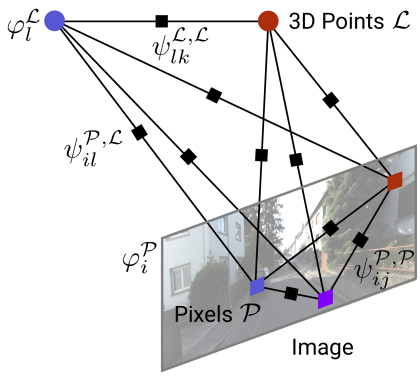
3D Pairwise Potentials: Similarly, we apply a Gaussian edge kernel to encourage label consistency between 3D points based on their 3D location and surface normals

$$\psi_{lk}^{\mathcal{L},\mathcal{L}}(s_l, s_k) = w^{\mathcal{L},\mathcal{L}}(s_l, s_k) \times \exp\left\{-\frac{\|\mathbf{p}_l^{3d} - \mathbf{p}_k^{3d}\|^2}{2\,\theta_1^{\mathcal{L},\mathcal{L}}} - \frac{(n_l - n_k)^2}{2\,\theta_2^{\mathcal{L},\mathcal{L}}}\right\} \tag{5}$$

where $\mathbf{p}_l^{3d}$ is the 3D location of point l and n_l denotes the vertical (up) component of its normal. We use the



(a) Projection of 3D Annotations at Multiple Timestamps



(b) Graphical Model at Timestamp t

(c) Inference Result at Timestamp t

Fig. 4: **3D-to-2D Label Transfer.** (a) We illustrate the 3D-to-2D projection of static and dynamic object annotations. A static 3D primitive is projected to multiple frames while a dynamic 3D object is projected only into the corresponding frame. (b) Factor graph representation of our model. Note that the CRF model is defined over all pixels and visible 3D points at a single timestamp. (c) We show the semantic inference result at timestamp t.

normal's z-component as it is the most discriminative cue for label changes between horizontal (e.g., road, sidewalk) and vertical (e.g., side of car, wall) surfaces. We estimate the respective normals using principal component analysis in a local neighborhood around each 3D point.

2D/3D Pairwise Potentials: Finally, we encourage coherence between all 3D points and the image pixels

$$\psi^{\mathcal{P},\mathcal{L}}_{il}(s_i, s_l) = w^{\mathcal{P},\mathcal{L}}(s_i, s_l) \exp\left\{-\frac{\|\mathbf{p}_i - \boldsymbol{\pi}_l\|^2}{2\,\theta^{\mathcal{P},\mathcal{L}}}\right\} \quad (6)$$

where $\boldsymbol{\pi}_l$ denotes the projection of the 3D laser or stereo point l onto the image plane. Importantly, we project only points into the image which are likely to be visible. We determine these points by meshing the 3D point cloud using the ball-pivoting method of Bernardini et al. [10], and considering only 3D points in front of the mesh. We also experimented with multi-view reconstruction approaches [48] for mesh generation, but obtained better results using this simpler approach. As applying the meshing algorithm independently for every frame is time-consuming, we generate meshes on entire batches, processing the static part and dynamic objects independently. This allows us to reuse the mesh of the static part for all frames of a batch.

4.3 Learning and Inference

This section describes inference and parameter estimation in our label transfer model.

Inference: At test time, we are interested in estimating the marginal distribution of each semantic or instance label in $\mathbf{s}$ under our model, specified by the Gibbs distribution defined in Eq. 1. A likely configuration can then be estimated by variable-wise maximization of these marginals. As our graphical model is loopy, exact inference in polynomial time is intractable. We thus resort to variational inference and approximate the probability distribution on $\mathbf{s}$ by replacing it with a factorized mean field distribution $Q(\mathbf{s}) = \prod_{i \in \mathcal{P} \cup \mathcal{L}} Q_i(s_i)$. This mean field approximation can be computed efficiently using bilateral filtering [55]. As our model comprises three sets of densely connected variables (namely $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{P} \leftrightarrow \mathcal{L}$), we exploit the algorithm of [51], [101] which generalizes [55] to multiple fields. Fig. 4c illustrates the inference result for a single frame, overlaid on the corresponding input image. Moreover, we obtain an uncertainty estimate for each pixel/3D point by computing the entropy over the respective marginal distribution. We will use this estimate in Section 6 to weigh the evaluation metrics according to the confidence of our label estimates.

Learning: We employ empirical risk minimization in order to learn the parameters in our model, considering the univariate logistic loss, defined as $\Delta(s) = -\log(P(s))$ where $P(\cdot)$ denotes the marginal distribution at the respective site. Let us subsume all model parameters into $\Theta = \{w^{\mathcal{P}}_1, w^{\mathcal{P}}_2, w^{\mathcal{L}}, w^{\mathcal{P},\mathcal{P}}_1, w^{\mathcal{P},\mathcal{P}}_2, w^{\mathcal{P},\mathcal{L}}, w^{\mathcal{L},\mathcal{L}}\}$. We define our minimization objective $f(\Theta)$ as the regularized univariate logistic loss:

$$f(\Theta) = \sum_{n=1}^{N} \sum_{i \in \mathcal{P}} -\log\left(Q_{n,i}(s^*_{n,i})\right) + \lambda\, C(\Theta) \quad (7)$$

Here, N is the number of training images, $s^*_{n,i}$ denotes the ground truth semantic label and $Q_{n,i}(\cdot)$ the approximate marginal at pixel i in image n, calculated via mean field approximation. $C(\Theta)$ is a quadratic regularizer on the parameter vector Θ. We whiten all features and use a single value λ which we select via cross-validation on the training set. For learning the instance segmentation parameters we exploit the same loss $f(\Theta)$ as for semantic segmentation, but assign unique labels to each individual object, e.g., different cars will be assigned different labels even if they occlude each other. In order to associate 2D ground truth instances with 3D instances we project all visible 3D points into the

image and find a consensus via the majority vote which gave good results in practice. As the number of instances per semantic class varies between images, we learn intra- and inter-class pairwise potentials using parameter tying. We optimize the objective function $f(\Theta)$ using stochastic gradient descent and obtain $\partial Q / \partial \Theta$ using auto differentiation. We make use of the ADADELTA algorithm [113] with decay parameter 0.95 and $\epsilon = 10^{-8}$, and randomly sample a batch of 16 training images at each iteration for which all gradients can be computed in parallel.

5 Label Transfer Evaluation

In this section, we first introduce the datasets we use for training and evaluating our label transfer method. Next, we evaluate our method in ablation studies and compare it against several label transfer baselines. Finally, we also show qualitative results of our method.

5.1 Training and Evaluation Data

We manually annotate a set of images with pixel-wise ground truth to train and evaluate our label transfer method. The *training* set contains 125 images selected from diverse scenarios such that a substantial amount of pixels are labeled within each class. These training images are different from those used in our conference version [107]. We create this new training set following the label definition of CityScapes [25] as [107] considers fewer classes. To enable comparison to 2D label transfer methods which require images with large overlapping regions, we additionally annotate 240 adjacent frames from 13 different suburbs in equidistant steps of 5 frames in the 2D image domain for *evaluation*. The evaluation set has no spatial overlapping with the training set, allowing us to assess the generalization ability of our method. We evaluate our label transfer method on static and dynamic objects separately. Following [107], the performance of static objects is evaluated on 120 densely labeled frames from 5 suburbs containing the most frequently occurring 14 classes. The remaining 120 frames are sampled from 8 different suburbs which contain dynamic objects. For these frames we label the dynamic objects while leaving the static region unannotated. We consider 7 common dynamic objects, see Appendix E.1 for details.

5.2 Quantitative Evaluation

This section presents our quantitative evaluation on semantic and instance segmentation. We compare our method with several label transfer baselines and conduct ablation studies.

5.2.1 Semantic Segmentation Transfer

For evaluating semantic segmentation transfer performance, we measure overall performance by the mean intersection over union (mIoU) and the average pixel accuracy (Acc). While [107] evaluates the *weighted* mean IoU which is biased by object occurrences, we follow Cityscapes [25] and measure the mean IoU without weighting. For all experiments, we provide results for individual classes in Appendix E.1.

Baselines: We compare our method to several 2D to 2D label transfer methods on both static and dynamic objects in Table 2. Here, the task is to predict the center frame from two annotated images (± 5 frames corresponding to 0.5 seconds of driving or ~ 5 meters travel distance). Our first baseline ("Label Prop.") is the label transfer approach presented in [100]. To ensure that all baselines have access to the same information, we do not select frames actively but use equidistantly spaced frames for all methods. We construct a second baseline ("Sparse Track. + GC") using the feature tracking approach of [95] to propagate semantic labels from the two closest labeled frames to the target frame. To densify the label map, we apply graph cuts (GC) with contrast sensitive edge potentials [11]. In order to evaluate the value of 3D information, we implemented a third baseline ("3D Prop. + GC") which works similar to the previous one, but replaces the sparse tracking part with correspondences obtained by transferring pixels of the two closest labeled frames to the target image via the visible vertices of our 3D mesh followed by graph cuts propagation.

Method	Label source	Static		Dynamic	
		mIoU	Acc	mIoU	Acc
Label Prop. [100]	2D	49.0	81.0	37.2	59.1
Sparse Track. + GC [95]	2D	51.2	79.1	8.2	12.5
3D Prop. + GC	2D	72.1	87.4	14.5	21.7
Fully Conn. CRF [55]	2D	63.6	88.7	–	–
PSPNet [116]	CS + 3D	67.2	90.4	–	–
3D Primitives + GC	3D	49.4	73.4	–	–
3D Mesh + GC	3D	66.8	85.7	–	–
3D Points + GC	3D	72.6	87.8	–	–
Proposed Method	3D	**81.2**	**93.1**	**63.5**	**94.1**

TABLE 2: **Comparison to Label Transfer Baselines on Semantic Segmentation Transfer**. We compare our method to 2D label transfer baselines (top) and to 3D to 2D label transfer baselines (bottom). Label source: "2D": Labeled neighboring image frames, "CS": Cityscapes training images, "3D": 3D bounding primitives.

While all aforementioned baselines require labeled adjacent frames as input at inference time, we consider two more methods that generalize to arbitrary frames. First, we train the segmentation model of Krähenbühl et al. [55] ("Fully Conn. CRF") which was also used in [107] and which uses a similar inference algorithm as our label transfer method on all annotated adjacent frames of the test sequence. Finally, we evaluate the deep semantic segmentation network [116] ("PSPNet") that also provides dense unary information for our method. As discussed in Section 4.2, this model is trained on non-occluded sparse 3D projections combined with the CityScapes training set [25]. Note that neither PSPNet nor our method has access to adjacent annotated frames for training or inference.

We further consider several 3D to 2D label transfer baselines that exploit our 3D annotations without requiring equidistantly labeled 2D annotations. Specifically, we project 3D primitives, meshes or visible 3D points into the 2D image domain, followed by graph cut inference ("3D Primitives + GC"; "3D Mesh + GC"; "3D Points + GC").

Static Objects: Table 3 (left) shows the comparison on

Method	Semantic		Instance	
	mIoU	Acc	mIoU	Acc
LA	67.6	88.7	72.3	86.7
LA+3D	70.4	89.7	72.9	88.7
LA+PW	66.6	87.9	72.9	85.8
LA+PW+CO	76.8	91.8	81.6	91.0
LA+PW+CO+3D	78.2	92.4	83.6	91.7
Full Model	**81.2**	**93.1**	**83.7**	**91.8**
Full Model (90%)	88.3	96.0	89.0	94.9
Full Model (80%)	92.5	97.6	91.3	96.6
Full Model (70%)	**94.3**	**98.4**	**92.7**	**97.4**

TABLE 3: **Semantic Instance Segmentation Transfer Ablation** evaluated on static objects. The components are abbreviated as follows: LA = local appearance ($p^{\mathcal{P}}$), PW = 2D pairwise constraints ($\psi^{\mathcal{P},\mathcal{P}}$), CO = 3D primitive constraints ($\xi^{\mathcal{P}}$), 3D = 3D points ($\varphi^{\mathcal{L}}$,$\psi^{\mathcal{P},\mathcal{L}}$), Full Model = all potentials including 3D pairwise constraints ($\psi^{\mathcal{L},\mathcal{L}}$). Percentages denote fractions of estimated pixels.

120 consecutive images of static objects.[7] From the 2D label transfer baselines shown at the top, the mesh transfer method which uses projected 3D information performs best in terms of mIoU. Furthermore, and maybe surprisingly, the sequence-specific fully connected CRF model performs on par or even better than special purpose label transfer methods. This is caused by the fact that optical flow (as used in [95], [100]) often fails for street scenes like ours due to large displacements, perspective distortions, textureless regions and challenging lighting conditions. Interestingly, PSPNet achieves the best accuracy while performing worse on mIoU. Despite obtaining superior results on large objects (e.g., "Building"), it struggles with less-occurring classes such as "Trailer" and "Gate".

The bottom half of Table 2 (left) compares the proposed method with respect to the 3D to 2D label transfer baselines. As evidenced by our results, simply projecting 3D primitives or meshes into the image and smoothing via GC does not perform well due to the crude approximation of the geometry. Better results are obtained when projecting the visible 3D points followed by spatial propagation. Finally, we observe that all baselines are outperformed by the proposed method (last row). Note that we also map the 37 semantic labels of our 3D annotations to the most common 14 categories considered in the static evaluation images (see Appendix C) for all 3D to 2D label transfer methods.

Dynamic Objects: We evaluate our method on dynamic objects against 2D label transfer baselines. Here, we consider all static regions as a single background class during evaluation. Note that we neglect "Fully Conn. CRF" and "PSPNet" as both methods address semantic segmentation and thus cannot distinguish static and dynamic objects within the same class. Table 2 (right) shows that our method also outperforms all 2D label transfer baselines on dynamic objects. While the mIoU is calculated over a different set of classes, the average performance of our method on dynamic objects is slightly degraded compared to our result on static objects. Labeling of dynamic objects is more challenging in our annotation pipeline for two reasons: Since we accumulate 3D points of dynamic points according to the annotated bounding primitives over multiple frames, slight misalignments of the primitives may lead to inaccurate accumulation and thus erroneous 3D cues. Furthermore, the accumulation of deformable objects ("Rider", "Person") leads to noisy 3D point clouds. Despite these challenges, our method achieves satisfying performance on all dynamic objects.

Annotation Time Comparison: While all 2D methods require every 10th frame to be labeled, our method (as well as the other 3D baselines) requires 3D annotations in the form of 3D primitives. Assuming 60 minutes annotation time per image, this amounts to 20 hours of annotation time per batch of 200 frames when labeling one 2D image every 10th frame, while the respective 3D annotations for this scene can be obtained in about 3 hours. This gain multiplies with the frame rate and the number of cameras (our setup has four).

Ablation Study: We validate the importance of the individual components of our model on semantic segmentation in Table 3 (upper left), evaluated on the densely labeled images of static objects. Starting with the appearance classifier $p^{\mathcal{P}}$ trained on the projected sparse 3D points ("LA"), we incrementally add the terms $\varphi^{\mathcal{L}}$,$\psi^{\mathcal{P},\mathcal{L}}$ related to the 3D points ("3D"), the semantic pairwise term $\psi^{\mathcal{P},\mathcal{P}}$ between pixels ("PW"), the 3D primitive constraints $\xi^{\mathcal{P}}$ ("CO") and finally the 3D pairwise constraints $\psi^{\mathcal{L},\mathcal{L}}$ as specified in Eq. 1. We note that each component is able to increase performance. We obtain the largest improvement by reasoning about the relationship between points in 3D and pixels in the image.

Label Uncertainty: Here, we leverage our model's awareness of label uncertainty to demonstrate that higher accuracy can be achieved in confident regions. To quantify uncertainty, we measure the entropy of the label marginal distribution at every pixel, see Fig. 5 (last row). Sorting all pixels according to their entropy allows us to predict the most certain regions in the image. Table 3 (bottom) shows our results on static objects when predicting only those parts of the image. Note how this helps to boost our performance to 94.3% mIoU and 98.4% accuracy when predicting at 70% pixel density, demonstrating that our uncertainty estimates are well calibrated. In contrast, uncertainty is not directly accessible in most baseline models as they are deterministic or rely on MAP estimates. In the benchmarks introduced in Section 6 where our inferred labels are considered as pseudo-ground truth, we adopt confidence weighted evaluation metrics leveraging the uncertainty to take into account the ambiguity in our automatically generated annotations.

5.2.2 Instance Segmentation Transfer

As time consistent 2D instance ground truth is hard to obtain, most existing 2D label transfer methods focus on the semantic segmentation problem. Therefore, we chose to evaluate instance segmentation performance in an ablation study. We annotated the classes "Building", "Car", "Trailer", "Caravan" and "Box" with instances in our 2D

7. The results differ slightly from those presented in [107] as 1) we updated the ground truth labels to be consistent with the extended label definition and 2) we measure mIoU following Cityscapes [25] while [107] reports weighted mIoU.

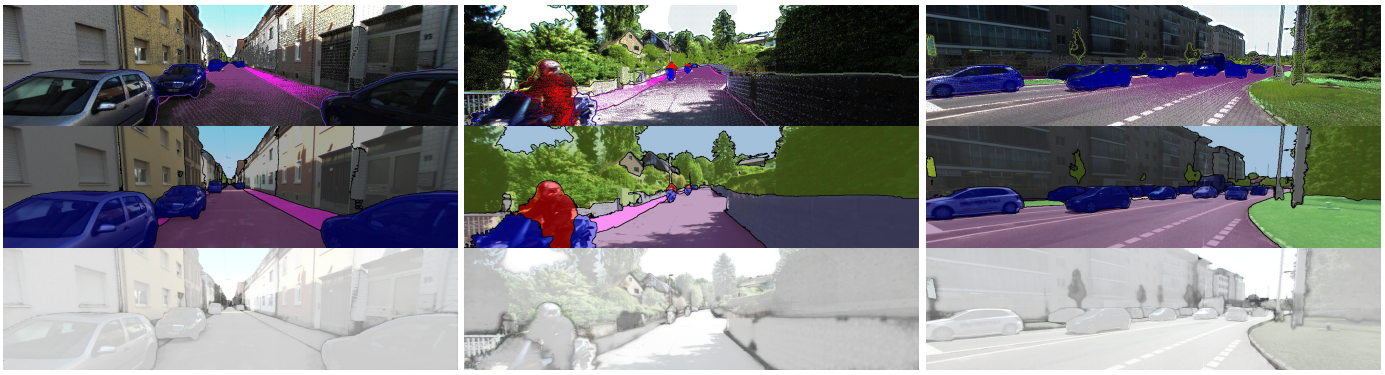


Fig. 5: **Qualitative Results on Semantic Instance Segmentation Transfer.** Each subfigure shows from top-to-bottom: the input image with the projected 3D points and inferred semantic segmentation boundaries, the inferred semantic instance segmentation, as well as the confidence map of the inferred label with bright and dark colors indicating high and low confidence, respectively. See supplementary material and text for details. The first scene (1st column) contains only static objects while the others (2nd and 3rd columns) also contain dynamic objects.

ground truth[8]. For evaluation, we exploit the mIoU metric defined on instances following [107]. Specifically, we first match the ground truth instances to the predicted instances. A pixel is then classified as true positive only when its predicted instance index matches the ground truth. Table 3 (right) shows our results. Note how the instance segmentation results are on par with the semantic segmentation, demonstrating our model's intra-class separation ability. Moreover, we also observe higher instance segmentation accuracy when filtering uncertain predictions.

5.3 Qualitative Evaluation

Fig. 5 illustrates our dense inference results qualitatively for 3 different scenes in terms of semantic instance segmentation on both static and dynamic objects. The first two rows illustrate both semantic and instance labels, where semantic information is color-coded and instances are separated by boundaries. The last row shows the confidence maps. While the proposed method is able to delineate most object boundaries satisfyingly, some challenges remain. Errors occur in regions where 3D points are absent due to far distance (1st & 3rd scene: far building). Another source of errors is inherent label ambiguities that occur for porous objects such as fences or trees (3rd scene: tree boundary) where even 2D ground truth annotation is a hard and ambiguous task. Finally, 3D points of dynamic objects are accumulated over multiple frames (2nd & 3rd scene), providing dense but less accurate 3D cues to the CRF model. However, note that our probabilistic inference algorithm is able to successfully identify those uncertain regions as demonstrated in the last row, where far buildings and object boundaries are predicted as less certain compared to other image regions.

6 Dataset & Benchmarks

We apply the proposed label transfer method to all frames captured by perspective cameras, resulting in 2×78k 2D semantic/instance segmentation maps, 1.0B 3D semantic points and 172.4M 3D instance points. We provide a statistical analysis of the 2D & 3D labels in Appendix F.1. We further deploy an online evaluation server and establish benchmarks on a set of challenging tasks relevant to autonomous driving. For all tasks, we split the data at the batch-level into disjoint training, validation and held-out test sets as specified in Appendix F.2. Specifically, we leverage KITTI-360 to address tasks at the intersection of vision, graphics and robotics which are commonly viewed as relevant towards achieving full autonomy, including tasks within the scope of semantic scene understanding, novel view synthesis and semantic SLAM. We now describe each task and the corresponding evaluation protocol in detail. Furthermore, we introduce initial baselines for each task.

6.1 Semantic Scene Understanding

In this section, we establish scene perception benchmarks in both 2D image space and 3D domain. We first implement benchmarks for the traditional tasks of 2D semantic segmentation and 2D instance segmentation on perspective images, using the inferred semantic/instance segmentation maps as pseudo ground-truth. While not the main focus of this work, we establish these standard 2D benchmarks to investigate whether there is a performance gap between methods operating in 2D and 3D. Furthermore, as our label definition is compatible with Cityscapes, this benchmark opens up the possibility for studying domain adaption across datasets in future work. Next, we establish benchmarks in the 3D domain, including bounding box detection and semantic/instance segmentation. Moreover, we consider a semantic scene completion task where the goal is to simultaneously complete the scene and infer corresponding semantic labels given limited observations. This task allows autonomous vehicles to hallucinate future possibilities and thus can benefit downstream tasks, e.g., predictive control.

2D Semantic Segmentation: We train and evaluate 2D segmentation baselines on the densely labeled images in KITTI-360. We consider two well-known methods, Fully Convolutional Neural Network (FCN) [61] and Pyramid

8. While [107] uses two sets of parameters for semantic and instance segmentation, we train a single model for instance segmentation and read semantic labels directly from the instance maps. Therefore, our predictions on classes without instance labels are the same in both semantic and instance segmentation maps.

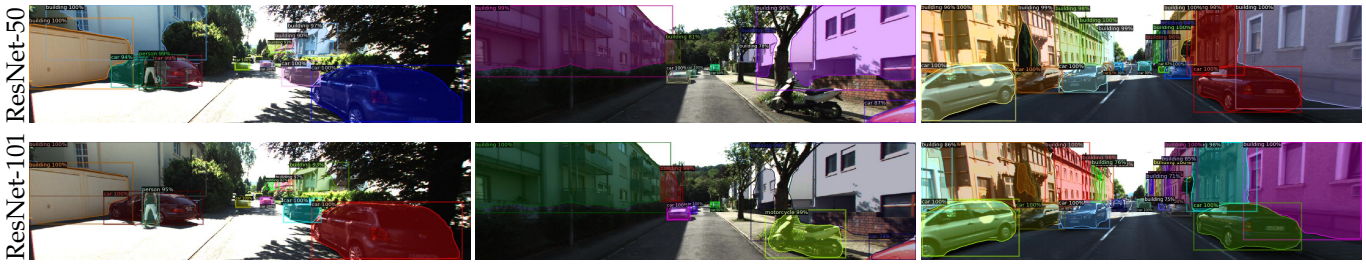


Fig. 6: **Qualitative Results for 2D Instance Segmentation.** The first row shows inference results of Mask-RCNN with a ResNet-50 backbone while the second row uses a ResNet-101 backbone. The ResNet-101 backbone leads to better results, e.g., with the ResNet-50 backbone, the car occluded by the person is split into two instances (left) and the motorcycle is not detected (middle). Note that both variants are able to predict "Building" instances after being trained on KITTI-360.

Method	$\text{mIoU}_{\text{class}}$	$\text{mIoU}_{\text{category}}$
FCN [61]	54.0	77.6
PSPNet [116]	**64.9**	**82.2**

(a) 2D Semantic Segmentation

Method	Backbone	AP	AP_{50}
Mask R-CNN [40]	Res. 50	19.5	36.3
	Res. 101	**20.9**	**40.1**

(b) 2D Instance Segmentation

Method	AP_{50}	AP_{25}
BoxNet [76]	**4.1**	23.6
VoteNet [76]	3.4	**30.6**

(c) 3D Bounding Box Detection

Method	$\text{mIoU}_{\text{class}}$	$\text{mIoU}_{\text{category}}$
PointNet [77]	13.1	30.4
PointNet++ [78]	**35.7**	**58.3**

(d) 3D Semantic Segmentation

Method	AP	AP_{50}
PointNet++ [78]+ [30]	23.7	40.1
PointGroup [49]	**34.8**	**53.6**

(e) 3D Instance Segmentation

Method	Acc / Cmp / F_1	$\text{mIoU}_{\text{class}}$
Raw Input	**98.2** / 19.1 / 32.4	–
Enc-Dec	41.4 / **41.2** / **41.3**	9.1

(f) Semantic Scene Completion

TABLE 4: **Quantitative Results for 2D & 3D Scene Understanding on Various Different Tasks.**

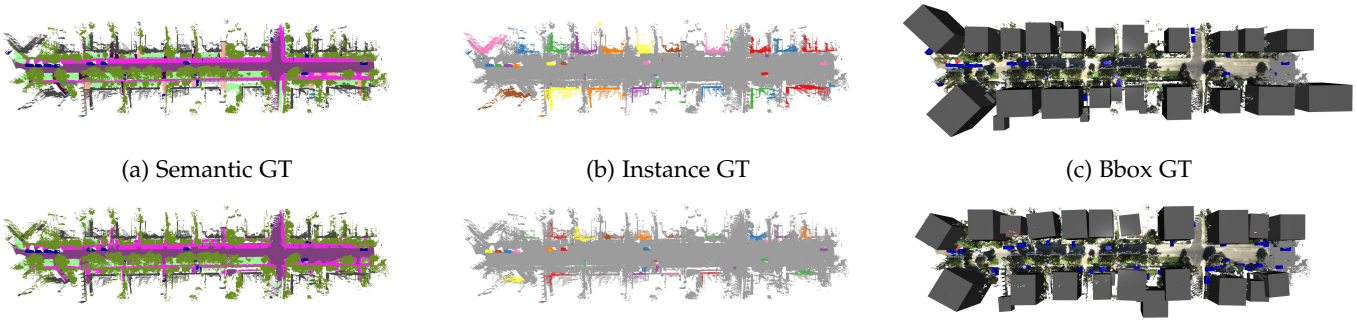
Scene Parsing Network (PSPNet) [116], as a reference. Following Cityscapes [25], we adopt mean intersection over union (mIoU) at two semantic granularities, i.e., classes and categories, where 19 classes are grouped into 7 coarse-grained categories. To account for label uncertainty, the mIoU is weighted by the confidence of our pseudo-ground truth labels. A formal definition of our metrics and a detailed definition of the classes and categories can be found in Appendix G.1. Table 4a shows that, unsurprisingly, PSPNet outperforms the naïve FCN on the test set.

2D Instance Segmentation: We use the established Mask R-CNN framework [40] with different backbones as our baselines, see Table 4b. We measure the Average Precision (AP) weighted by the label confidence over 10 thresholds, ranging from 0.5 to 0.95 with a step size of 0.05. The mean AP is then calculated over 7 classes that contain instance labels. We also compare mean AP_{50} given a threshold of 0.5. Both Table 4b and Fig. 6 suggest that Mask R-CNN with a deeper backbone leads to better performance. Note that we provide instance segmentation labels of "Buildings" which are not available for other outdoor datasets [18], [25], [36], [45]. This information allows future works to explore scene compositionality [57], [72], [74] in real-world street scenes.

3D Bounding Box Detection: In this benchmark we measure the mean AP over two classes, "Building" and "Car", since it is particularly challenging for learning-based algorithms to generalize well to other classes with fewer training samples. Following [76], the mean AP is calculated at a threshold of 0.25 and 0.5, respectively. We consider VoteNet [76] and its simplified version BoxNet [76] as baseline methods. Both methods require 3D point locations as input and output 3D bounding boxes and their semantic labels. Table 4c suggests that VoteNet can make reasonable predictions for both building and cars while it fails to predict 3D bounding boxes with high IoU values, see Fig. 7f.

3D Semantic Segmentation: We establish a 3D semantic segmentation benchmark on the accumulated point clouds, where PointNet [77] and PointNet++ [78] are trained and evaluated as baselines. Both methods take as input point locations and colors to predict a semantic label for each 3D point. Following the 2D semantic segmentation task, we measure mIoU weighted by label confidence over classes and categories, respectively. Table 4d shows the quantitative comparison and Fig. 7d illustrates the performance of PointNet++. Interestingly, comparing Table 4a and Table 4d shows that the 3D semantic segmentation baselines' overall performances are inferior compared to the 2D semantic segmentation methods, suggesting that parsing the semantic meaning of irregularly structured 3D point clouds remains more challenging and requires further work.

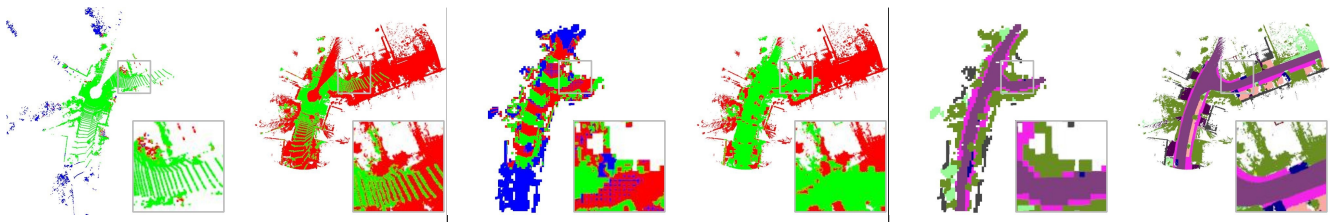
3D Instance Segmentation: We evaluate 3D instance segmentation results for "Building" and "Car". Specifically, we measure the mean AP over a set of thresholds ranging from 0.5 to 0.95 with a step size of 0.05 and AP at a threshold of 0.25 and 0.5. As a first simple baseline, we naïvely cluster semantically labeled points into instances. We use PointNet++ [78] for semantic segmentation and DBSCAN [30] for clustering. We further evaluate PointGroup [49] as a state-of-the-art method for 3D instance segmentation which takes as input point locations and colors. Table 4e demonstrates that PointGroup outperforms the naïve clustering-based method. The qualitative result of PointGroup is shown in Fig. 7e. While the 2D and 3D results in Table 4b and Table 4e are defined over different sets of classes, we provide detailed results on each class in Appendix G.5. Interestingly, 3D instance segmentation methods achieve better performance on "Car" than 2D methods while performing worse on "Building". We hypothesize that unlike in 2D where occlusions strongly impact the results (e.g. Fig. 6 left, pedestrian standing in front of a car), cars can be more



(a) Semantic GT (b) Instance GT (c) Bbox GT

(d) PointNet++ [78] (e) PointGroup [49] (f) VoteNet [76]

Fig. 7: **Qualitative Results for 3D Scene Perception.** We establish benchmarks for 3D semantic/instance segmentation and 3D bounding box detection. This figure shows the ground truth and the prediction of a baseline method for each sub-task.



Raw Input Acc. Raw Input Comp. Enc-Dec Acc. Enc-Dec Comp. Enc-Dec Semantic GT Semantic

Fig. 8: **Qualitative Results for Semantic Scene Completion** evaluated at a distance threshold of 20cm. Green denotes complete/accurate, red denotes incomplete/inaccurate and blue denotes points in unobserved region.

easily separated in 3D. As for buildings, many instances are spatially connected (e.g., Fig. 6 right), making the instance segmentation task harder in 3D where boundaries are harder to detect on the sparse point cloud.

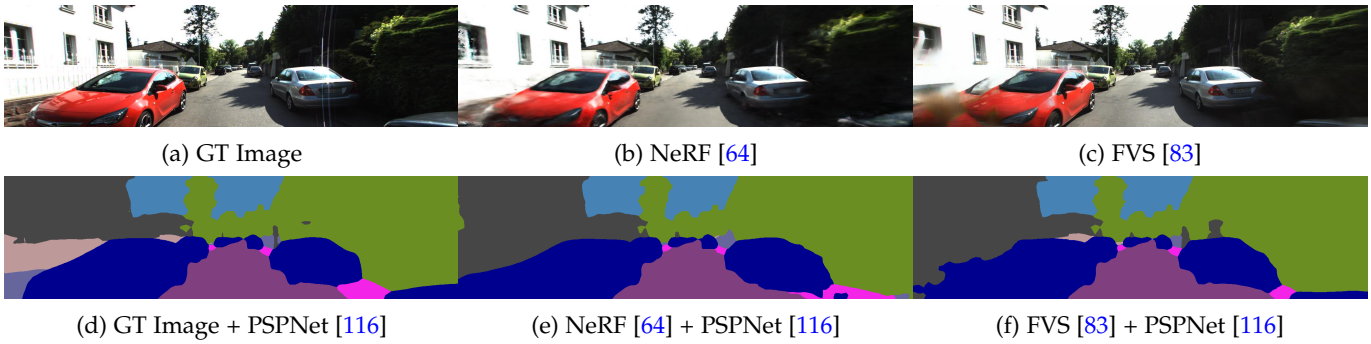
Semantic Scene Completion: While standard scene perception tasks aim to predict a semantic label for each observed scene point, the semantic scene completion task additionally requires predicting geometry and semantics in unobserved regions. Given a single LiDAR scan as input, this task requires semantic scene completion within a corridor of 30m around the vehicle poses of a 100m trajectory. For evaluation, we measure reconstruction quality and semantic prediction accuracy. The former measures geometric accuracy independent of semantics, using *completeness* and *accuracy* over a range of distance thresholds following common practice [89]. We consider a threshold of 20cm as the main metric. As our ground truth reconstruction may not be complete, we evaluate accuracy only in observed regions. We further measure the F_1 score as the harmonic mean of the completeness and the accuracy. The semantic prediction quality is conditioned on the geometric reconstruction. Specifically, we measure the confidence weighted *mIoU* over the same set of thresholds where a true positive prediction is made when 1) a ground truth point is classified as complete at the given threshold and 2) its closest reconstructed point has the correct label. See Appendix G.6 for more details of the ground truth construction and evaluation metrics.

We consider two baselines for this task, both taking a single raw LiDAR frame as input. For calibration, we implement a naïve baseline which returns the input as output. The second baseline is a learning-based approach where we use an encoder-decoder architecture to predict the complete scene structure from the raw LiDAR scan. More details about this baseline can be found in Appendix G.6. Table 4f and Fig. 8 illustrate the results. As expected, the raw LiDAR scans are accurate but incomplete. The learning-based approach instead achieves higher completeness but the predictions are less accurate. For the learning-based approach we also predict a semantic label at each 3D point. Fig. 8 shows that the model is able to correctly predict semantic labels at a coarse level but struggles to predict smaller objects like cars.

Discussion: Our results show that 3D semantic segmentation is harder than 2D semantic segmentation. In contrast, our conclusions for instance segmentation vary for different classes. Some classes, e.g., cars, are easier to segment in 3D, suggesting further works can explore 3D information to enhance 2D instance segmentation. 3D bounding box detection remains challenging, especially when a high IoU is desired. Lastly, while inferring dense geometry and semantics from raw sparse observations can benefit autonomous driving, completing the scene and predicting semantics jointly is a difficult task that requires further research.

6.2 Novel View Synthesis

Simulation is an essential tool for training and evaluating autonomous vehicles. While existing methods trained in simulated scenes struggle to generalize to real scenes, creating a simulation environment based on real-world images is a promising direction to close the gap between real-world scenarios and synthetic environments [24], [110]. We thus establish challenging benchmarks towards this goal,



(a) GT Image (b) NeRF [64] (c) FVS [83]

(d) GT Image + PSPNet [116] (e) NeRF [64] + PSPNet [116] (f) FVS [83] + PSPNet [116]

Fig. 9: **Qualitative Results for Novel View Appearance & Semantic Synthesis.** The first row shows the GT image and novel view appearance synthesis results. The second row shows the corresponding semantic segmentation using PSPNet [116].

including novel view appearance synthesis and novel view semantic synthesis.

Novel View Appearance Synthesis: In this benchmark, we are interested in novel view RGB image synthesis for driving scenarios. While we evaluate on a set of held-out perspective images, the benchmark participant can choose from a set of input modalities[9], including posed perspective/fisheye images or accumulated point clouds. For perspective and fisheye images, we release approximately 50% of the frames for training and use the remaining 50% for testing. In addition, the evaluation server also provides a harder setting with a 90% drop rate. See appendix for details. The point cloud is accumulated over all frames where each point fuses colors from different viewpoints. We adopt three standard evaluation metrics for this benchmark: peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and perceptual metric (LPIPS) [115].

We evaluate two sets of baselines on two different input modalities. We first consider a naïve baseline using the accumulated point cloud (PCL) as input. Specifically, we project non-occluded colored points to test viewpoints, followed by nearest neighbor interpolation to fill in the missing values. As there are no 3D points in the sky, we heuristically assign a mean blue color to the sky region. We further consider several state-of-the-art baselines for image-based novel view synthesis, including methods based on Neural Radiance Fields [7], [27], [64] or per-view depth maps [54], [83]. Results in Table 5 (left) and Fig. 9 (1st row) reveal the challenges of this benchmark. We observe that NeRF [64] shows promising results but struggles to synthesize fine structure. FVS performs better in rendering fine details (e.g., license plate) but exhibits noticeable artifacts due to the inaccurate underlying geometry (e.g., left car). Interestingly, FVS/PBNR performs better in LPIPS but has a lower PSNR compared to NeRF-based methods, suggesting that LPIPS is more sensitive to fine detail than larger regional errors.

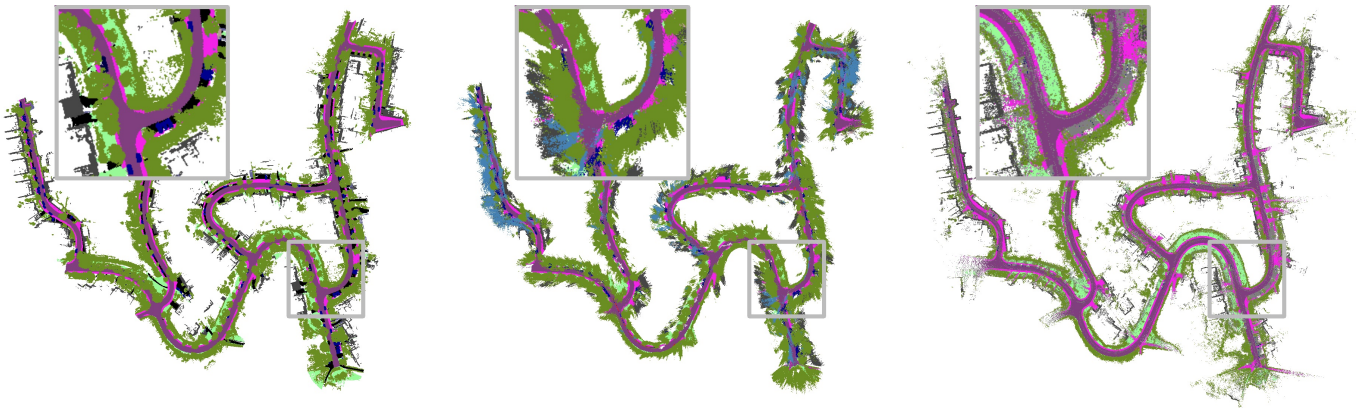
Novel View Semantic Synthesis: An important property of simulation environments like CARLA [29] is that they provide not only RGB images but also auxiliary information like semantic label maps. Towards a real-world simulator with the same capability, we therefore consider a novel benchmark that requires joint novel view and semantic synthesis. The input data for this task is the same as for the novel view synthesis task, while the methods are tasked to predict both an RGB image and a semantic segmentation map at a given target camera pose. Therefore, the evaluation metric of this task additionally comprises mIoU for semantic segmentation as shown in Table 5 (right). As no prior work has addressed this problem yet, we consider a naïve two-stage solution as baseline to bootstrap this benchmark, i.e., we apply an existing semantic segmentation model (PSPNet [116]) on the synthesized images. For comparison, we also evaluate the semantic segmentation performance on the original ground truth images (GT Image). Note that the artifacts in the synthesized images lead to a significant performance drop for semantic segmentation. As illustrated in Fig. 9, the fence is misclassified as building when the synthesized images are taken as input to PSPNet, despite that the fence is still visible in these images. It is also interesting to note that semantic segmentation performance is aligned with the LPIPS metric, as both apply pre-trained networks on synthesized images.

Method	Input	PSNR	SSIM	LPIPS	$\text{mIoU}_{\text{class}}$	$\text{mIoU}_{\text{category}}$
GT Image	–	–	–	–	**72.0**	**83.6**
PCL	PC	12.81	0.576	0.549	39.4	46.8
NeRF [64]	PI	21.18	0.779	0.343	53.0	73.9
mip-NeRF [7]	PI	**21.54**	0.778	0.365	51.2	72.0
DS-NeRF [27]	PI	21.28	0.777	0.347	54.8	75.5
FVS [83]	PI	20.00	0.790	**0.193**	67.1	78.5
PBNR [54]	PI	19.91	**0.811**	**0.191**	65.1	77.8

TABLE 5: **Quantitative Results for Novel View Appearance & Novel View Semantic Synthesis using a 50% drop rate.** Input: "PC" denotes the accumulated point cloud and "PI" means perspective images.

Discussion: Our baselines reveal the different challenges in novel view appearance synthesis with different input modalities. While point clouds provide a good representation of 3D geometry, it is not easy to model view dependency. When instead taking a sparse set of multi-view images as input, the task is similarly difficult despite little variation in the camera orientation. We believe that future works should explore the combination of different input modalities to improve image fidelity further. Moreover, given the low performance of our simple baselines on the novel view semantic synthesis task, there is a large potential for future improvements, i.e., by learning view and semantic synthesis jointly.

9. The used input modalities will be indicated on the leaderboard.



(a) GT Semantic Reconstruction (b) ORB-SLAM2 + SGM (c) SUMA++

Fig. 10: **Qualitative Results for Semantic Mapping** on test sequence 3 colored based on semantic class labels.

6.3 Semantic SLAM

We further establish a semantic SLAM benchmark at the intersection of robotics and computer vision. Here, the goal is to simultaneously estimate poses and reconstruct a *semantic* map from monocular/stereo images and/or LiDAR scans. While there is a growing interest in evaluating indoor semantic reconstructions of SLAM algorithms at room-level [87], [102], existing works on outdoor semantic SLAM typically evaluate only pose estimation while ignoring the quality of the semantic reconstruction [12], [23]. Considering that the semantic reconstruction is valuable on its own for down-stream tasks, e.g. planning [28], we thus additionally evaluate geometric and semantic reconstructions where the latter is enabled by the dense semantic annotations of KITTI-360. For this benchmark, the test sequences are separated from those used for 3D scene perception such that the accumulated point cloud is held out from the public.

Localization: Given an estimated trajectory, we adopt the standard Absolute Pose Error (APE) and Relative Pose Error (RPE) [37] as metrics for evaluating pose estimation. We consider four test sequences for this task and report the evaluation results on each test sequence without averaging.

We evaluate two baseline methods, ORB-SLAM2 [66] and SUMA++ [23], where the former takes stereo images as input and the latter is applied on LiDAR scans. Table 6a compares the localization results of ORB-SLAM2 and SUMA++. For both methods, the APE exceeds 2 meters and the RPE is around 2% in general. ORB-SLAM2 achieves better overall performance compared to SUMA++, suggesting that the stereo images of our dataset contain rich features for the purpose of localization. One possibility for improving localization accuracy is to exploit the 3D bounding boxes and instance labels available in our dataset [71], [109].

Geometric and Semantic Mapping: We measure the quality of the geometric and semantic mapping using the same metrics considered in the semantic scene completion benchmark. As richer input observations are available in this task, we adopt a smaller distance threshold of 10cm as the main metric. Specifically, we first measure geometry accuracy using completeness and accuracy, and then evaluate semantics on the completed ground truth points via the confidence weighted mIoU. As the mapping accuracy is highly correlated with the APE, we compare ground truth and estimated reconstruction in local windows to minimize the impact of pose drifts. Each local window consists of 50 consecutive frames and is aligned to the ground truth based on the trajectory, see Appendix I.2 for more details.

Test Seq.	ORB-SLAM2 APE (m)	ORB-SLAM2 RPE (%)	SUMA++ APE (m)	SUMA++ RPE (%)
0	**1.53** $\pm$ 0.74	**2.42** $\pm$ 1.34	2.27 $\pm$ 1.38	2.66 $\pm$ 2.12
1	**2.22** $\pm$ 0.78	2.46 $\pm$ 1.37	2.87 $\pm$ 1.50	**2.43** $\pm$ 1.80
2	**2.12** $\pm$ 0.94	**1.50** $\pm$ 1.01	4.62 $\pm$ 4.24	2.90 $\pm$ 2.90
3	**1.79** $\pm$ 0.96	**1.72** $\pm$ 1.22	2.77 $\pm$ 1.44	2.88 $\pm$ 2.42

(a) Localization. RPE evaluated with a delta unit of 1 meter.

Test Seq.	ORB-SLAM2 + PSPNet Acc./ Comp. / F_1	ORB-SLAM2 + PSPNet mIoU	SUMA++ Acc./ Comp. / F_1	SUMA++ mIoU
0	78.5 / **72.8** / **75.5**	**35.3**	**90.8** / 63.1 / 74.5	19.9
1	81.8 / **76.8** / **79.2**	**31.6**	**89.3** / 62.9 / 73.8	17.3
2	82.5 / **70.8** / **76.2**	**30.4**	**89.6** / 64.5 / 75.0	17.6
3	84.3 / **79.1** / **81.6**	**32.7**	**94.2** / 66.3 / 77.8	22.8

(b) Semantic mapping evaluated at a threshold of 10cm.

TABLE 6: **Quantitative Results for Semantic SLAM.** Evaluated on 4 test sequences.

We use the same baselines considered in the localization benchmark. As ORB-SLAM2 does not provide dense reconstruction nor semantic information, we obtain dense semantic reconstruction by unprojecting 2D semantic segmentations (PSPNet [116]) using depth maps from semi-global matching (SGM) [42]. SUMA++ aims for semantic SLAM and estimates poses and a semantic surfel map from LiDAR scans. We experimentally observe that it is sufficient to take the center of the surfels as the reconstructed points.

Table 6b and Fig. 10 show the reconstruction and semantic prediction results. We observed that both baselines produce good reconstructions on the ground region. For regions above the ground, SUMA++ is less complete as it only uses LiDAR scans and thus the maximum height is limited. ORB-SLAM2 + SGM results in higher completeness but worse accuracy. In terms of semantic predictions, SUMA++ produces reasonable results on the LiDAR scans but struggles to achieve good overall performance due to its low completeness. In contrast, ORB-SLAM2 + PSPNet contains more flying points due to the outliers of stereo

matching (e.g., sky points colored blue). Exploring semantic information to remove sky points may further improve the performance of this baseline.

Discussion: We evaluate localization accuracy of existing SLAM methods and suggest exploring 3D instance-level information in further works. Further, reconstructing accurate geometry and semantics remains a challenging task. Our benchmark allows to investigate important questions towards solving this challenging task, e.g., which input modality is better suited for this task, whether semantic prediction and geometric reconstruction can benefit each other and if joint optimization is desirable.

7 Conclusion

We present KITTI-360, a large scale 3D video dataset comprising 300k images and laser point clouds with consistent semantics in both 2D and 3D. We create a WebGL-based annotation tool and annotate both static and dynamic objects in 3D. We propose a method to obtain dense semantic instance labels from annotated 3D primitives. In the presence of 3D data, our method yields better results compared to several 2D label transfer baselines while lowering the annotation time.

Furthermore, we establish novel online benchmarks for several challenging tasks at the intersection of computer vision, graphics and robotics. We evaluate several baselines for each benchmark. Our results show that existing methods achieve satisfactory results on well-established benchmarks, e.g., 2D/3D segmentation, where inference is directly performed on given observations. However, it is much harder to solve tasks that require jointly recovering the geometry, appearance and estimating the semantics as in the newly introduced tasks for semantic scene completion, novel view appearance/semantic synthesis and semantic SLAM. We hope that our dataset, online benchmarks and annotation tools will fertilize new research across communities, fostering progress towards the grand goal of full autonomy.

Acknowledgments

The authors thank Siyuan Peng, Bernhard Jaeger, Shrisha Bharadwaj, Apratim Bhattacharyya, Paul Henderson, and Zehao Yu for their help in implementing the baselines, Kashyap Chitta, Katja Schwarz, and Yue Wang for proofreading, and SurfingTech for annotating parts of the dataset. Andreas Geiger was supported by the ERC Starting Grant LEGO-3D (850533) and the DFG EXC number 2064/1 - project number 390727645. Yiyi Liao was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A.

References

[1] "Daimler urban segmentation dataset," http://www.6d-vision.com/scene-labeling. 3

[2] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," 2018. 4

[3] M. Andriluka, J. Uijlings, and V. Ferrari, "Fluid annotation: a human-machine collaboration interface for full image annotation," 2018. 4

[4] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixe, "4d panoptic lidar segmentation," in *CVPR*, 2021. 4

[5] V. Badrinarayanan, I. Budvytis, and R. Cipolla, "Mixture of trees probabilistic graphical model for video segmentation," *IJCV*, vol. 110, no. 1, pp. 14–29, 2014. 4

[6] V. Badrinarayanan, F. Galasso, and R. Cipolla, "Label propagation in video sequences," in *CVPR*, 2010. 1, 4

[7] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *ICCV*, 2021. 14,

[8] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *ICCV*, 2019. 1, 3, 4,

[9] J. Behley, V. Steinhage, and A. B. Cremers, "Performance of histogram descriptors for the classification of 3d laser range data in urban environments," in *ICRA*, 2012. 3

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *VCG*, vol. 5, no. 4, pp. 349–359, 1999. 8,

[11] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *PAMI*, vol. 26, pp. 1124–1137, 2004. 9

[12] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari, "Semantic monocular SLAM for highly dynamic environments," in *ICRA*, 2018. 15

[13] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 1 2009. 3

[14] T. Bruls, W. Maddern, A. A. Morye, and P. Newman, "Mark yourself: Road marking segmentation via weakly-supervised annotations from multimodal data," in *ICRA*, 2018. 4

[15] I. Budvytis, V. Badrinarayanan, and R. Cipolla, "Label propagation in complex video sequences using semi-supervised learning," in *BMVC*, 2010. 4

[16] I. Budvytis, P. Sauer, T. Roddick, K. Breen, and R. Cipolla, "Large scale labelled video data augmentation for semantic segmentation in driving scenarios," in *ICCV*, 2017. 4

[17] Y. Cabon, N. Murray, and M. Humenberger, "Virtual KITTI 2," *arXiv.org*, vol. 2001.10773, 2020. 4

[18] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020. 1, 3, 4, 12

[19] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-rnn," in *CVPR*, 2017. 4

[20] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," in *3DV*, 2017. 3

[21] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019. 3

[22] L.-C. Chen, S. Fidler, A. L. Yuille, and R. Urtasun, "Beat the mturkers: Automatic image labeling from weak 3d supervision," in *CVPR*, 2014. 4

[23] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *IROS*, 2019. 15,

[24] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," in *CVPR*, 2021. 13

[25] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016. 3, 4, 6, 7, 9, 10, 12,

[26] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017. 3, 4

[27] K. Deng, A. Liu, J. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," *arXiv.org*, vol. 2107.02791, 2021. 14,

[28] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, 2019. 15

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *CoRL*, 2017. 14

[30] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996. 12,

[31] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," in *ICCV*, 2017. 4

[32] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016. 4

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *IJRR*, vol. 32, no. 11, pp. 1231–1237, 2013. 4, 5

[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *CVPR*, 2012. 1, 3, 5

[35] A. Geiger and C. Wang, "Joint 3d object and layout inference from a single rgb-d image," in *GCPR*, 2015. 2

[36] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2D2: audi autonomous driving dataset," *arXiv.org*, vol. 2004.06320, 2020. 3, 4, 12

[37] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017. 15,

[38] M. Guillaumin, D. Küttel, and V. Ferrari, "Imagenet auto-annotation with segmentation propagation," *IJCV*, vol. 110, no. 3, pp. 328–348, 2014. 4

[39] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d.net: A new large-scale point cloud classification benchmark," in *APRS*, vol. IV-1-W1, 2017, pp. 91–98. 3

[40] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *PAMI*, vol. 42, no. 2, pp. 386–397, 2020. 12

[41] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *IROS*, 2013. 5

[42] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *PAMI*, vol. 30, no. 2, pp. 328–341, 2008. 15,

[43] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation," *arXiv.org*, vol. 1612.02649, 2016. 4

[44] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," in *AR*, vol. 34, no. 3. Springer, 2013, pp. 189–206.

[45] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *PAMI*, 2020. 3, 4, 6, 12

[46] S. D. Jain and K. Grauman, "Supervoxel-consistent foreground propagation in video," in *ECCV*, 2014. 4

[47] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends in Computer Graphics and Vision*, 2020. 1

[48] M. Jancosek and T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in *CVPR*, 2011. 8

[49] L. Jiang, H. Zhao, S. Shi, S. Liu, C. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *CVPR*, 2020. 12, 13,

[50] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Level 5 perception dataset 2020," https://level-5.global/level5/data/, 2019. 3

[51] M. Kiefel and P. Gehler, "Human pose estimation with fields of parts," in *ECCV*, 2014. 8

[52] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, "Video panoptic segmentation," in *CVPR*, 2020. 3

[53] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. on Graphics*, vol. 36, no. 4, 2017.

[54] G. Kopanas, J. Philip, T. Leimkühler, and G. Drettakis, "Point-based neural rendering with per-view optimization," *Computer Graphics Forum*, vol. 40, no. 4, pp. 29–43, 2021. 14,

[55] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *NIPS*, 2011. 7, 8, 9,

[56] M. Larsson, E. Stenborg, L. Hammarstrand, M. Pollefeys, T. Sattler, and F. Kahl, "A cross-season correspondence dataset for robust semantic segmentation," in *CVPR*, 2019. 3

[57] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger, "Towards unsupervised learning of generative models for 3d controllable image synthesis," in *CVPR*, 2020. 12

[58] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[59] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-gcn," in *CVPR*, 2019. 4

[60] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing via label transfer," *PAMI*, vol. 33, no. 12, pp. 2368–2382, 2011. 4

[61] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015. 11, 12,

[62] V. Madhavan and T. Darrell, "The bdd-nexar collective: A large-scale, crowsourced, dataset of driving scenes," Master's thesis, 2017. 3

[63] A. Martinović, J. Knopp, H. Riemenschneider, and L. Van Gool, "3d all the way: Semantic segmentation of urban scenes from start to end in 3d," in *CVPR*, 2015. 4

[64] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020. 1, 14,

[65] D. Munoz, J. A. Bagnell, and M. Hebert, "Co-inference machines for multi-modal scene analysis," in *ECCV*, 2012. 3, 4

[66] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," vol. 33, no. 5, pp. 1255–1262, 2017. 15,

[67] A. Mustafa and A. Hilton, "Semantically coherent co-segmentation and reconstruction of dynamic scenes," in *CVPR*, 2017. 4

[68] N. S. Nagaraja, P. Ochs, K. Liu, and T. Brox, "Hierarchy of localized random forests for video annotation," in *DAGM*, 2012. 4

[69] S. T. Namin, M. Najafi, M. Salzmann, and L. Petersson, "A multi-modal graphical model for scene analysis," in *WACV*, 2015. 4

[70] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *ICCV*, 2017. 3

[71] L. Nicholson, M. Milford, and N. Sünderhauf, "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented SLAM," *IEEE Robotics and Automation Letters*, 2019. 15

[72] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *CVPR*, 2021. 12

[73] P. Osborne, "The mercator projections," 2008. [Online]. Available: http://mercator.myzen.co.uk/mercator.pdf 5

[74] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, "Neural scene graphs for dynamic scenes," *CVPR*, 2021. 12

[75] Q.-H. Pham, P. Sevestre, R. S. Pahwa, H. Zhan, C. H. Pang, Y. Chen, A. Mustafa, V. Chandrasekhar, and J. Lin, "A*3d dataset: Towards autonomous driving in challenging environments," in *ICRA*, 2020. 3

[76] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *ICCV*, 2019. 12, 13,

[77] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017. 12,

[78] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017. 12, 13,

[79] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *CVPR*, 2021. 4

[80] S. Qiao, Y. Zhu, H. Adam, A. L. Yuille, and L. Chen, "Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation," in *CVPR*, 2021. 3, 4

[81] M. A. Reza, H. Zheng, G. Georgakis, and J. Kosecka, "Label propagation in rgb-d video," in *IROS*, 2017. 4

[82] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *ECCV*, 2016. 4

[83] G. Riegler and V. Koltun, "Free view synthesis," in *ECCV*, 2020. 14,

[84] H. Riemenschneider, A. Bódis-Szomorú, J. Weissenberg, and L. V. Gool, "Learning where to classify in multi-view semantic segmentation," in *ECCV*, 2014. 3

[85] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology, 1963. 1

[86] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *CVPR*, 2016. 4

[87] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *ICRA*, 2020. 15

[88] X. Roynard, J. Deschaud, and F. Goulette, "Paris-lille-3d: A point cloud dataset for urban scene segmentation and classification," in *CVPR Workshops*, 2018. 3

[89] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *CVPR*, 2017. 13,

[90] M. Schönbein and A. Geiger, "Omnidirectional 3d reconstruction in augmented manhattan worlds," in *IROS*, 2014. 5

[91] M. Schönbein, T. Strauss, and A. Geiger, "Calibrating and centering quasi-central catadioptric cameras," in *ICRA*, 2014. 5

[92] S. M. Seitz and R. Szeliski, "Applications of computer vision to computer graphics," in *Computer Graphics*, 1999. 1

[93] S. Song, S. Lichtenberg, and J. Xiao., "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *CVPR*, 2015. 3

[94] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *CVPR*, 2020. 3

[95] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *ECCV*, 2010. 9, 10,

[96] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang, and J. Li, "Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways," in *CVPR Workshops*, 2020. 1, 3

[97] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *CVPR*, 2018. 4

[98] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *PAMI*, vol. 13, no. 4, pp. 376–380, 1991.

[99] J. P. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. Torr, "Mesh based semantic modelling for indoor and outdoor scenes," in *CVPR*, 2013. 3

[100] S. Vijayanarasimhan and K. Grauman, "Active frame selection for label propagation in videos," in *ECCV*, 2012. 9, 10,

[101] V. Vineet, G. Sheasby, J. Warrell, and P. H. S. Torr, "Posefield: An efficient mean-field based method for joint estimation of human pose, segmentation, and depth," in *EMMCVPR*, 2013. 8

[102] J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari, "Real-time fully incremental scene understanding on mobile platforms," *RA-L*, vol. 3, no. 4, pp. 3402–3409, 2018. 15

[103] M. Weber, J. Xie, M. Collins, Y. Zhu, P. Voigtlaender, H. Adam, B. Green, A. Geiger, B. Leibe, D. Cremers, A. Osep, L. Leal-Taixe, and L.-C. Chen, "STEP: Segmenting and tracking every pixel," in *NeurIPS Datasets and Benchmarks*, 2021. 3

[104] P. H. Winston, "Heterarchy in the m.i.t. robot," MIT Artificial Intelligence Laboratory, Tech. Rep., 1971. 1

[105] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *ICCV*, 2013. 3

[106] J. Xiao and L. Quan, "Multiple view semantic segmentation for street view images." in *ICCV*, 2009. 4

[107] J. Xie, M. Kiefel, M.-T. Sun, and A. Geiger, "Semantic instance annotation of street scenes by 3d to 2d label transfer," in *CVPR*, 2016. 2, 9, 10, 11,

[108] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "Upsnet: A unified panoptic segmentation network," in *CVPR*, 2019. 7,

[109] S. Yang and S. A. Scherer, "Cubeslam: Monocular 3-d object SLAM," vol. 35, no. 4, pp. 925–938, 2019. 15

[110] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar, "Surfelgan: Synthesizing realistic sensor data for autonomous driving," in *CVPR*, 2020. 13

[111] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende, C. Witt, and H. Rashed, "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," in *ICCV*, 2019. 3

[112] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3d objects with differentiable rendering of SDF shape priors," in *CVPR*, 2020. 4

[113] M. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv.org*, vol. 1212.5701, 2012. 9

[114] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *ICCV*, 2013. 2

[115] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018. 14,

[116] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017. 7, 9, 12, 14, 15,

[117] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *CVPR*, 2017. 4

[118] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. D. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *CVPR*, 2019. 4

[119] S. M. I. Zolanvari, S. Ruano, A. Rana, A. Cummins, R. E. da Silva, M. Rahbar, and A. Smolic, "Dublincity: Annotated lidar point cloud and its applications," in *BMVC*, 2019. 1, 3

Yiyi Liao received her Ph.D. degree from the Department of Control Science and Engineering, Zhejiang University, China in 2018. She is currently a PostDoctoral researcher at the Autonomous Vision Group, University of Tübingen and Max Planck Institute for Intelligent Systems, Germany. Her research interests include 3D vision and scene understanding.

Jun Xie received her Ph.D. degree from the Electrical Engineering Department of University of Washington in 2016. She is currently a research engineer at Google. Her research interests include image segmentation and 3D vision.

Andreas Geiger received his Diploma in computer science and his Ph.D. degree from Karlsruhe Institute of Technology in 2008 and 2013. Currently, he is leading the Autonomous Vision Group at the University of Tübingen and the Max Planck institute for Intelligent Systems in Tübingen. His research interests include computer vision, machine learning and scene understanding with a focus on self-driving vehicles.

APPENDIX A
ANNOTATION DATA PREPARATION

A.1 Point Cloud Accumulation

To facilitate annotation, we accumulate all laser measurements in a common world coordinate system and augment them with 3D points from stereo matching [42]. To reduce outliers of stereo matching, we consider only points up to 15m distance, and apply left-right as well as forward-backward consistency checks over 5 frames. We fuse all 3D points sequentially and ignore a point closer than 5 cm to its nearest neighbor in the fused point cloud. This downsampling operation allows for reducing the data loading traffic and memory of the web based annotation tool.

A.2 Dense Point Cloud on Dynamic Objects

Our simple dynamic object detection consists of two steps. In the first step, we apply volumetric fusion over a sequential of laser scans and search for 3D points located in (mostly) free regions. As a dynamic object always moves in the free space, the voxels along its moving trajectory are occupied occasionally and thus the expected status over time should be free. Hence the dynamic points can be detected by finding points inside of all "free" voxels. Specifically, we build a 3D occupancy volumetric grid $\mathcal{V}$ for each batch by fusing Velodyne observations using Octomap [44]. Given a set of measurements $z_{1:t}$ from frame 1 to t, the occupancy probability of each voxel $v \in \mathcal{V}$ is updated as follow:

$$p(v|z_{1:t}) = \begin{cases} \max(\min(p(v|z_{1:t-1}) + p(v|z_t), p_{max}), p_{min}) \\ \qquad\qquad \text{if } p(v|z_{1:t-1}) > p_{min} \\ p_{min} \\ \qquad\qquad \text{if } p(v|z_{1:t-1}) = p_{min} \end{cases} \tag{8}$$

where $p(v|z_{1:t})$ is the log-odds of the probability, p_{max} and p_{min} are the upper bound and lower bound of the log-odds respectively. Note that we clamp a voxel to be free if there is sufficient evidence from previous frames to support its free status. We denote the set of free voxels as $\widetilde{\mathcal{V}}$.

Due to the noise in measurements and poses, the free voxels may also contain many 3D points of static objects as shown in Fig. 11a. It is hard to avoid these false positive detections as each voxel is classified as free or occupied independently. Therefore, we consider a second step to filter out clusterings of noisy detections. Specifically, we segment the original accumulated point cloud into a branch of clusters using the Region Growing algorithm[10], and we calculate the occupancy probability of each cluster c based on the detection in the first step:

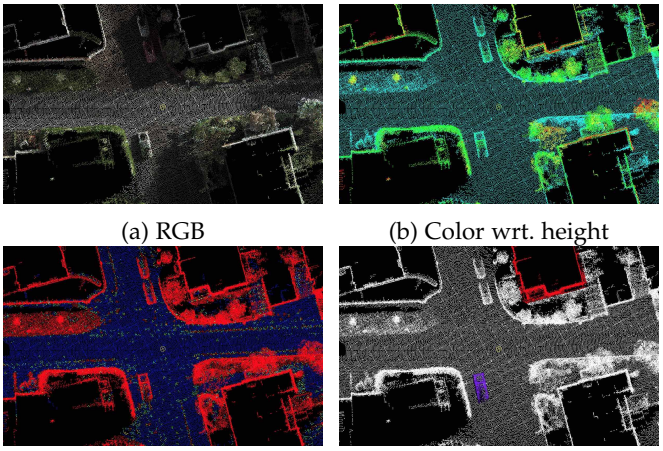
$$p(c) = \frac{1}{N} \sum_{i}^{N} \left[c_i \in \widetilde{\mathcal{V}} \right] \tag{9}$$

where c_i denotes a single point in the cluster, N denotes number of points, and $c_i \in \widetilde{\mathcal{V}}$ means that c_i is spatially located within $\mathcal{V}$. A cluster c is considered as dynamic if $p(c)$ is larger than a given threshold. With the second step, we are able to filter out false positive detections as shown in Fig. 11b. It is acceptable if a few false positive detections remains since the dense point cloud will be further labeled by our annotators.

10. https://pcl.readthedocs.io/projects/tutorials/en/latest/region_growing_segmentation.html

(a) Fine-grained detection (b) Coarse-grained detection

Fig. 11: **Dynamic Object Detection.** (a) Detected dynamic points with decision on 3D grids. (b) Detected dynamic points with decision on point cloud clusters.



(a) RGB (b) Color wrt. height

(c) Color wrt. height variance (d) Color wrt. annotation

Fig. 12: **Color Codings** of 3D points supported by our annotation interface.

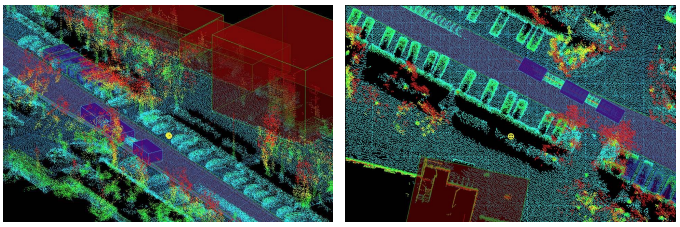
APPENDIX B
ANNOTATION INTERFACE

In this section, we demonstrate the annotation tool and process in detail.

B.1 Annotation Scene

Color Coding: Fig. 12 shows different color codings of 3D points that we provide in the annotation tool. Annotators can choose different color codings accordingly. For example, Fig. 12c helps annotators to identify the boundary between "Road" and "Sidewalk" and Fig. 12d allows annotators to check unannotated region (shown as white).

3D Viewpoint: To assist annotators to better visualize the scene, we also provide different viewports, namely, normal and orthographic viewports as shown in Fig. 13. The orthographic viewport helps annotators accurately identify stuff classes' boundaries and annotate individual objects



(a) Normal viewport (b) Orthographic viewport

Fig. 13: **3D Viewport.**

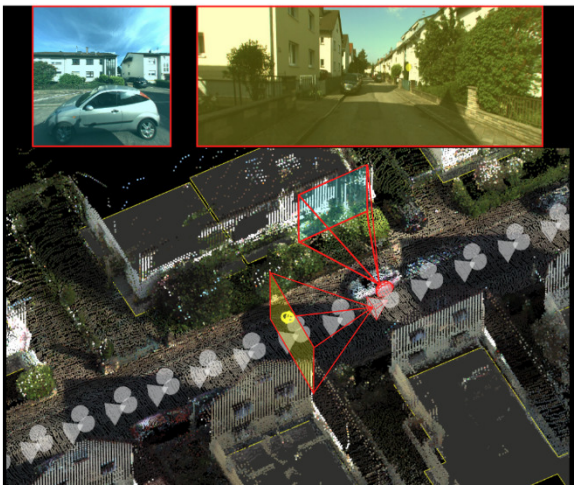


Fig. 14: **2D Camera View.** We illustrate fisheye and perspective camera views and virtual camera poses.

efficiently (see "fast object annotation mode" in Section B.2). Besides, annotators can also adjust point size and brightness to work with different levels of detail.

2D Camera View: For better perceiving the scene, we also show fisheye and perspective images as well as the pose of each camera, enabling annotators to select informative viewpoints efficiently, as shown in Fig. 14.

B.2 Annotation Functions

We provide a few shortcuts and functions to facilitate the annotation process.

Bounding Primitive Manipulation: To best enclose the 3D object, each bounding primitive can be manipulated with translation, scaling, and rotation, resulting in 9 degrees of freedom. In addition, annotators are also asked to assign an orientation to each bounding primitive to understand objects' (especially instances) orientation. See Fig. 15 for each operation's shortcuts.

Bounding Primitive Copy: We also provide a "copy" shortcut to allow annotators to quickly insert new annotations with the same label and similar pose to previously annotated objects. This is especially useful for objects appearing frequently with similar sizes such as building and car.

Fast Object Annotation: We support quickly annotating a subset of object classes by simply drawing a line along the object. This fast annotation mode is enabled under the orthographic view for a few pre-selected classes, where

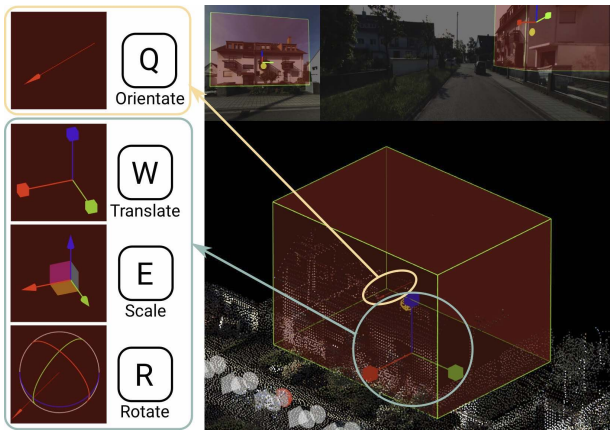


Fig. 15: **Bounding Primitive Manipulation.**

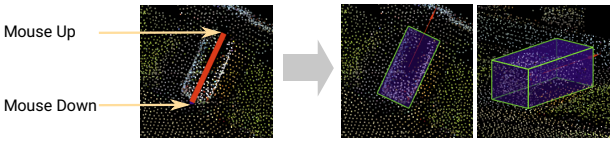


Fig. 16: **Fast Object Annotation.**

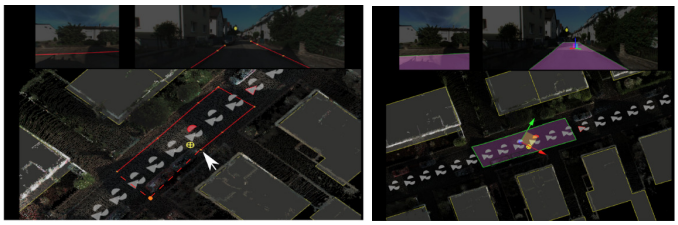
the annotator draws a line along the longest side in the middle. While this line only specifies the length of the object in one dimension and its orientation, we heuristically place a bigger bounding primitive centered at this line and iteratively shrink the bounding primitive until it touches any non-ground points. As shown in Fig. 16, this simple technique allows for efficient and accurate annotation.

Object-Centric Mode: To enable a clear observation of a single object from the accumulated point cloud, we also help annotators with the "object-centric" mode. With a single 3D bounding box selected, triggering the "object-centric" mode hides all the other bounding primitives as well as points far from the selected primitive. In addition, both front view and side views images are automatically switched to the ones in which the selected primitive is most visible.

Completeness Check: As illustrated in Fig. 12d, the annotator can check the completeness level of the annotation by visualizing the point cloud based on existing bounding primitives. Specifically, we color each 3D point if it is enclosed by a bounding primitive and leave the unlabeled region as white. This helps the annotator easily identify any unlabeled 3D points.

B.3 Ground Annotation

Ground bounding primitives are simply annotated as 2D polygons. The extruded height of the ground polygon is automatically determined as follows: for each vertex $v = \{x, y\}$ on the polygon, we first search the nearest camera of this given vertex, and assign the height of the camera to this vertex as its initial height $\hat{z}$. Then we search nearest neighbors of point $\{x, y, \hat{z}\}$ in the 3D point cloud, and update height z as the median height of these nearest neighbors. See Fig. 17 as an example for "Road" annotation. Annotators can also modify the 2D polygon anytime by dragging its control points.



(a) Ground annotation process (b) Finished annotation

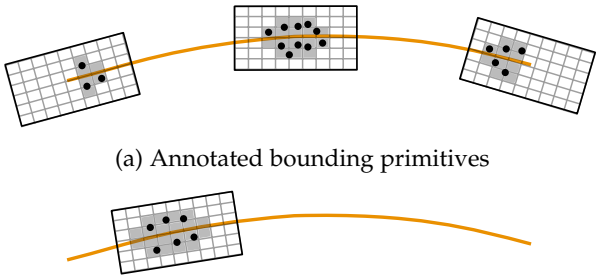
Fig. 17: **Ground Annotation.**

B.4 Dynamic Object Annotation

We implement a semi-automatic annotation scheme to label dynamic objects efficiently. Our semi-automatic annotation relies on two assumptions: the size of the dynamic object is fixed over time, and its trajectory is smooth. Therefore, the required annotation is reduced to adding posed 3D primitives at several keyframes. Our annotation tool then automatically places the remaining primitives along the trajectory. The smooth trajectory is obtained via spline interpolation based on the primitives at the keyframes. We annotate articulated dynamic objects, e.g., pedestrians, using the maximum extent bounding box.

As the speed of the dynamic object may not be constant, we place the remaining primitives based on the observed 3D points at each timestamp. Specifically, we first discretize the annotated 3D primitives into voxels and fuse the occupancy status of each voxel over all annotated primitives as shown in Fig. 18a. A voxel is considered as *occupied* if it is occupied in any of the annotated 3D primitives, otherwise it is *free*. This fused occupancy status is considered an "occupancy template" for searching matching 3D primitive along the trajectory. Given a timestamp between the first and the last annotated timestamp, we slide the 3D primitive on the interpolated spline and calculate the occupancy status based on 3D points collected at the given timestamp. We choose the pose that provides the maximum overlap with the occupancy template, see Fig. 18b.

To facilitate users' interaction, we plot the interpolated spline in the annotation tool and allow the annotator to refine the spline by simply adjusting the 3D bounding primitives inserted at the keyframes. We also display the automatically generated 3D primitives to help the user check if they are accurate. The poses of each automatically generated bounding primitive can also be adjusted if necessary. Typically, it requires $2 \sim 5$ annotated keyframes to produce accurate annotations for the full moving trajectory. Fig. 19 illustrates the dynamic annotation at a given timestamp.



(a) Annotated bounding primitives

(b) Automatically generated bounding primitive

Fig. 18: **Semi-Automatic Dynamic Object Annotation.** (a) We discretize the annotated 3D primitives and fuse the occupancy status. (b) We search along the spline and find the matching bounding primitive by maximizing the overlap.

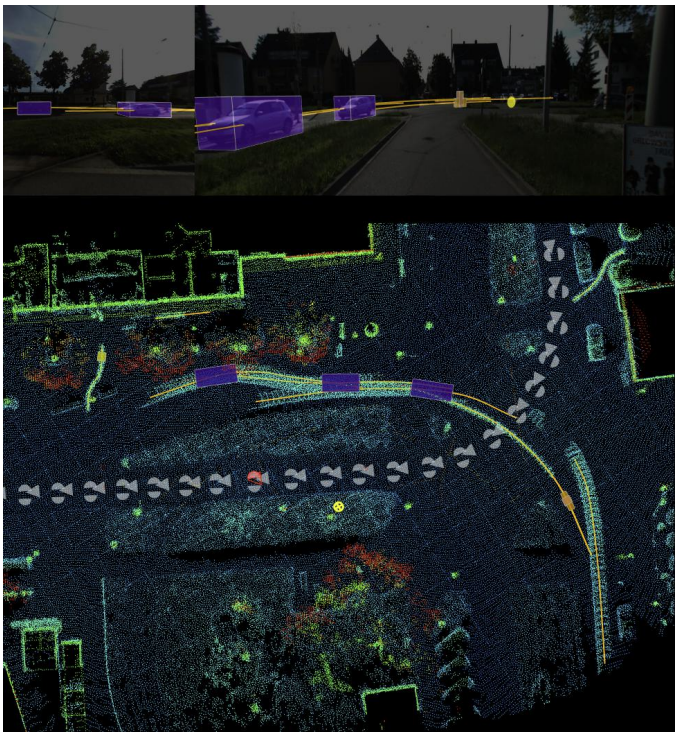


Fig. 19: **Dynamic Annotations.** The above 3D primitives are automatically generated along the interpolated spline visualized in orange.

APPENDIX C
LABEL DEFINITION

Table 7 shows the definition of the 37 classes that we use for annotating 3D scenes. We adhere to the definition of Cityscapes as close as possible while a few inconsistent label definitions are inevitable as our annotations are performed in 3D. For example, the "Traffic Sign" in Cityscapes only includes the front side while we consider both the front and the back. We do not distinguish the back as each traffic sign is labeled by a single 3D bounding primitive and it might be observed in 2D from both sides. Thus, each traffic sign has a consistent label regardless of which side it is observed.

APPENDIX D
MORE DETAILS OF LABEL TRANSFER INFERENCE

D.1 Accumulation of Input Point Cloud

In contrast to the point cloud accumulation for *annotation* as introduced in Appendix A, here we need to distinguish static and dynamic points for label transfer *inference* and thus determine visible points on every frame. Specifically, we consider a point static if it is not enclosed by any dynamic bounding primitive and accumulate all static points

Category	Class	Instance	Definition
flat	road	✗	Horizontal surfaces on which cars usually drive, including road markings. Typically delimited by curbs, rail tracks, or parking areas. However, road is not delimited by road markings and thus may include bicycle lanes or roundabouts.
	sidewalk	✗	Horizontal surfaces designated for pedestrians or cyclists. Delimited from the road by some obstacle, e.g. curbs or poles (might be small), but not only by markings. Often elevated compared to the road and often located at the side of a road. The curbs are included in the sidewalk label. Also includes the walkable part of traffic islands, as well as pedestrian-only zones, where cars are not allowed to drive during regular business hours. If it's an all-day mixed pedestrian/car area, the correct label is ground.
	parking	✗	Horizontal surfaces that are intended for parking and separated from the road, either via elevation or via a different texture/material, but not separated merely by markings.
	rail track	✗	Horizontal surfaces on which only rail cars can normally drive. If rail tracks for trams are embedded in a standard road, they are included in the road label.
construction	building †	✓	Includes structures that house/shelter humans, e.g. low-rises, skyscrapers. Translucent buildings made of glass still receive the label building. Also includes scaffolding attached to buildings.
	garage*	✓	Structures for parking.
	wall	✗	Individually standing walls that separate two (or more) outdoor areas, and do not provide support for a building.
	fence	✗	Structures with holes that separate two (or more) outdoor areas, sometimes temporary.
	guard rail	✗	Metal structure located on the side of the road to prevent serious accidents. Rare in inner cities, but occur sometimes in curves. Includes the bars holding the rails
	bridge	✗	Bridges (on which the ego-vehicle is not driving) including everything (fences, guard rails) permanently attached to them.
	tunnel	✗	Tunnel walls and the (typically dark) space encased by the tunnel, but excluding vehicles.
	gate*	✗	A passageway or opening in a wall of fence for entrance or exit.
	stop*	✓	A place where a bus or train stops for passengers to get on or off. It intends to protect waiting pedestrians from rain, wind and snow.
object	pole†	✓	Long, vertically oriented poles, e.g. sign poles or traffic light poles. This does not include objects mounted on the pole, which have a larger diameter than the pole itself (e.g. most street lights).
	smallpole*	✓	Small, vertically oriented poles, e.g. sign poles or traffic light poles. This does not include objects mounted on the pole, which have a larger diameter than the pole itself (e.g. most street lights).
	traffic light	✓	The traffic light box without its poles in all orientations and for all types of traffic participants, e.g. regular traffic light, bus traffic light, train traffic light
	traffic sign †	✓	Signs installed by the state/city authority with the purpose of conveying information to drivers/cyclists/pedestrians, e.g. traffic signs, parking signs, direction signs, or warning reflector posts. *Both frontal and back side are included.*
	lamp*	✓	A lamp usually mounted on a pole and constituting one of a series spaced at intervals along a public road or highway.
	trash bin*	✓	A container that holds materials that have been thrown away.
	vending machine*	✓	An automated machine for selling merchandise.
	box*	✓	Any rigid typically rectangular container excluding trash bin and vending machine. Some examples are electric box, honey bucket, package, etc.
nature	vegetation	✗	Trees, hedges, and all kinds of vertically growing vegetation. Plants attached to buildings/walls/fences are not annotated separately, and receive the same label as the surface they are supported by.
	terrain	✗	Grass, all kinds of horizontally spreading vegetation, soil, or sand. These are areas that are not meant to be driven on. This label may also include a possibly adjacent curb. Single grass stalks or very small patches of grass are not annotated separately and thus are assigned to the label of the region they are growing on.
sky	sky	✗	Open sky (without tree branches/leaves)
human	person	✓	All humans that would primarily rely on their legs to move if necessary.
	rider	✓	Humans relying on some device for movement. This includes drivers, passengers, or riders of bicycles, motorcycles.
vehicle	car	✓	This includes cars, jeeps, SUVs, vans with a continuous body shape (i.e. the driver's cabin and cargo compartment are one). Does not include trailers, which have their own separate class.
	truck	✓	This includes trucks, vans with a body that is separate from the driver's cabin, pickup trucks, as well as their trailers.
	bus	✓	This includes buses that are intended for 9+ persons for public or long-distance transport.
	caravan	✓	Vehicles that (appear to) contain living quarters. This also includes trailers that are used for living and has priority over the trailer class.
	trailer	✓	Includes trailers that can be attached to any vehicle, but excludes trailers attached to trucks. The latter are included in the truck label.
	train	✓	All vehicles that move on rails, e.g. trams, trains.
	motorcycle	✓	This includes motorcycles, mopeds, and scooters without the driver or other passengers. The latter receive the label rider
	bicycle	✓	This includes bicycles without the cyclist or other passengers. The latter receive the label rider.
void	unknown construction	✓	All remaining construction regions which are not mentioned in the "construction" session
	unknown vehicle	✓	All remaining vehicle regions which are not mentioned in the "vehicle" session
	unknown object	✓	All remaining object regions which are not mentioned in the "object" session

* classes do not exist in Cityscapes
† classes with definitions slightly different from Cityscapes

TABLE 7: **Label Definition.** We adhere to the label definition of Cityscapes as close as possible. Inconsistent classes are marked.

first. For each dynamic object, we retrieve all points inside the corresponding bounding primitives $\{b_t^m\}$ for every timestamp t and accumulate them in the canonical object-centered coordinate system by taking the inverse transform of the object pose defined by $\{b_t^m\}$ (world-to-object transformation). Next, we insert the accumulated dynamic point clouds back to the world coordinate following the object pose (object-to-world transformation). This allows us to obtain dense 3D points during inference for both static and dynamic regions.

D.2 Accumulation of Inferred 3D Label

Our inference is performed individually on each frame defined over the corresponding 2D pixels and visible 3D points. To obtain 3D labels on the accumulated point clouds, we thus fuse 3D labels obtained from each frame. Specifically, if a 3D point is visible in multiple frames, we take the majority of its inferred classes as its final label. The confidence of this 3D point is also averaged over confidence values of these points of the majority label. If a 3D point is not visible in any of the frames but is uniquely labeled by a single class, we assign this unique label to the 3D point and a confidence value of 1.0. In the remaining cases, we treat the 3D point's label as unknown.

D.3 Pixel Unary Potentials of Ground Objects

The first term of the pixel unary potential is a binary feature $\xi_i^{\mathcal{P}}(s_i) \in \{0, 1\}$ which indicates admissible labels. For non-planar object classes, $\xi_i^{\mathcal{P}}(s_i)$ is obtained based on the projection of *3D* primitives, whereas planar object classes directly use projections of *2D* polygons to obtain more accurate boundaries. As introduced in Appendix B.3, the ground bounding primitives are extruded to 3D to enclose the 3D points. This leads to oversized 2D projections, making it hard to determine the boundary between two adjacent ground object annotations (e.g., "Road" and "Sidewalk"). As opposed to our conference version [107], which exploits a geometric unary potential to address this problem, we instead directly project the 2D polygons of the ground objects before extruding them to 3D. This allows us to obtain accurate ground object boundaries from $\xi_i^{\mathcal{P}}(s_i)$ and avoid the complexity introduced by the additional geometric unary term.

D.4 Instance Augmentation of Pixel Unary Potentials

We adopt a state-of-the-art panoptic segmentation method UPSNet [108] pre-trained on Cityscapes [25] to obtain instance hypotheses for "Car", "Truck", and "Pedestrian". We first run UPSNet on our images to get probability maps of all instances. For each annotated instance within the aforementioned classes, we retrieve matched instances from the predictions of UPSNet. More specifically, given a set of 3D points annotated with one instance (e.g., one car) and a probability map of one instance predicted by UPSNet, we consider they match if more than 50% of the 3D points fall into the high-probability region of the predicted instance. This allows for improving instance boundaries as shown in Fig. 20.

APPENDIX E
MORE RESULTS OF LABEL TRANSFER INFERENCE

E.1 Detailed Quantitative Comparisons

Here, we show detailed quantitative comparisons for individual classes. Table 8 and Table 9 show quantitative comparison to label transfer baselines on static and dynamic objects, respectively. We evaluate the intersection over union (IoU) of each class where the mIoU is the average over all classes. We further show detailed ablation study in Table 10 and Table 11 for semantic and instance label transfer.

E.2 Qualitative Comparison to Baselines

We compare our method qualitatively to several 2D-to-2D and 3D-to-2D label transfer baselines in Fig. 21. Note how the 2D-to-2D label transfer baselines fail in the presence of strong occlusions and large displacements.

E.3 Qualitative Comparison of Ablation Study

Fig. 22 compares different variants of our label transfer model. Consistent with the quantitative analysis, our full model achieves the best performance.

APPENDIX F
DATASET

F.1 Statistical Analysis

Fig. 23 shows the distribution of the semantic labels in KITTI-360. Fig. 23a and Fig. 23b suggests that the semantic distribution of the 2D pixels and the 3D points are similar (except for the "Sky" class). We also show the distribution of our 3D bounding boxes in Fig. 23c.

F.2 Dataset Split

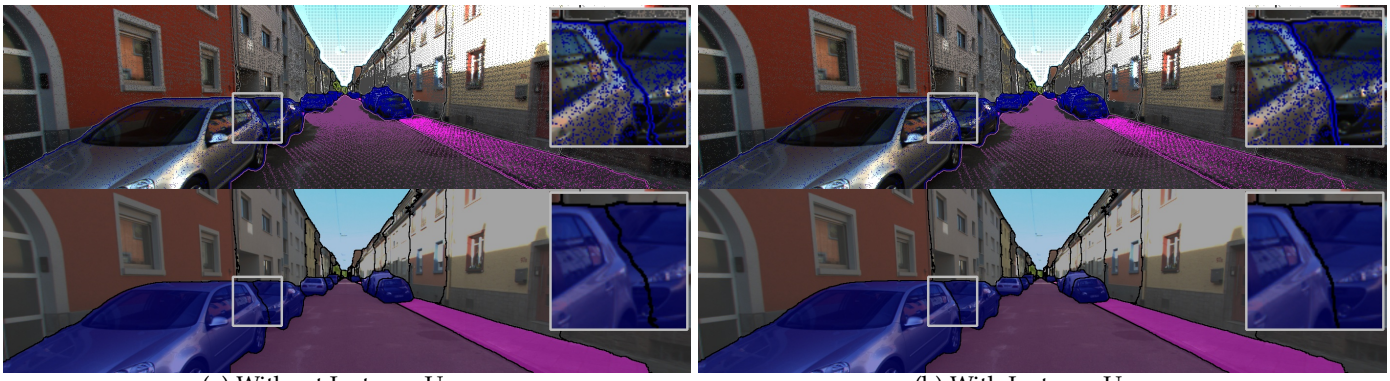
We split KITTI-360 into training and test sets without spatial overlapping as shown in Fig. 24. We maintain an online evaluation server and hold back the labels of the test set. Considering that different tasks involve different label modalities, the test set is further divided into two parts with different information released. Specifically, the first part of the test dataset is used for semantic scene understanding (except for semantic scene completion) and novel view synthesis, where we hold back the 2D semantic/instance segmentation maps and 3D pointwise labels. Note that the accumulated point clouds are released while their labels are removed. The other part is adopted for semantic scene completion and semantic SLAM where the accumulated point clouds are further removed. We release vehicle poses for both test sets.

APPENDIX G
SEMANTIC SCENE UNDERSTANDING BENCHMARK

G.1 Benchmark of 2D Semantic segmentation

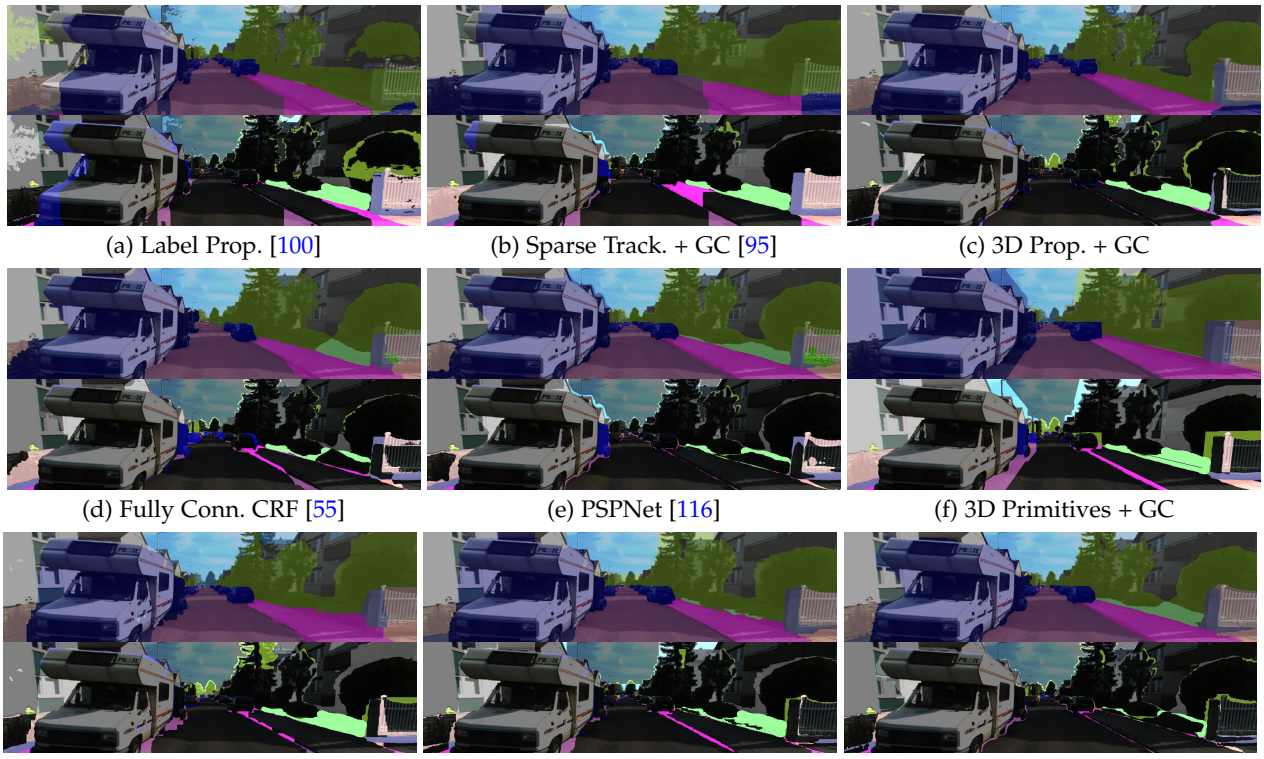
G.1.1 Evaluation Metrics

We evaluate confidence weighted mIoU where both the intersection and the union are weighted (per-pixel) by the confidence of our pseudo-ground truth. More formally, let {TP} and {TP, FP, FN} denote the set of image pixels in the



(a) Without Instance Unary (b) With Instance Unary

Fig. 20: **Ablation Study of Instance Unary.** The top row shows the input image with the projected 3D points and inferred semantic segmentation boundaries. The second row shows the inferred semantic instance segmentation. As can be seen from (a), it is challenging to distinguish instance boundaries given sparse projections of point clouds. This can be effectively improved by incorporating instance unary as shown in (b).



(a) Label Prop. [100] (b) Sparse Track. + GC [95] (c) 3D Prop. + GC

(d) Fully Conn. CRF [55] (e) PSPNet [116] (f) 3D Primitives + GC

(g) 3D Mesh + GC (h) 3D Points + GC (i) Proposed Method

Fig. 21: **Qualitative Comparison to Baselines.** Each subfigure shows from top-to-bottom: the input image with inferred semantic segmentation and the errors with respect to 2D ground truth annotation where colors indicate ground truth labels.

intersection and the union of one class label (or one category label), respectively. The weighted IoU of this class can be defined as follow:

$$\text{IoU} = \frac{\sum_{i \in \{\text{TP}\}} c_i}{\sum_{i \in \{\text{TP, FP, FN}\}} c_i} \tag{10}$$

where $c_i \in [0, 1]$ denotes the confidence value at pixel i. In the standard evaluation $c_i = 1$ for all pixels. The mIoU is then calculated as the mean of the weighted IoU over all class labels or category labels.

While we provide 19 classes for training following Cityscapes [25], we omit two classes, "Train" and "Bus" during evaluation since these two classes are rarely observed in the test region when we split the training and test sets according to the camera poses as shown in Fig. 24.

G.1.2 Baselines

We train and evaluate two well-known methods, Fully Convolutional Neural Network (FCN) [61] and Pyramid Scene Parsing Network (PSPNet) [116]. For FCN, we adopt the ResNet-101 model provided by PyTorch as a backbone. The model is pre-trained on a subset of the Microsoft COCO dataset. As for PSPNet, we use the official PyTorch imple-

(a) LA (b) LA + 3D (c) LA + PW

(d) LA + PW + CO (e) LA + PW + CO + 3D (f) Full Model

Fig. 22: **Qualitative Results for Ablation Study.** Each subfigure shows from top-to-bottom: the input image with inferred semantic segmentation, and the errors with respect to 2D ground truth annotation where colors indicate ground truth labels.

Method	Road	Park	Sdwlk	Terr	Bldg	Vegt	Car	Trler	Crvn	Gate	Wall	Fence	Box	Sky	mIoU	Acc
Label Prop. [100]	85.9	27.8	59.4	48.7	78.8	67.5	51.0	12.6	48.1	25.8	47.1	44.3	0.0	89.6	49.0	81.0
Sparse Track. + GC [95]	83.4	33.0	38.7	49.8	76.2	61.0	75.1	73.5	78.8	24.6	6.7	25.0	11.6	79.0	51.2	79.1
3D Prop. + GC	77.1	50.5	75.4	64.5	83.1	76.8	82.1	91.2	**91.7**	62.6	59.9	58.1	55.5	81.6	72.1	87.4
Fully Conn. CRF [55]	90.1	46.4	67.4	61.3	88.3	78.4	85.6	48.9	78.1	30.5	33.7	45.6	43.1	92.7	63.6	88.7
PSPNet [116]	**95.6**	46.2	77.1	64.8	88.9	81.7	91.5	46.5	84.0	30.6	41.7	50.2	52.3	89.2	67.2	90.4
3D Primitives + GC	81.7	31.4	45.9	22.5	59.6	56.7	63.0	67.1	61.7	42.3	25.5	52.3	31.3	50.3	49.4	73.4
3D Mesh + GC	91.7	54.6	67.6	31.4	81.3	72.1	85.2	93.5	86.0	59.4	35.9	61.2	50.1	65.6	66.8	85.7
3D Points + GC	93.5	62.2	76.5	37.2	82.0	74.1	87.5	**94.7**	85.7	73.2	52.2	69.0	**61.1**	68.0	72.6	87.8
Proposed Method	95.2	**72.9**	**84.5**	**67.9**	**90.3**	**84.2**	**92.2**	93.4	90.8	**78.8**	**64.3**	**73.1**	56.8	**92.8**	**81.2**	**93.1**

TABLE 8: **Comparison to Label Transfer Baselines on Semantic Segmentation Transfer of Static Objects**. We compare our method to 2D label transfer baselines (top) and to 3D to 2D label transfer baselines (bottom) on 120 consecutive images of static objects.

Method	Car	Truck	Trailer	Motor	Bicycle	Rider	Person	mIoU	Acc
Label Prop. [100]	28.5	53.2	**71.7**	11.1	42.0	32.2	21.8	37.2	59.1
Sparse Track. + GC [95]	10.1	22.0	13.2	5.2	1.4	5.6	0.1	8.2	12.5
3D Prop. + GC	15.7	40.3	24.1	2.2	0.2	2.0	17.1	14.5	21.7
Proposed Method	**77.7**	**85.1**	69.8	**60.5**	**42.9**	**49.5**	**59.0**	**63.5**	**94.1**

TABLE 9: **Comparison to Label Transfer Baselines on Semantic Segmentation Transfer of Dynamic Objects**. We compare our method to 2D label transfer baselines (top) and to 3D to 2D label transfer baselines (bottom) on 120 consecutive images that contain dynamic objects.

mentation[11] which also uses ResNet-101 as backbone. The model is pre-trained on the ImageNet dataset.

G.1.3 Additional Results

We show the IoU of each class in Table 12a. We observe that PSPNet consistently outperforms FCN in most of the classes.

G.2 Benchmark of 2D Instance Segmentation

G.2.1 Evaluation Metric

Following [58], we measure the Average Precision (AP) over 10 IoU thresholds, ranging from 0.5 to 0.95 with a step size of 0.05. We calculate confidence weighted IoU per *instance* using Eq. 10. In this task, we consider 7 classes that contain instance labels, including "Building", "Person", "Rider", "Car", "Truck", "Motorcycle" and "Bicycle".

G.2.2 Baselines

We evaluated two Mask R-CNN models with different backbones, i.e., ResNet-50 and ResNet-101, based on the official implementation[12]. Both backbones are pre-trained on the ImageNet dataset.

G.2.3 Additional Results

Table 12b shows the AP of each individual class as well as the mean AP. We observe that performance of different backbones is similar in more frequently observed classes (e.g., "Building" and "Car") while differs in less occurred classes.

11. https://github.com/hszhao/semseg

12. https://github.com/facebookresearch/detectron2

Method	Road	Park	Sdwlk	Terr	Bldg	Vegt	Car	Trler	Crvn	Gate	Wall	Fence	Box	Sky	mIoU	Acc
LA	95.4	34.0	48.5	66.6	88.4	82.8	91.8	54.2	88.9	61.7	53.8	52.0	53.6	89.6	68.7	89.1
LA+3D	95.3	48.9	75.7	66.6	85.8	81.5	92.1	56.1	**91.2**	68.3	58.8	45.7	48.4	88.7	71.7	90.1
LA+PW	95.4	26.5	38.3	66.2	88.5	83.5	92.0	56.9	89.4	66.2	51.6	50.0	51.2	89.8	67.5	88.2
LA+PW+CO	**95.5**	70.0	82.5	67.4	89.8	83.7	**92.4**	87.2	89.0	75.4	57.9	68.3	**60.0**	89.9	79.2	92.5
LA+PW+CO+3D	95.1	72.7	84.0	67.3	90.3	84.1	92.2	93.1	90.8	77.3	63.0	72.1	56.7	**92.8**	80.8	93.0
Full Model	95.2	**72.9**	**84.5**	**67.9**	**90.3**	**84.2**	92.2	**93.4**	90.8	**78.8**	**64.3**	**73.1**	56.8	92.8	**81.2**	**93.1**
Full Model (90%)	97.5	83.1	92.0	80.3	93.9	90.1	95.1	95.0	93.2	86.3	74.6	81.4	79.9	93.7	88.3	96.0
Full Model (80%)	98.4	**89.2**	94.2	89.5	96.4	94.2	96.6	96.3	95.5	93.5	80.2	86.4	90.0	95.3	92.5	97.6
Full Model (70%)	**98.8**	88.8	**95.0**	**92.2**	**97.7**	**95.8**	**97.3**	**97.1**	**96.7**	**96.1**	**85.2**	**88.2**	**95.0**	**96.3**	**94.3**	**98.4**

TABLE 10: **Ablation Study on Semantic Segmentation of Static Objects.** This table shows the importance of the different components in our model on all 120 images. The components are abbreviated as follows: LA = local appearance ($p^{\mathcal{P}}$), PW = 2D pairwise constraints ($\psi^{\mathcal{P},\mathcal{P}}$), CO = 3D primitive constraints ($\xi^{\mathcal{P}}$), 3D = 3D points ($\varphi^{\mathcal{L}}$,$\psi^{\mathcal{P},\mathcal{L}}$), Full Model = all potentials including 3D pairwise constraints ($\psi^{\mathcal{L},\mathcal{L}}$). Percentages denote fractions of estimated pixels with highest confidence.

Method	Bldg	Car	Trler	Crvn	Box	mIoU	Acc
LA	79.0	88.4	54.4	89.3	50.3	72.3	86.7
LA+3D	77.6	89.6	56.3	**91.5**	49.7	72.9	88.7
LA+PW	79.0	90.3	57.2	89.8	48.4	72.9	85.8
LA+PW+CO	82.2	**90.6**	87.7	89.4	58.2	81.6	91.0
LA+PW+CO+3D	**84.4**	**90.6**	93.6	91.2	58.2	83.6	91.7
Full Model	**84.4**	**90.6**	**93.8**	91.2	**58.6**	**83.7**	**91.8**
Full Model (90%)	88.8	93.4	95.3	93.4	74.3	89.0	94.9
Full Model (80%)	92.0	94.8	96.4	95.6	77.8	91.3	96.6
Full Model (70%)	**93.6**	**95.7**	**97.2**	**96.8**	**79.9**	**92.7**	**97.4**

TABLE 11: **Comparison to Label Transfer Baselines on Instance Segmentation on Static Objects**. We compare our method to 2D label transfer baselines (top) and to 3D to 2D label transfer baselines (bottom) on 120 consecutive images.

Method	Road	Sdwlk	Bldg	Wall	Fence	Pole	Trlgt	Trsgn	Vegt	Terr	Sky	Persn	Rider	Car	Truck	Motor	Bicyc	$\text{mIoU}_{\text{class}}$
FCN [61]	95.6	84.5	84.1	43.4	38.6	31.1	0.0	38.0	90.6	85.7	91.2	40.5	29.3	94.6	42.4	28.4	0.0	54.0
PSPNet [116]	96.6	87.3	87.0	65.0	55.6	40.1	0.0	43.0	92.6	88.4	91.9	55.5	48.3	95.6	60.4	52.1	44.1	64.9

(a) 2D Semantic Segmentation

Method	Bldg	Persn	Rider	Car	Truck	Motor	Bicyc	AP
ResNet50	26.5	27.2	10.9	53.2	6.2	5.3	7.3	19.5
ResNet101	27.0	22.9	15.9	52.0	10.2	12.7	5.7	20.9

(b) 2D Instance Segmentation

Method	Bldg	Car	AP_{50}	Bldg	Car	AP_{25}
BoxNet [76]	8.2	0.0	4.1	46.6	0.6	23.6
VoteNet [76]	5.7	1.1	3.4	40.3	20.9	30.6

(c) 3D Bounding Box Detection

Method	Road	Sdwlk	Bldg	Wall	Fence	Pole	Trlgt	Trsgn	Vegt	Terr	Sky	Persn	Rider	Car	Truck	Motor	Bicyc	$\text{mIoU}_{\text{class}}$
PointNet [77]	54.2	14.4	28.1	2.7	1.4	0.3	0.0	2.2	56.5	16.4	–	0.0	–	19.9	0.0	0.0	0.0	13.1
PointNet++ [78]	82.1	66.3	62.1	30.5	24.9	38.3	0.0	23.4	71.2	47.3	–	2.0	–	84.8	0.5	1.6	0.0	35.7

(d) 3D Semantic Segmentation

Method	Bldg	Car	AP
PointNet++ [78]+ [30]	11.5	35.9	23.7
PointGroup [49]	9.9	59.6	34.8

(e) 3D Instance Segmentation

Method	Road	Sdwlk	Bldg	Wall	Fence	Pole	Trlgt	Trsgn	Vegt	Terr	Sky	Persn	Rider	Car	Truck	Motor	Bicyc	$\text{mIoU}_{\text{class}}$
Enc-Dec	42.0	16.9	16.2	2.3	0.1	4.7	0.0	2.6	25.4	9.0	–	0.0	–	16.2	0.0	0.0	0.0	9.1

(f) Semantic Scene Completion

TABLE 12: **Additional Quantitative Results for Semantic Scene Understanding.** In each table, we show the performance of individual classes with the overall metric in the last column.

G.3 Benchmark of 3D Bounding Box Detection

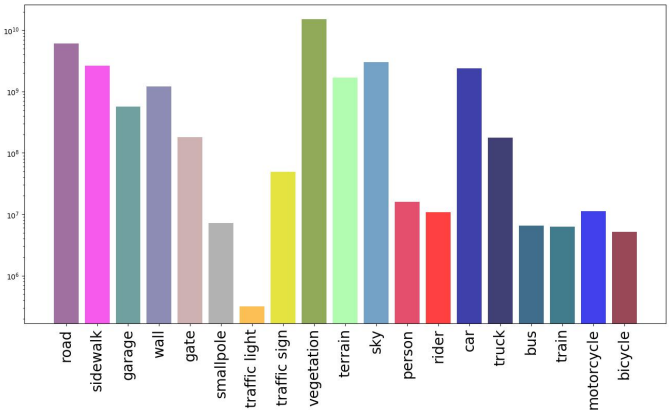
G.3.1 Evaluation Metric

We evaluate AP at a threshold of 0.5 and 0.25 for 3D bounding box detection. As it is particularly challenging for learning-based algorithms to generalize well to other classes with fewer training samples, we measure the mean AP over two classes: "Building" and "Car".

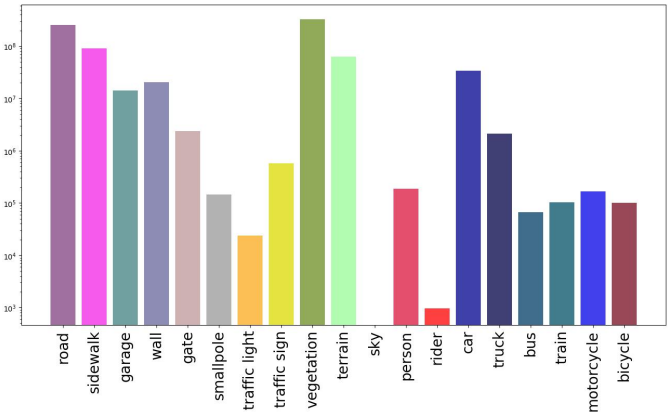
G.3.2 Baselines

We evaluate the state-of-the-art 3D bounding box detection method, VoteNet [76], and its simplified version, BoxNet [76] as baselines. We adopt the official implementation [13] for both methods.

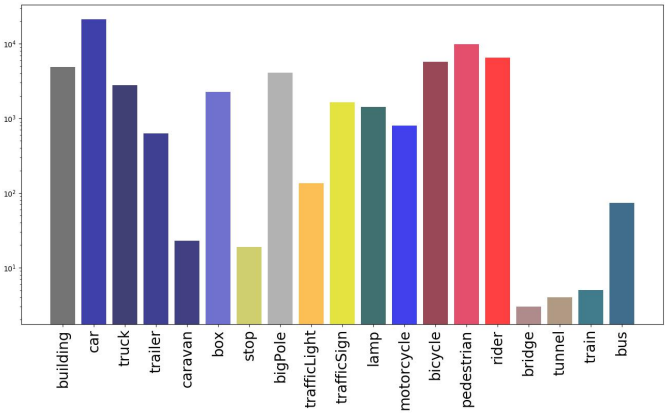
13. https://github.com/facebookresearch/votenet



(a) Distribution of 2D semantic labels over 78k frames.



(b) Distribution of 3D semantic labels over 1B points.



(c) Distribution of 3D semantic labels over 68k bounding boxes.

Fig. 23: **Dataset Statistics.** (a) and (b) show the histogram of the 19 training semantic classes in 2D and 3D, respectively. (c) shows the class distribution of 3D bounding boxes that contains instance IDs.

G.3.3 Additional Results

Table 12c shows the AP on each class as well as the mean AP. Both methods achieve reasonable performance at the IoU threshold of 0.25 while struggle at the higher threshold.

G.4 Benchmark of 3D Semantic Segmentation

G.4.1 Evaluation Metric

For 3D semantic segmentation, we also evaluate confidence weighted mIoU using Eq. 10. Here, the confidence of each 3D point is obtained by averaging the confidence of 3D points on multiple frames, as introduced in Appendix D.2. Similar to 2D semantic segmentation, we omit "Train" and "Bus" during evaluation. Note that there is no "Sky" point in 3D, thus it is also discarded. Moreover, since we train and evaluate only in static regions, we ignore "Rider" as it only appears as a dynamic object.

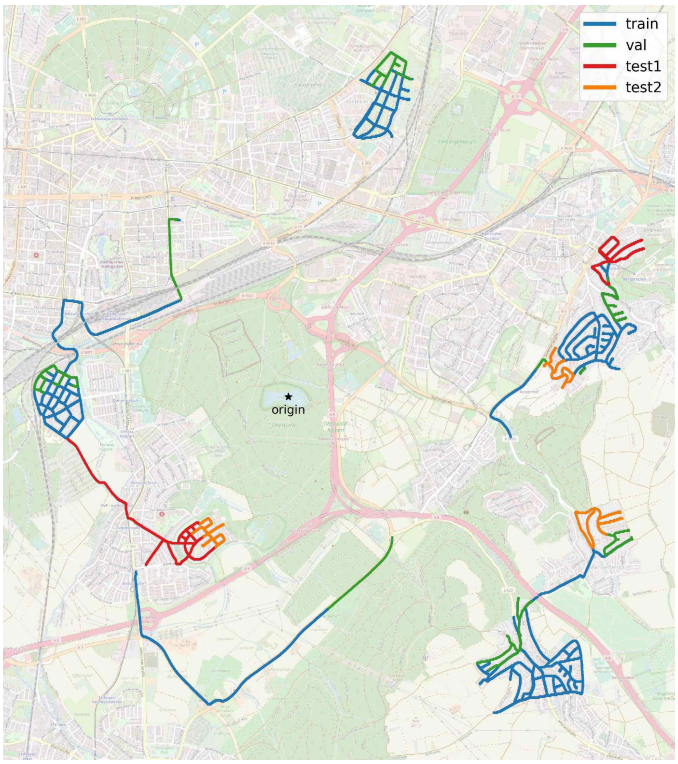


Fig. 24: **Dataset Split.** We split the dataset into one training set, one validation set, and two test sets with held-out ground truth.

G.4.2 Baselines

We train and evaluate two baselines, PointNet [77] and PointNet++ [78]. As the original implementations are built on Tensorflow, we adopt a faithful Pytorch reimplementation[14] that contains both methods.

G.4.3 Additional Results

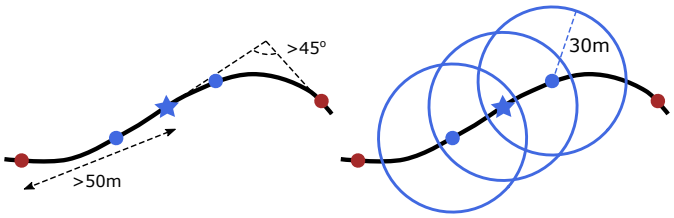
Table 12d shows the detailed results of 3D semantic segmentation. As expected, PointNet++ achieves better performance on all classes compared to PointNet.

G.5 Benchmark of 3D Instance Segmentation

G.5.1 Evaluation Metric

In 3D instance segmentation, we also evaluate AP over 10 IoU thresholds, ranging from 0.5 to 0.95 with a step size of 0.05. The IoU of each 3D instance is weighted (per-point) by the confidence of our pseudo-ground truth. Here, we evaluate on "Building" and "Car" the same as the 3D box bounding detection benchmark.

14. https://github.com/yanx27/Pointnet_Pointnet2_pytorch



(a) Valid neighbor points (b) Crop point cloud

Fig. 25: **Ground Truth Generation for Scene Completion Benchmark.** The blue star denotes the center of the input laser scan. The blue and red dots denote valid and invalid neighbors along the corridor, respectively.

G.5.2 Baselines

We first consider a naïve baseline based on the results we obtained from 3D semantic segmentation using PointNet++ [78]. Specifically, we first extract points of the same class label ("Building" or "Car") based on the semantic segmentation results. Next, we group the extracted point cloud using DBSCAN [30][15], where clusters with less than 500 points are ignored. Each valid cluster is then considered as an instance with a confidence score of 1.0. The second baseline is a state-of-the-art approach, PointGroup [49]. We follow the official implementation[16] to train and evaluate this baseline.

G.5.3 Additional Results

Table 12e shows the detailed results of 3D instance segmentation. Interestingly, the 3D instance segmentation performance of "Car" is higher than the 2D baselines in Table 12b. We hypothesize that unlike in 2D where occlusions strongly impact the results, cars can be more easily separated in 3D.

G.6 Benchmark of Semantic Scene Completion

G.6.1 Data Preparation

The ground truth of the semantic scene completion task is the accumulated point cloud within a corridor of 30m around the vehicle poses of a 100m trajectory (50m in each direction), see Fig. 25 for an illustration. The input to this task is a single LiDAR scan whose center is visualized by the blue star point. We first determine a set of neighboring vehicle poses close to the given center illustrated in Fig. 25a, and then crop the accumulated point cloud using the union of circles located at those poses as shown in Fig. 25b. To avoid evaluating in significantly occluded regions that typically occur when the vehicle turns a large angle, we also check the orientation of each pose as shown in Fig. 25a. Specifically, if the forward direction of one pose deviates more than 45° compared to the heading angle of the given center, it is eliminated from the neighboring poses.

G.6.2 Evaluation Metric

In this task we evaluate geometric completion and semantic estimation, respectively. Geometric completion is evaluated via completeness and accuracy at a threshold of 20cm. Completeness is calculated as the fraction of ground truth points of which the distances to their closest reconstructed points are below the threshold. Accuracy instead measures the percentage of reconstructed points that are within a distance threshold to the ground truth points. As our ground truth reconstruction may not be complete, we prevent punishing reconstructed points by dividing the space into observed and unobserved regions, which are determined by the unobserved volume from a 3D occupancy map obtained using OctoMap [44]. A reconstructed point is only evaluated when it falls into the observed region within the union of the neighboring circles shown in Fig. 25b. We further measure the F_1 score as the harmonic mean of the completeness and the accuracy. Note that SemanticKITTI [8] also considers a semantic scene completion task, but considers voxel as representation and measures mIoU over voxels for both reconstruction and semantics. We instead avoid discretization and directly evaluate on point clouds using standard metrics to separately assess accuracy and completeness.

G.6.3 Baselines

We implement two baselines for this task. For calibration, the first baseline returns the input LiDAR scan as output. The second baseline is a learning-based approach that adopts an encoder-decoder structure. Specifically, the encoder first learns features from the input point cloud. It then merges the point-wise features to voxels such that a 3D U-Net is applied to predict a volumetric reconstruction. The network is trained using a cross-entropy loss where the ground truth point cloud is also discretized into a volume. As our evaluation server requires submission in the form of point clouds, we uniformly and densely sample points from each occupied voxel as the final output.

G.6.4 Additional Results

Table 12f shows detailed semantic estimation performance of the learning-based baseline. As can be seen, it is challenging to predict the geometry and the semantics jointly. The overall performance of this baseline is worse compared to baselines that directly perform 3D semantic segmentation in Table 12d.

APPENDIX H
NOVEL VIEW SYNTHESIS BENCHMARK

H.1 Benchmark of Novel View Appearance Synthesis

H.1.1 Evaluation Metric

We adopt three standard metrics to evaluate novel view appearance synthesis: peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM), and perceptual metric (LPIPS) [115].

H.1.2 Data Preparation

We select 5 static scenes with a driving distance of $\sim$ 50 meters each for evaluating NVS at a 50% drop rate. We select one frame every $\sim$ 0.8 meters driving distance (corresponding to the overall average distance between frames) to avoid redundancy when the vehicle is slow. We release 50% of the frames for training and retain 50% for evaluation.

15. http://www.open3d.org/docs/0.12.0/python_api/open3d.geometry.PointCloud.html
16. https://github.com/dvlab-research/PointGroup

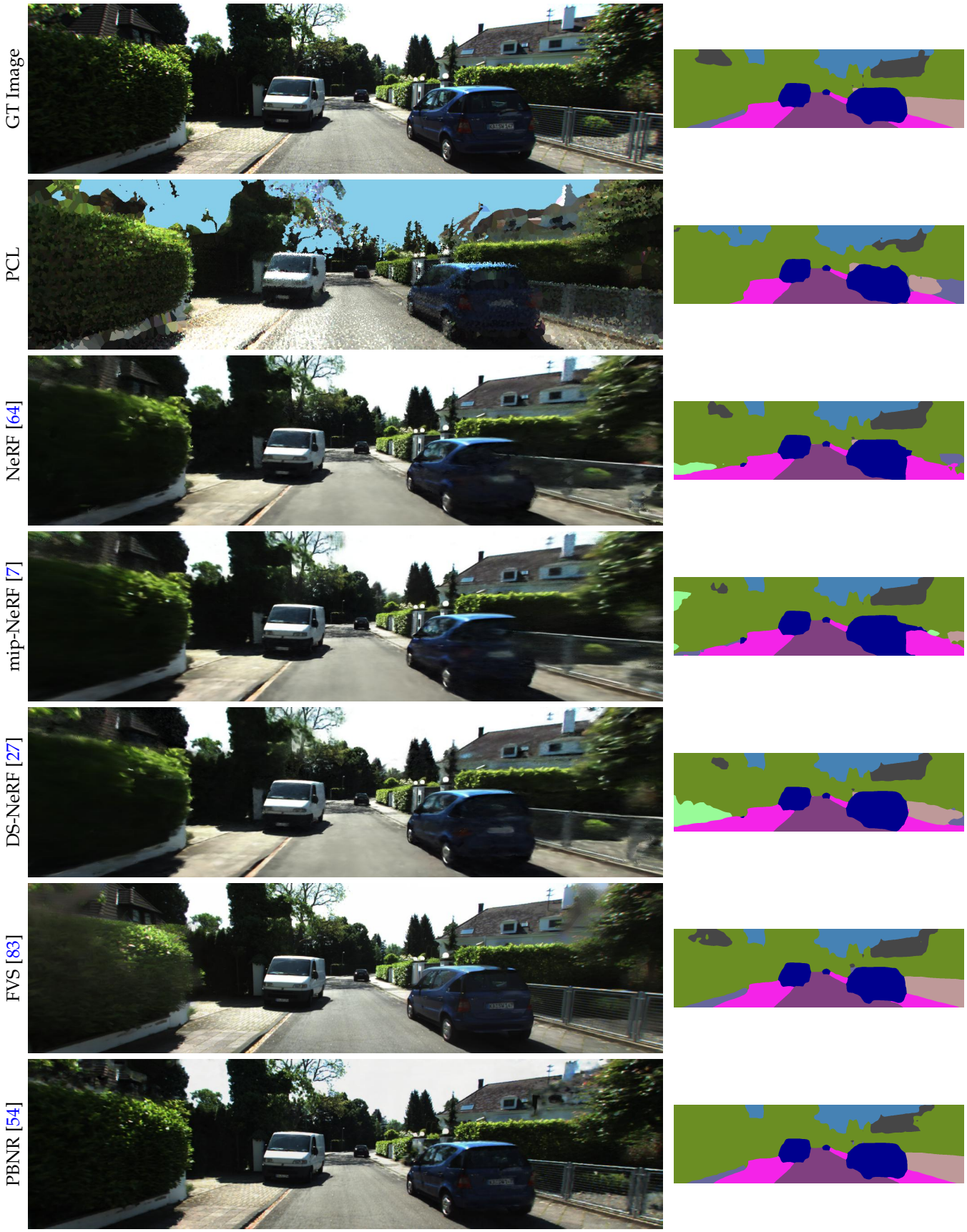


Fig. 26: **Additional Qualitative Results for Novel View Appearance & Semantic Synthesis.** The left column row shows the GT image and novel view appearance synthesis results. The right column shows the corresponding semantic segmentation using PSPNet [116].

Moreover, we select 10 static scenes with a driving distance of ~ 50 meters each for evaluating NVS at a 90% drop rate. On average, we select one frame every 4 meters driving distance in this setting. We release 50% of the frames for training and retain 50% for evaluation.

H.1.3 Baselines

We evaluate two sets of baselines for this task. The first baseline (PCL) takes a colored point cloud as input. We project non-occluded points to the test viewpoint and interpolate the missing values to obtain the full image. To determine non-occluded points, we reconstruct a mesh using the ball-pivoting method [10] on the accumulated point cloud. As there is no point in the sky region, we in-paint the sky using a constant blue color. The sky region is heuristically determined based on the projected 3D points, i.e., a large connected area in the upper half of the image without any 3D projections is considered as the sky.

The second set of baselines takes a set of images as input. For all NeRF-based methods [7], [27], [64], we train one model on each scene individually, using cascaded sampling with 256 coarse samples and 256 fine samples. We adopt the PyTorch reimplementation of NeRF [17], the original implementation of mip-NeRF [18] and DS-NeRF [19]. As for Free View Synthesis (FVS) [83], we follow its original implementation[20] and use their released model trained on the Tanks and Temples dataset [53] which generalizes well. We follow the original implementation[21] of PBNR [54] that optimizes a set of attributes such as reprojected features or depth in each input view.

H.1.4 Additional Results

We show additional qualitative results of all methods in Fig. 26 (left). The PCL baseline exhibits blocky artifacts due to interpolation. The vanilla NeRF shows promising performance but sometimes struggles due to the sparse input views. While mip-NeRF and DS-NeRF both improve the performance, the thin structures (e.g., fence) are still not well recovered. Interestingly, FVS and PBNR are better at preserving the fine details (e.g., license plate) but have lower PSNR. This could be due to small misalignments in the image space.

H.2 Benchmark of Novel View Semantic Synthesis

H.2.1 Evaluation Metric

We evaluate the confidence weighted mIoU using Eq. 10. Similar to the 2D semantic segmentation task, we omit "Train" and "Bus" during evaluation. We additionally omit "Truck", "Person"', "Rider", "Bicycle" and "Traffic Light" as these classes do not appear in the 5 static scenes for evaluating NVS.

17. https://github.com/yenchenlin/nerf-pytorch
18. https://github.com/google/mipnerf
19. https://github.com/dunbar12138/DSNeRF
20. https://github.com/isl-org/FreeViewSynthesis
21. https://gitlab.inria.fr/sibr/projects/pointbased_neural_rendering

H.2.2 Baselines

As there is no existing research work on this new benchmark, we directly apply PSPNet used in the 2D semantic segmentation task to synthesized images for semantic label prediction.

H.2.3 Additional Results

We show confidence weighted IoU on individual classes in Table 13. Note that this naïve baseline leads to significantly degraded performance on most of the classes. As shown in Fig. 26 (right), small changes in the image space sometimes lead to sigficant changes in the semantic prediction.

APPENDIX I
SEMANTIC SLAM BENCHMARK

I.1 Localization

I.1.1 Evaluation Metric

We adopt the standard Absolute Pose Error (APE) and Relative Pose Error (RPE) [37] as metrics for evaluating pose estimation. We align the predicted trajectory to the ground truth using a rigid transformation to evaluate the APE [98]. The RPE is evaluated between two frames with a distance of 1 meter.

I.1.2 Baselines

We evaluate ORB-SLAM2 [66][22] and SUMA++ [23][23] using their official implementations as baselines.

I.1.3 Additional Results

Fig. 27 shows qualitative comparison of predicted trajectories. As can be seen, both methods achieve reasonable performance while SUMA++ has a larger maximum error than ORB-SLAM2.

I.2 Geometric & Semantic Mapping

I.2.1 Evaluation Metric

We adopt the same evaluation metrics considered in the semantic scene completion benchmark, as introduced in Appendix G.6. When evaluating the quality of reconstruction, we compare ground truth and estimated reconstruction in local windows to minimize the impact of pose drifts. Specifically, we divide the test sequences into a set of local windows, each consisting of 50 consecutive frames. We first crop the ground truth and the reconstructed point cloud wrt. the region of interest of each window. These two local point clouds are then aligned using the similarity transformation between the corresponding poses [98] and compared afterwards. Finally, we average the completeness, accuracy, and mIoU metrics over the entire test sequence. Following [89], we measure completeness and accuracy over discretized voxels such that these metrics are insensitive to the density of the point clouds.

22. https://github.com/raulmur/ORB_SLAM2
23. https://github.com/PRBonn/semantic_suma

Method	Road	Sdwlk	Bldg	Wall	Fence	Pole	Trsgn	Vegt	Terr	Sky	Car	Motor	$mIoU_{class}$
Original	95.6	85.6	75.2	57.4	55.0	42.7	19.8	93.2	91.8	93.4	95.3	58.5	72.0
PCL	92.0	70.4	39.4	29.7	15.3	2.2	1.5	67.2	81.7	42.1	31.3	0.0	39.4
NeRF [64]	93.2	74.2	58.4	35.1	15.2	16.6	13.8	83.8	64.9	91.6	88.1	1.2	53.0
mip-NeRF [7]	93.0	72.9	55.9	30.4	14.8	12.6	9.5	82.0	63.2	91.5	87.9	0.0	51.2
DS-NeRF [27]	93.2	75.4	56.5	35.4	19.1	26.5	16.6	83.5	63.9	91.5	89.2	6.5	54.8
FVS [83]	95.5	83.2	65.0	54.2	44.0	17.8	33.8	90.5	87.3	91.0	92.6	50.2	67.1
PBNR [54]	95.3	82.4	67.1	46.6	44.4	28.8	18.9	90.0	87.7	89.1	88.5	42.2	65.1

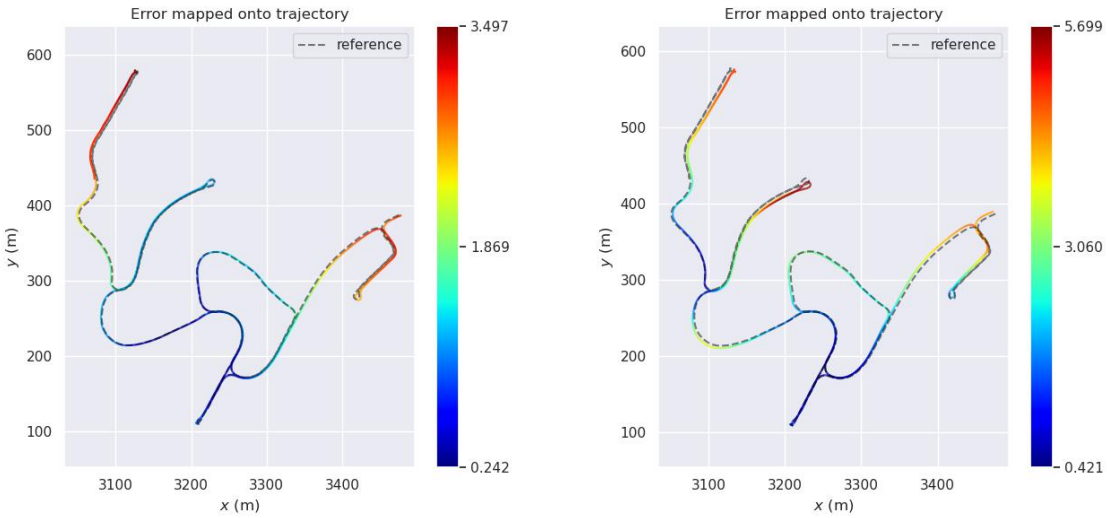
TABLE 13: **Additional Quantitative Results for Novel View Semantic Synthesis.**

I.2.2 Baselines

To obtain dense semantic reconstruction given the localization results of ORB-SLAM2, we unproject 2D semantic segmentation maps obtained from PSPNet [116] using depth maps estimated by semi-global matching (SGM) [42]. We merge the unprojected 3D points using the poses predicted by ORB-SLAM2. As for SUMA++, we use the semantic estimation model pre-trained on KITTI.

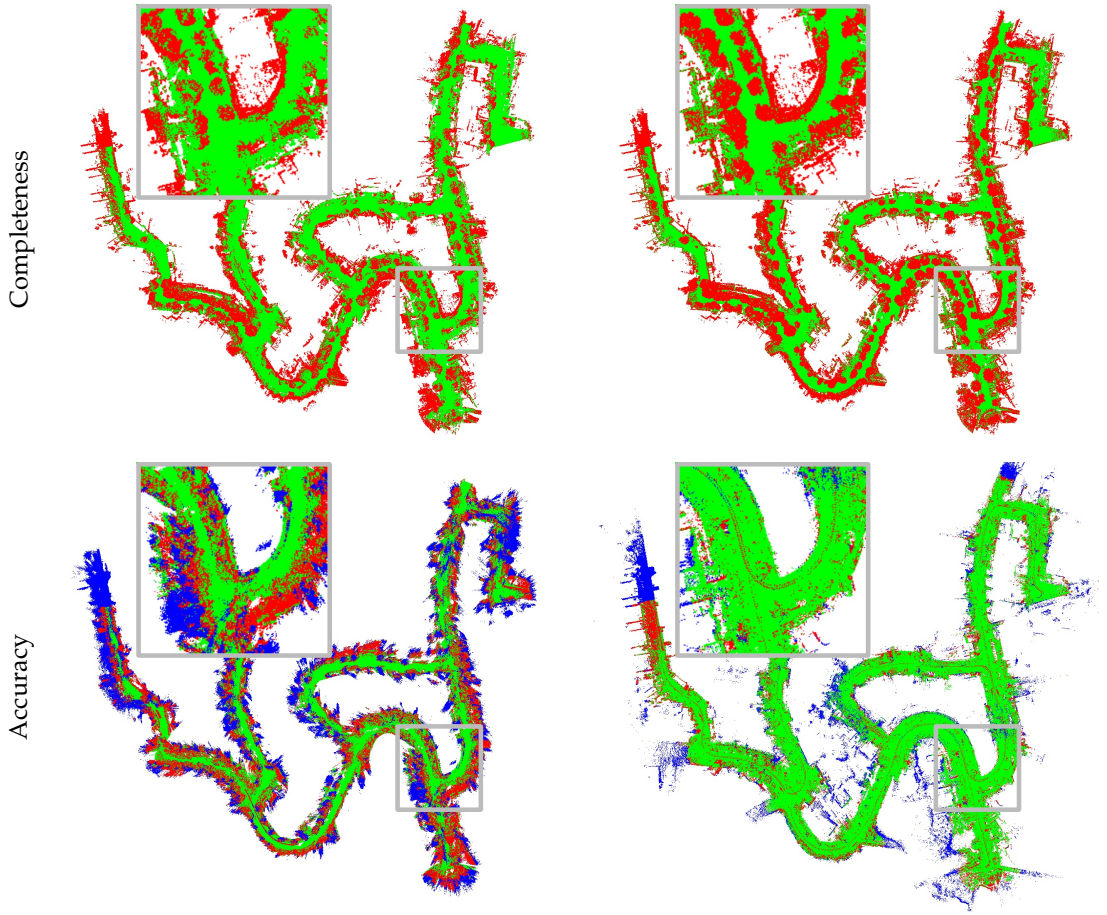
I.2.3 Additional Results

We show additional qualitative results of the geometric mapping in Fig. 28 in terms of completeness and accuracy at the threshold of 10cm. Consistent with the quantitative results, SUMA++ is more accurate while ORB-SLAM2+SGM is more complete.



(a) ORB-SLAM2 (b) SUMA++

Fig. 27: **Qualitative Results on Localization.** Each figure compares the predicted trajectory to the ground truth. Color indicates the APE in meters.



(a) ORB-SLAM2 (b) SUMA ++

Fig. 28: **Qualitative Results on Localization and Mapping.** Completeness and accuracy evaluated at a threshold of 10cm. The first row shows the ground truth point cloud with green denoting complete and red for incomplete. The second row shows the predicted point cloud with green denoting accurate, red for inaccurate, and blue for points in unobserved regions. Note that SUMA++ is more accurate while ORB-SLAM2+SGM is more complete.