The ApolloScape Open Dataset for Autonomous Driving and Its Application

Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang

Abstract—Autonomous driving has attracted tremendous attention especially in the past few years. The key techniques for a self-driving car include solving tasks like 3D map construction, self-localization, parsing the driving road and understanding objects, which enable vehicles to reason and act. However, large scale data set for training and system evaluation is still a bottleneck for developing robust perception models. In this paper, we present the *ApolloScape* dataset [1] and its applications for autonomous driving. Compared with existing public datasets from real scenes, e.g., KITTI [2] or Cityscapes [3], ApolloScape contains much large and richer labelling including holistic semantic dense point cloud for each site, stereo, per-pixel semantic labelling, lanemark labelling, instance segmentation, 3D car instance, high accurate location for every frame in various driving videos from multiple sites, cities and daytimes. For each task, it contains at lease 15x larger amount of images than SOTA datasets. To label such a complete dataset, we develop various tools and algorithms specified for each task to accelerate the labelling process, such as joint 3D-2D segment labeling, active labelling in videos etc. Depend on *ApolloScape*, we are able to develop algorithms jointly consider the learning and inference of multiple tasks. In this paper, we provide a sensor fusion scheme integrating camera videos, consumer-grade motion sensors (GPS/IMU), and a 3D semantic map in order to achieve robust self-localization and semantic segmentation for autonomous driving. We show that practically, sensor fusion and joint learning of multiple tasks are beneficial to achieve a more robust and accurate system. We expect our dataset and proposed relevant algorithms can support and motivate researchers for further development of multi-sensor fusion and multi-task learning in the field of computer vision.

Index Terms—Autonomous driving, large-scale datasets, scene/lane parsing, self localization, 3D understanding

1 Introduction

A successful self-driving vehicle that is widely applied must include three essential components. Firstly, understanding the environment, where commonly a 3D semantic HD map at the back-end precisely recorded the environment. Secondly, understanding self-location, where an on-the-fly self-localization system puts the vehicles accurately inside the 3D world, so that it can plot a path to every target location. Thirdly, understanding semantics in the view, where a 3D perceptual system detects other moving objects, guidance signs and obstacles on the road, in order to avoid collisions and perform correct actions. The prevailing approaches for solving those tasks from self-driving companies are mostly dependent on LIDAR [4], whereas vision-based approaches, which have potentially very low-cost, are still very challenging and under research. It requires solving tasks such as learning to do visual 3D scene reconstruction [5], [6], [7], [8], self-localization [9], [10], semantic parsing [11], [12], semantic instance understanding [13], [14], object 3D instance understanding [15], [16], [17], [18], [19] online in a self-driving video etc. However, the SOTA datasets for supporting these tasks either have limited amount, e.g., KITTI [2] only has 200 training images for semantic understanding, or limited variation of tasks, e.g., Cityscapes [3] only has discrete semantic labelled frames without tasks like localization or 3D reconstruction. Therefore, in order to have a holistic training and evaluation of a vision-based self-driving system, in this paper, we build the *Apolloscape* [1] for autonomous driving, which is a growing and unified dataset extending previous ones both on the data scale, label density and variation of tasks.

Specifically, in current stage, ApolloScape contains properties of,

1) dense semantics 3D point cloud for the environment (20+ driving site)
2) stereo driving videos (100+ hours)
3) high accurate 6DoF camera pose. (translation ≤ 50 mm, rotation $\leq 0.015°$)
4) videos at same site under different day times, (morning, noon, night)
5) dense per-pixel per-frame semantic labelling (35 classes, 144K+ images)
6) per-pixel lanemark labelling (35 classes, 160K+ images)
7) semantic 2D instances segmentation (8 classes, 90K+ images)
8) 2D car keypoints and 3D car instance labelling (70K cars)

• *The authors are with Baidu Research, Beijing 100085, China. E-mail: {huangxinyu01, wangpeng54, chengxinjing, zhoudingfu, gengqichuan}@baidu.com, ryang@cs.uky.edu.*

Manuscript received 23 Sept. 2018; revised 30 May 2019; accepted 16 June 2019. Date of publication 2 July 2019; date of current version 2 Sept. 2020. (Corresponding author: Peng Wang).
Recommended for acceptance by M. Bennamoun and Y. Guo.
Digital Object Identifier no. 10.1109/TPAMI.2019.2926463

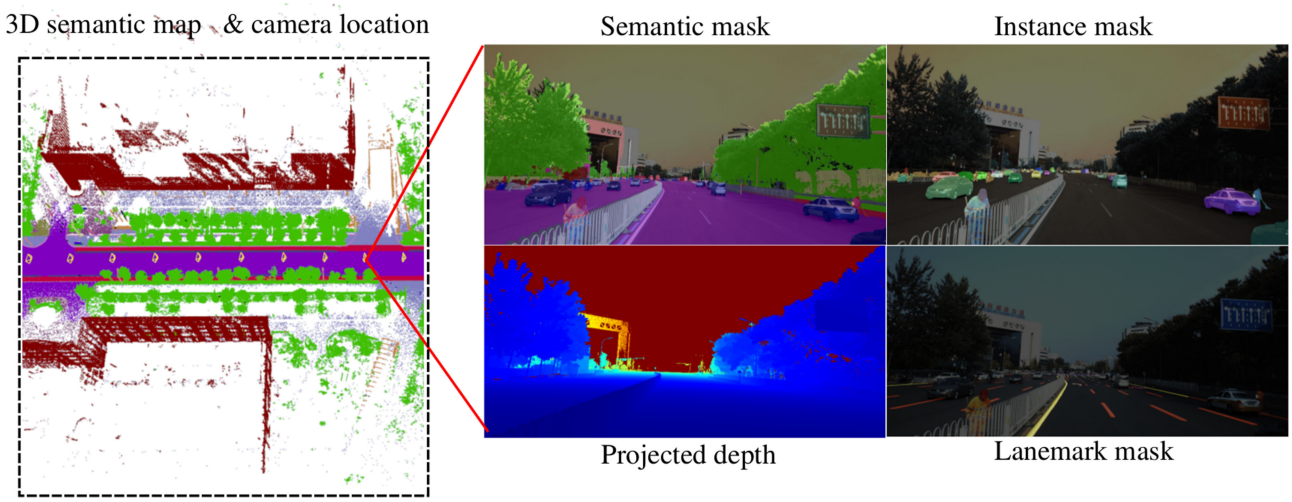


Fig. 1. A glance of ApolloScape with various properties. The images are cropped for better visualization.

With these information, we have released several standard benchmarks for scene parsing [20], instance segmentation [21], lanemark parsing [22], self-localization [23] by withholding part of the data as test set, and our toolkit for visualization and evaluation has also published [24]. Here, for 3D car instance, we list the car number we already labelled, and since it is still under development, we will elaborate it in our future work. Fig. 1 shows a glance of *ApolloScape*, which illustrates various information from the dataset that is necessary for autonomous driving. Our dataset is still growing and evolving, and will shortly contains new tasks such as 3D car instance shape and pose, 3D car tracking etc., which are important for scene understanding with finer granularity. In addition, thanks to our efficient labelling pipeline, we are able to scale the dataset to multiple cities and sites, and we have already contained 10 cities in China under various driving conditions.

Based on *ApolloScape*, we are able to develop algorithms for jointly considering 3D and 2D simultaneously with multiple tasks like segmentation, reconstruction, self-localization etc. These tasks are traditionally handled individually [9], [12], or jointly handled offline with semantic SLAM [25] which could be time consuming. However, from a more practical standpoint, self-driving car needs to handle localization and parsing the environment on-the-fly efficiently. Therefore, in this paper, we propose a deep learning based online algorithm jointly solving localization and semantic scene parsing when a 3D semantic map is available. In our system, we assume to have (a) GPS/IMU signal to provide a coarse camera pose estimation; (b) a semantic 3D map for the static environment. The GPS/IMU signals serve as a crucial prior for our pose estimation system. The semantic 3D map, which can synthesize a semantic view for a given camera pose, not only provides strong guidance for scene parsing, but also helps maintain temporal consistency.

With our framework, the camera poses and scene semantics are mutually beneficial. The camera poses help establish the correspondences between the 3D semantic map and 2D semantic label map. Conversely, scene semantics could help refine camera poses. Our unified framework yields better results, in terms of both accuracy and speed, for both tasks than doing them individually. In our experiments, using a single Titan Z GPU, the networks in our system estimates the pose in 10ms with accuracy under 1 degree, and segments the image 512×608 within 90 ms with pixel accuracy around 96 percent without model compression, which demonstrates its efficiency and effectiveness.

In summary, the contributions of this work are in three folds,

1) We propose a large and rich dataset, named as *ApolloScape*, which includes various tasks, e.g., 3D reconstruction, self-localization, semantic parsing, instance segmentation etc., supporting the training and evaluation of vision-based autonomous driving algorithms and systems.
2) For developing the dataset, we design an efficient and scalable 2D/3D joint-labelling pipeline, where various tools are developed for 2D segmentation, 3D instance understanding etc. For example, compared with fully manual labelling, our 3D/2D labelling pipeline saves 70 percent labeling time for semantic segmentation.
3) Based on *ApolloScape*, we developed a deep learning based joint self-localization and segmentation algorithm, which is relying on a semantic 3D map. The system fuses sensors from camera and customer-grad GPS/IMU, which runs efficiently and improves the robustness and accuracy for camera localization and scene parsing.

The structure of this paper is organized as follows. We provide related work in Section 2, and elaborate the collection and labelling of *ApolloScape* in Section 3. In Section 4, we explain the developed efficient joint segmentation and localization algorithm. Finally, we present the evaluation results of our algorithms, the benchmarks for multiple tasks and corresponding baseline algorithms performed on these tasks in Section 5.

2 Related Works

Autonomous driving datasets and related algorithms has been an active research area for years. Here we summarize the related works in aspects of datasets and most relevant

TABLE 1
Comparison between Our Dataset and the Other Street-View Self-Driving Datasets Published

Dataset	Real	Location Accuracy	Diversity	Annotation			
				3D	2D	Video	Lane
CamVid [26]	✓	-	day time	no	pixel: 701	✓	2D / 2 classes
Kitti [2]	✓	cm	day time	80k 3D box	box: 15k pixel: 400	-	no
Cityscapes [3]	✓	-	day time 50 cities	no	pixel: 25k	-	no
Toronto [27]	✓	cm	Toronto	focus on buildings and roads exact numbers are not available[1]			
Mapillary [28]	✓	meter	various weather day & night 6 continents	no	pixel: 25k	-	2D / 2 classes
BDD100K [29]	✓	meter	various weather day 4 regions in US	no	box: 100k pixel: 10k	-	2D / 2 classes
SYNTHIA [30]	-	-	various weather	box	pixel:213k	✓	no
P.F.B. [31]	-	-	various weather	box	pixel:250k	✓	no
ApolloScape	✓	cm	various weather day time 4 regions in China	3D semantic point 70K 3D fitted cars	pixel: 140k	✓	3D / 2D Video 27 classes

1. *database is not open to public yet.*
"pixel" represents 2D pixel-level annotations. "point" represents 3D point-level annotations. "box" represents bounding box-level annotations. "Video" indicates whether 2D video sequences are annotated. "3D fitted cars" gives the number of car instance we already fitted in the images with a 3D mesh model, which we will introduce in our future works.

algorithms without enumerating them all due to space limitation.

2.1 Datasets for Autonomous Driving

Most recently, various datasets targeting at solving each individual visual task for robot navigation have been released such as 3D geometry estimation [32], [33], localization [9], [34], instance detection and segmentation [35], [36]. However, focusing on autonomous driving, a set of comprehensive visual tasks are preferred to be collected consistently within a unified dataset from driving videos, so that one may explore the mutual benefits between different problems.

In past years, lots of datasets have been collected in various cities, aiming to increase variability and complexity of urban street views for self-driving applications. The Cambridge-driving Labeled Video database (CamVid) [26] is the first dataset with semantic annotated videos. The size of the dataset is small, containing 701 manually annotated images with 32 semantic classes. The KITTI vision benchmark suite [2] is later collected and contains multiple computer vision tasks such as stereo, optical flow, 2D/3D object detection and tracking. For semantics, it mainly focuses on detection, where 7,481 training and 7,518 test images are annotated by 2D and 3D bounding boxes, and each image contains up to 15 cars and 30 pedestrians. Nevertheless, for segmentation, very few images contain pixel-level annotations, yielding a relatively weak benchmark for semantic segmentation. Most recently, the Cityscapes dataset [3] is specially collected for 2D segmentation which contains 30 semantic classes. In detail, 5,000 images have detailed annotations, and 20,000 images have coarse annotations. Although video frames are available, only one image out of each video is manually labelled. Thus, tasks such as video segmentation can not be performed. Similarly, the Mapillary Vistas dataset [28] provides a larger set of images with fine annotations, which has 25,000 images with 66 object categories. The TorontoCity benchmark [27] collects LIDAR data and images including stereo and panoramas from both drones and moving vehicles. Although the dataset scale is large, which covers the Toronto area. as mentioned by authors, it is not possible to manually do per-pixel labelling of each frame. Therefore, only two semantic classes, i.e., building footprints and roads, are provided for benchmarks of segmentation. BDD100K database [29] contains 100K raw video sequences representing more than 1000 hours of driving hours with more than 100 million images. Similarly with the Cityscapes, one image is selected from each video clip for annotation. 100K images are annotated in bounding box level and 10K images are annotated in pixel level.

Real data collection is laborious, to avoid the difficulties in real scene collection, several synthetic datasets are also proposes. SYNTHIA [30] builds a virtual city with Unity development platform [37], and Play for benchmark [31] extracts ground truth with GTA game engine. Though large amount of data and ground truth can be generated, there is still a domain gap [38] between appearance of synthesized images and the real ones. In general, models learned in real scenario still generalize better in real applications such as object detection and segmentation [39], [40].

In Table 1, we compare the properties our dataset and other SOTA datasets for autonomous driving, and show that *ApolloScape* is unique in terms of data scale, granularity of labelling, task variations within real environments. Later in Section 3, we will present more details about the dataset.

Fig. 2. Acquisition system consists of two laser scanners, up to six video cameras, and a combined IMU/GNSS system.

2.2 Self-Localization and Semantic Scene Parsing

As discussed in Section 1, we also try to tackle real-time self-localization and semantic scene parsing back on *ApolloScape* given a video or a single image. These two problems have long been center focus for computer vision. Here we summarize the related works on outdoor cases with street-view images as input.

Visual Self-Localization. Traditionally, localizing an image given a set of 3D points is formulated as a Perspective-n-Point (PnP) problem [41], [42] by matching feature points in 2D and features in 3D through cardinality maximization. Usually in a large environment, a pose prior is required in order to obtain good estimation [43], [44]. Campbell et al. [45] propose a global-optimal solver which leverage the prior. In the case that geo-tagged images are available, Sattler et al. [46] propose to use image-retrieval to avoid matching large-scale point cloud. When given a video, temporal information could be further modeled with methods like SLAM [47] etc, which increases the localization accuracy and speed.

Although these methods are effective in cases with distinguished feature points, they are still not practical for city-scale environment with billions of points, and they may also fail in areas with low texture, repeated structures, and occlusions. Thus, recently, deep learned features with hierarchical representations are proposed for localization. PoseNet [9], [48] takes a low-resolution image as input, which can estimate pose in 10 ms w.r.t. a feature rich environment composed of distinguished landmarks. LSTM-PoseNet [49] further captures a global spatial context after CNN features. Given an video, later works incorporate Bi-Directional LSTM [50] or Kalman filter LSTM [51] to obtain better results with temporal information. Most recently, many works [10], [52] also consider adding semantic cues as more robust representation for localization. However, in street-view scenario, considering a road with trees aside, in most cases, no significant landmark appears, which could fail the visual models. Thus, signals from GPS/IMU are a must-have for robust localization in these cases [53], whereas the problem switched to estimating the relative pose between the camera view from a noisy pose and the real pose. For finding relative camera pose of two views, recently, researchers [54], [55] propose to stack the two images as a network input. In our case, we concatenate the real image with an online rendered label map from the noisy pose, which provides superior results in our experiments.

Street Scene Parsing. For parsing a single image of street views (e.g., these from CityScapes [3]), most state-of-the-arts (SOTA) algorithms are designed based on a FCN [11] and a multi-scale context module with dilated convolution [12], pooling [56], CRF [57], or spatial RNN [58]. However, they are dependent on a ResNet [59] with hundreds of layers, which is too computationally expensive for applications that require real-time performance. Some researchers apply small models [60] or model compression [61] for acceleration, with the cost of reduced accuracy. When the input is a video, spatial-temporal informations are jointly considered, Kundu et al. [62] use 3D dense CRF to get temporally consistent results. Recently, optical flow [63] between consecutive frames is computed to transfer label or features [64], [65] from the previous frame to current one. In our case, we connect consecutive video frames through 3D information and camera poses, which is a more compact representation for static background. In our case, we propose the projection from 3D maps as an additional input, which alleviates the difficulty of scene parsing solely from image cues. Additionally, we adopt a light weighted network from DeMoN [55] for inference efficiency.

Joint 2D-3D for Video Parsing. Our work is also related to joint reconstruction, pose estimation and parsing [25], [66] through embedding 2D-3D consistency. Traditionally, reliant on structure-from-motion (SFM) [66] from feature or photometric matching, those methods first reconstruct a 3D map, and then perform semantic parsing over 2D and 3D jointly, yielding geometrically consistent segmentation between multiple frames. Most recently, CNN-SLAM [67] replaces traditional 3D reconstruction module with a single image depth network, and adopts a segment network for image parsing. However, all these approaches are processed off-line and only for static background, which do not satisfy our online setting. Moreover, the quality of a reconstructed 3D model is not comparable with the one collected with a 3D scanner.

3 Build ApolloScape

In this section, we introduce our acquisition system, specifications about the collected data and efficient labelling process for building ApolloScape.

3.1 Acquisition System

In Fig. 2, we visualize our collection system. To collect static 3D environment, we adopt Riegl VMX-1HA [68] as our acquisition system that consists of two VUX-1HA laser scanners (360 degree FOV, range from 1.2m up to 420m with target reflectivity larger than 80 percent), one VMX-CS6 camera system (two front cameras are used with resolution 3384×2710), and a measuring head with IMU/GNSS (position accuracy $20 \sim 50$ mm, roll & pitch accuracy 0.005 degree, and heading accuracy $0.015°$). The laser scanners utilizes two laser beams to scan its surroundings vertically that are similar to the push-broom cameras. Comparing with common-used Velodyne HDL-64E [4], the scanners are able to acquire higher density of point clouds and obtain

TABLE 2
Total and Average Number of Instances in KITTI [2], Cityscapes [3], BDD100K [29], and *ApolloScape* (Instance-Level)

Count	Kitti (box)	Cityscapes (pixel)	BDD100K (box)	ApolloScape (pixel)		
total ($\times 10^4$)						
person	0.6	2.4	12.9	**54.3**		
vehicle	3.0	4.1	110.2	**198.9**		
average per image				**e**	**m**	**h**
person	0.8	7.0	1.3	1.1	6.2	**16.9**
vehicle	4.1	11.8	11.0	12.7	24.0	**38.1**

"pixel" Represents 2D Pixel-Level Annotations. "box" represents bounding box-level annotations. The letters, e, m, and h, indicate easy, moderate, and hard subsets in ApolloScape *respectively.*

higher measuring accuracy / precision (5 mm / 3 mm). The whole system has been internally calibrated and synchronized, and is mounted on the top of a mid-size SUV.

Additionally, the system contains two high frontal camera capturing with a resolution of 3384×2710, and is well calibrated with the LIDAR device. Finally, to obtain high accurate GPS/IMU information, a temporary GPS basement is set up near the collection site to make sure the localization of the camera is sufficiently accurate for us to match the 2D image and 3D point cloud. Commonly, our vehicle drives at the speed of 30 km per hour and the cameras are triggered once every meter, i.e., 30fps.

3.2 Specifications

Here, based on the acquisition system, we first present the specifications of *Apolloscape* w.r.t. different tasks, e.g., predefined semantic classes, lanemark classes and instance etc., to allow better overview of the dataset. In Section 3.3, we will introduce our active labelling pipeline which allows us to efficiently produce the ground truth of multiple tasks simultaneously.

Semantic Scene Parsing. In our current version released online [20], [21], we have 143,906 video frames and their corresponding pixel-level semantic labelling, from which 89,430 images contain instance-level annotations where movable objects are further separated. Notice that our labelled images contains temporal information which could also be useful for video semantic and object segmentation.

To make the evaluation more comprehensive, similar to the KITTI [2], we separate the recorded video with the level of easy, moderate, and heavy scene complexities based on the amount of movable objects, such as person and vehicles. Table 2 compares the scene complexities between *ApolloScape*, the Cityscapes [3] and KITTI [2], where we show the statistics for each individual classes of movable objects. *ApolloScape* contains more objects than others in terms of both total number and average number of object instances from images. More importantly, it contains stronger challenging environments, as shown in Fig. 3. For instance, high contrast regions due to sun light and large area of shadows from the overpass. Mirror reflections of multiple nearby vehicles on a bus glass due to highly crowded transportation. We hope these case can help and motivate researchers to develop more robust models against environment changes.

For semantic scene parsing, we annotate 24 different labels in four groups. The specifications of the classes are partially borrowed from the Cityscapes dataset. Fig. 4 gives the amount of labelled pixels for each class. As expected, *ApolloScape* provides much higher amount of average annotated pixels than Cityscapes, especially for some rare classes, e.g., *traffic-light*, *pole*. Here, we add several new classes common in China. For instance, we add "tricycle" that is one of the most popular means of transportation. This class covers all kinds of three-wheeled vehicles that could be both motorized and human-powered. The rider class in the Cityscape is defined as the person on means of transportation. Here, we consider the person and the means of transportation as a single moving object, and treat the two together as one class. The three classes related to rider, i.e., bicycle, motorcycle, and tricycle, represent means of transportation without rider and parked along the roads.

Semantic Lanemark Segmentation. Automatically understanding lane mark is perhaps the most important function for autonomous driving since it is the guidance for possible actions. In *ApolloScape*, 27 different lane markings are used for evaluation as elaborated in Table 3 and Fig. 5. The labels are defined based on lane mark attributes including color (e.g., white and yellow) and type (e.g., solid and broken). To be specific, 165949 images from 3 road sites are labelled and released online [22], where 33760 images are withheld for

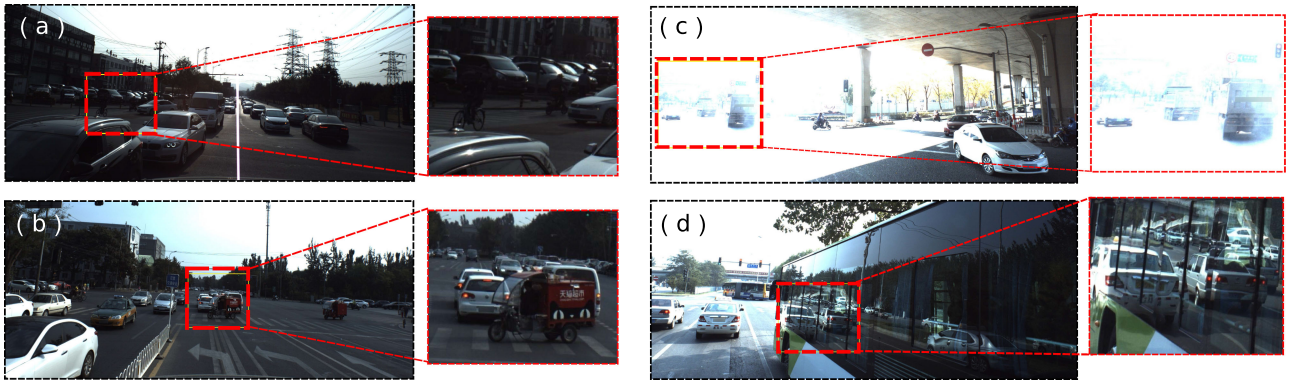


Fig. 3. Examples with challenging environments for object detection and segmentation (Images are center-cropped for better visualization). We highlight and zoom in the region of challenges in each image. (a) Objects with heavy occlusion and small scale. (b) Abnormal action by cyclist drivers. (c) High contrast and overexposure due to shadows and strong sunlight. (d) Mirror reflection on bus glasses.

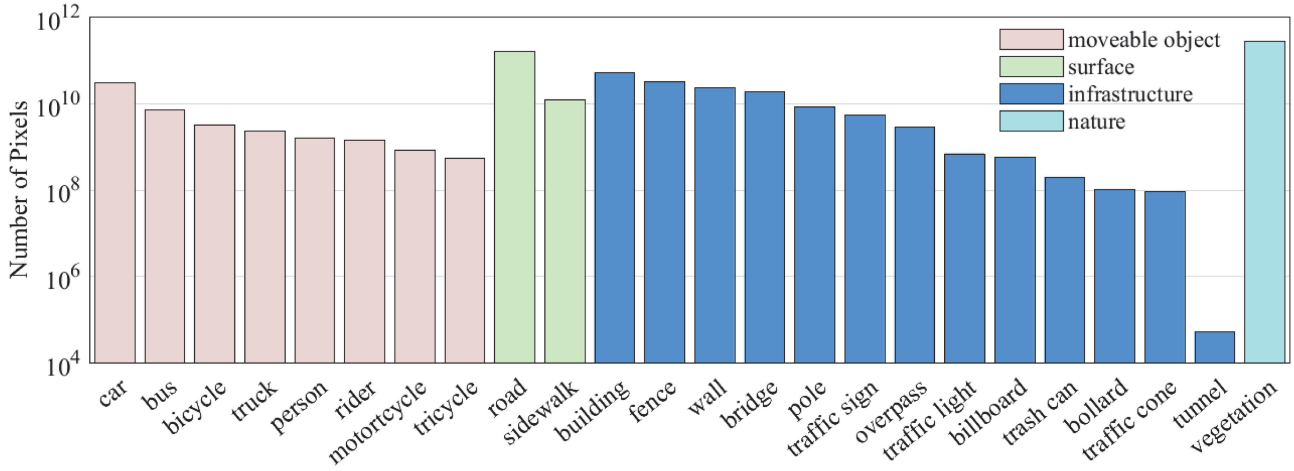


Fig. 4. 24 semantic classes and corresponding numbers of annotated pixels. Bar colors indicate different semantic groups.

testing. Comparing to other public available datasets such as KITTI [2] or the one from Tusimple [69], *ApolloScape* is the first large dataset containing rich semantic labelling for lane marks with many variations.

Self-Localization. Each frame of our recorded video is tagged with high accurate GPS/IMU signal automatically. Therefore, the videos we released for segmentation are also available for self-localization research. However, for setting up a benchmark, we additionally collected a much larger amount of videos, which has not been semantically labelled. Specifically, videos for localization, as published at [23], contain 6 more roads at 4 different cities, which include roughly 300k images, and road of 28 km.

TABLE 3
Details of Lane Mark Labels in Our Dataset (y: yellow, w:white)

Type	Color	Use
solid	w	dividing
solid	y	dividing
double solid	w	dividing, no pass
double solid	y	dividing, no pass
solid & broken	y	dividing, one-way pass
solid & broken	w	dividing, one-way pass
broken	w	guiding
broken	y	guiding
double broken	y	guiding
solid	w	stopping
solid	w	chevron
solid	y	chevron
solid	w	parking
crosswalk	w	zebra
arrow	w	u-turn
arrow	w	thru
arrow	w	thru & left turn
arrow	w	thru & right turn
arrow	w	left turn
arrow	w	right turn
arrow	w	left & right turn
arrow	w	left & u-turn
bump	n/a	speed reduction
diamond	w/y	zebra attention
rectangle	w/y	no parking
visible old marking	y/w	others
other markings	n/a	others

Our dataset has variations under different lighting, i.e., morning, noon and night, and driving conditions, i.e., rush and non-rush hours, with stereo pair of images available. In addition, each road has a survey-grade point cloud based 3D map that can be used in finding matching pixels for both supervised and unsupervised feature learning [70], [71] etc. Finally, we record each road by driving from start-to-end and then end-to-start, which means each position along a road will be looked at from two opposite directions. This enables the research of camera localization with large view changes such as that proposed in semantic visual localization [10].

3.3 Labeling Process

In order to make our labeling of video frames accurate and efficient, we propose an active labelling pipeline by jointly consider 2D and 3D information, as shown in Fig. 6. The pipeline mainly consists of two stages, 3D labeling and 2D labeling, to handle static background/objects and moving objects respectively. The basic idea of our pipeline is similar to the one described in [72], which transfers the 3D labelled results to 2D images by camera projection, while we need to handle much larger amount of data and have different set up of the acquisition vehicle. Thus some key techniques used in our pipeline are re-designed, which we will elaborate later.

Moving Object Removal. As mentioned in Section 3.1, LIDAR scanner *Riegl* is accurate in static background, while due to low scanning rate, the point clouds of moving objects, such as vehicles and pedestrians running on the road, could be compressed, expanded, or completely missing in the captured point clouds as illustrated in Fig. 8b. Thus, we design to handle labelling static background and moving object separately, as shown in Fig. 6. Specifically, in the first step, we do moving object removal from our collected point clouds by 1) scan the same road segment multiple rounds; 2) align these point clouds based on manually selected control points; 3) remove the points based on the temporal consistency. Formally, the condition to kept a point $\mathbf{x}$ in round j is,

$$\sum_{i=0}^{r} \mathbb{1}(\exists\ \mathbf{x}_i \in \mathcal{P}_i\ s.t.\ \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon_d)/r \geq \delta, \tag{1}$$

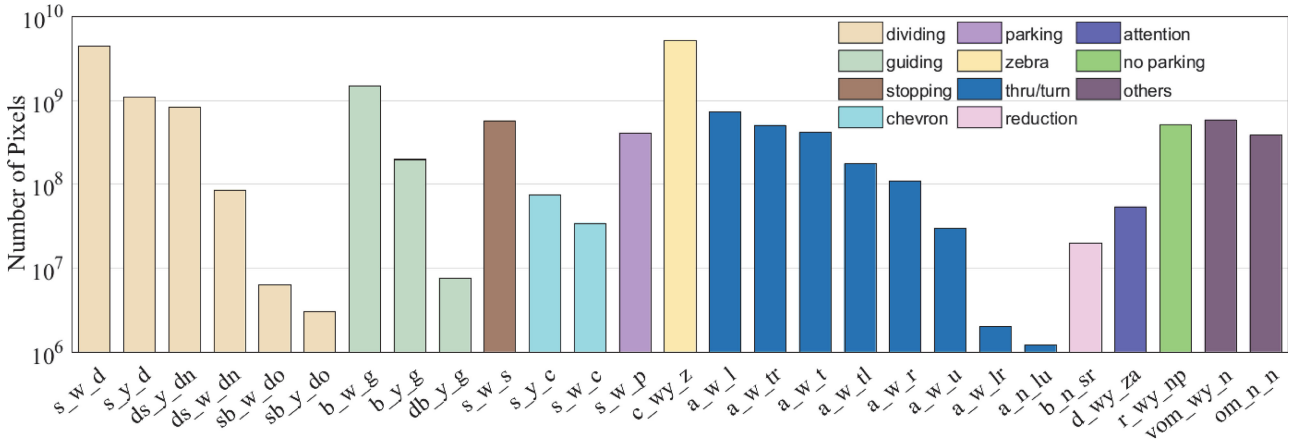


Fig. 5. 27 lane mark labels and corresponding numbers of annotated pixels. Bar colors indicate 11 different lane mark usages. Here,"s_w_d" is short for solid, white and dividing in Table 3 by combining the first letter of type, color and usage respectively, and other classes are named accordingly.

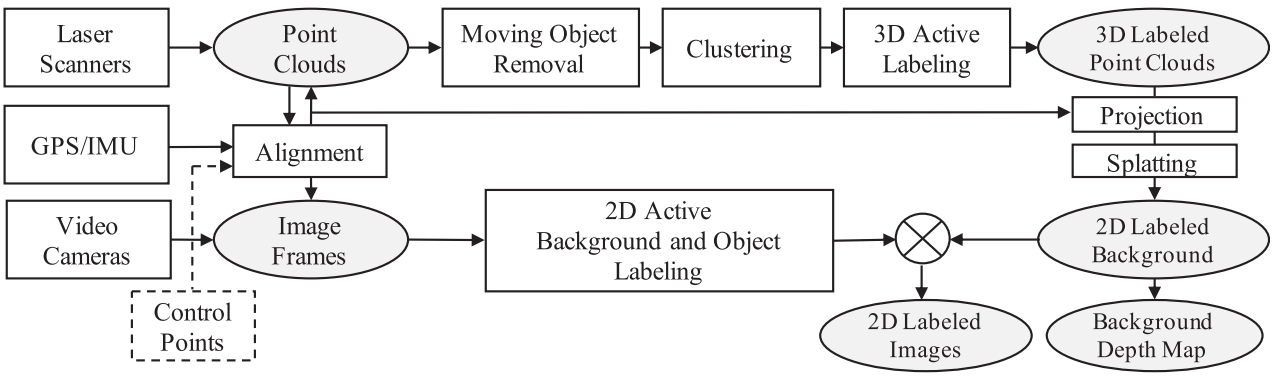


Fig. 6. Our 2D/3D labeling pipeline that label static background/objects and moving objects separately. We also adopt active strategies for accelerating the labelling process for scalability of the labelling process. For inputs, since GPS/IMU still has some errors, we manually add control points to better align the point clouds and our image frames. In Section 3.3, we present the details of each components.

where $\delta = 0.6$ and $\epsilon_d = 0.025$ m in our setting, and $\mathbb{1}()$ is an indicator function. It indicates that a 3D point will be kept if it appears with high frequency in many rounds of recording, i.e., 60 percent of all times. We keep the remained point clouds as a static background $\mathcal{M}$ for semantic labelling.

3D Labelling. Next, for labelling static background (3D Labeling), rather than label each 3D point and loading all the points, we first separate the 3D points into multiple parts, and over-segment each part of point clouds into point clusters based on spatial distances and normal directions using locally convex connected patches (LCCP) [73] implemented with PCL [74]. Then, we label these point clusters manually using our in-house developed 3D labelling tool as shown in Fig. 7, which can easily do point cloud rotation, (inverse-)selection by polygons, matching between point clouds and camera views, etc.. Notice at this stage, there will be point clouds belonging to movable but static objects such as bicycles and cars parking aside the road. These point clouds are remained in our background, and also labelled in 3D which are valuable to increase our label efficiency of objects in 2D images.

To further improve 3D point cloud labelling efficiency, after labelling of one road, we actively train a PointNet++ model [75] to pre-label the over-segmented point cloud clusters of the next road. Labellers are then asked to refine and correct the results by fixing wrong annotations, which often occur near the object boundaries. With the growing number of labelled point clouds, our learned model can label new roads with increasing accuracy, yielding accelerated labelling process, which scales up to various cities and roads.

Splatting & Projection. Once the 3D annotations are generated, the annotations of static background/objects for all the 2D image frames are generated automatically by 3D-2D projections. In our setting, the 3D map is a point cloud based environment. Although the density of the point cloud

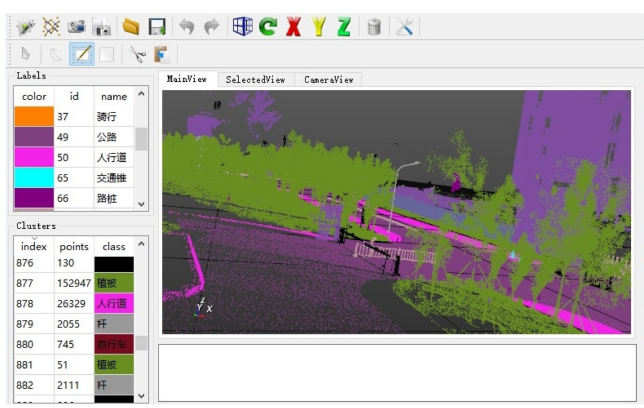


Fig. 7. The user interface of our 3D labeling tool. At left-top, we show the pre-defined color code of different classes. At left-bottom, we show the labelling logs which can be used to revert the labelling when mistakes happen. At center part, labelled point cloud is shown indicating the labelling progress.

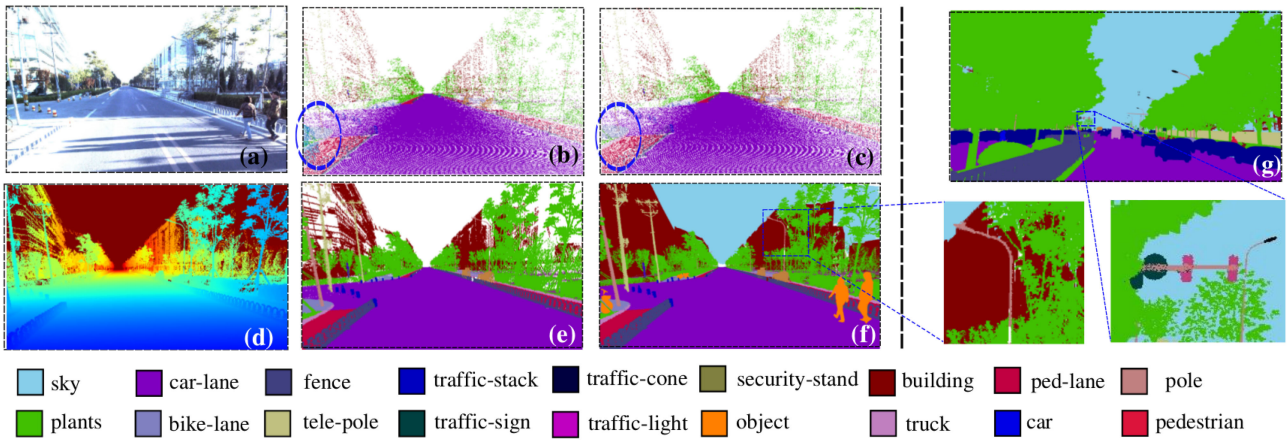


Fig. 8. A labelled example of the labelling pipeline for semantic parsing, a subset of color coded labels are shown below. (a) Image. (b) Rendered label map with 3D point cloud projection, with an inaccurate moving object (rider) circled in blue. (c) Rendered label map with 3D point cloud projection after points with low temporal consistency being removed. (d) & (e) Rendered depth map of background and rendered label map after class dependent splatting in 3D point clouds (Section 3.3). (f) Merged label map with missing region in-painted, moving objects and sky. (g) Another label map with very small traffic lights. Details are zoomed out highlighting the details of our rendered label maps. Other examples of our labeled videos is shown online [20].

is very high (one point per 25 mm within road regions), when the 3D points are far away from the camera, the projected labels could be sparse, e.g., regions of buildings shown in Fig. 8c. Thus for each point in the environment, we adopt the point splatting technique, by enlarging the 3D point to a square where the square size is determined by its semantic class.

Formally, given a 6-DOF camera pose $\mathbf{p} = [\mathbf{q}, \mathbf{t}] \in SE(3)$, where $\mathbf{q} \in SO(3)$ is the quaternion representation of rotation and $\mathbf{t} \in \mathbb{R}^3$ is translation, a label map can be rendered from the semantic 3D map, where z-buffer is applied to find the closest point at each pixel. For a 3D point $\mathbf{x}$ belonging a class c, its square size s_c is set to be proportional to the class' average distance to the camera. Formally,

$$s_c \propto \frac{1}{|\mathcal{P}_c|} \sum_{\mathbf{x} \in \mathcal{P}_c} \min_{\mathbf{t} \in \mathcal{T}} d(\mathbf{x}, \mathbf{t}), \tag{2}$$

where $\mathcal{P}_c$ is the set of 3D points belong to class c, and $\mathcal{T}$ is the set of ground truth camera poses. Then, given the relative square size between different classes, we define an absolute range to obtain the actual square size for splatting. This is non-trivial since too large size will result in dilated edges, while too small size will yield many holes. In our experiments, we set the range as $[0.025, 0.05]$, and find that it provides the highest visual quality. As shown in Fig. 8e, invalid values in-between those projected points are well in-painted, meanwhile the boundaries separating different semantic classes are also well preserved, yielding the both the background depth map and 2D labelled background. With such a strategy, we increase labelling efficiency and accuracy for video frames. For example, it could be very labor-intensive to label texture-rich regions like trees, poles and traffic lights further away, especially when occlusion happens like fence on the road as illustrated in Fig. 8g.

2D Labelling of Objects and Backgrounds. Finally, to generate the final labels (Fig. 8f), we need to label the moving objects in the environments, and fix missing parts at background like part of building regions. Similar with 3D point cloud labelling, we also developed an in-house 2D labelling tool with the same interface as 3D tool in Fig. 7. To speed up the 2D semantic labeling, we also use a labelling strategy by training a CNN network for movable objects and background [76] to pre-segment the 2D images. For segmenting background, we test with original image resolution collected by our camera, where the resolution is much higher than that used in the original paper to increase the quality of predicted region boundaries. For segment objects, similar with MaskRCNN [13], we first do 2D object detection with faster RCNN [77], and segment object masks inside. However, since we consider high requirements for object boundaries rather than class accuracy, for each bounding box with high confidence (≥ 0.9), we enlarge the bounding box and crop out the object region with context similar to [78]. Then, we upsample the cropped image to a higher resolution by setting a minimum resolution of prediction (minimum len greater than 512), and segment out the mask with an actively trained mask CNN network with the same architecture in [76]. The two networks for segmenting background and objects are updated when images in one road is labelled. Here, the learning parameters from these networks follow the original papers.

Finally, the segmented results from the networks are fused with our rendered label map from the semantic 3D point clouds following two rules: 1) for fusing segmented label map from the background network, we fill the predicted label in the pixels without 3D projection, yielding a background semantic map. 2) for fusing semantic object label segmented by object network, we pasted the object mask over the fused background map, without replacing the projected static movable object mask rendered from 3D points as mentioned in 3D labelling. We provided this fused label map for labellers to further fine tuning when error happens especially around object boundary or occlusion from the object masks. In addition, the user can omit any of the pre-segmented results from CNNs to do relabelling if the segmented results are far from satisfaction. Our label tool supports multiple actions such as polygons and pasting brushes etc., which are commonly adopted by many popular open source label tools.[1]

1. https://github.com/topics/labeling-tool

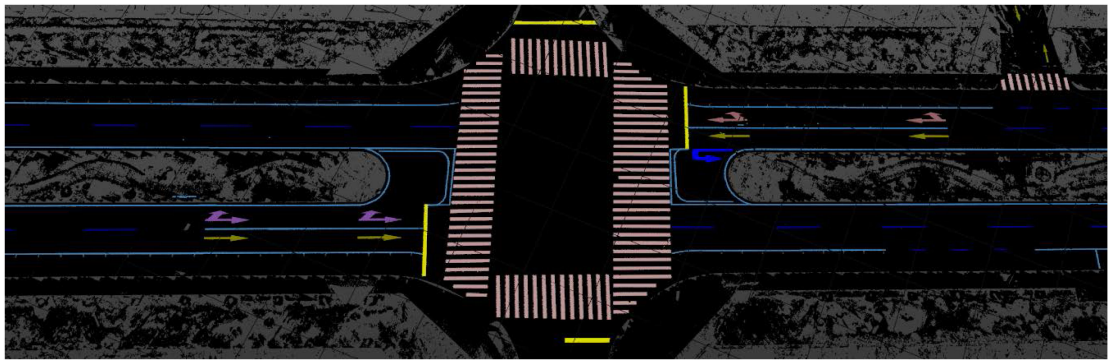


Fig. 9. Bird view of our projected road lane marks with labelling.

A final labelled example is shown in Fig. 8f & 8g. Notice that some background classes such as fence, traffic light, and vegetation are annotated in details using our projection and missing parts such as building glass can be fill in. Thanks to 3D and active learning, our overall pipeline save us significant efforts in dense per-pixel and per-frame semantic labelling for background and objects. In practice, our labelling pipeline can reduce the time cost of dense labelling task per-image from nearly 1 hour to around 10 minutes, with the guarantee of passing our quality control process.

Labelling of Lane Mark Segments on Road. In self-driving, lane marks are information solely from static background. Fortunately, our collected survey-grade 3D points not only have high density, but also contain lighting intensity, dependent on which we can distinguish the lane mark on the roads. Specifically, we perform similar labelling process as 3D labelling of rigid background by labelling each 3D point to pre-defined lane mark labels listed in Table 3.

Nevertheless, different from labelling 3D point clusters where point clouds from buildings and trees are important, for lane marks, we only need to consider points on the road. Therefore, we take out the road point clouds based on normal directions, and perform orthogonal projection of these points from the bird view to a high resolution 2D image, as shown in Fig. 9, over which labellers draw a polygon for each lane mark on the road. In the meantime, our tool brings out the corresponding images, and highlights the regions in 2D for each labelled polygon, where the color and type of the labelled lanemark can be determined.

Labelling of Instance Segments. Thanks to an active labelling component with detection, it is easy for us to generalize the segmentation label map to produce instance masks given the segmented results from the object detection and segmentation networks. Specifically, we ask the labellers to refine the boundary between different instances when it is necessary, i.e., visually significantly not aligned with true object boundaries.

Control of Label Quality. Following the existing standard work flows of crowdsourcing object annotations [79], [80], [81], all our 2D/3D labeling tasks, e.g., 3D point cloud, 2D background, 2D instance and 3D lanemark, contain verification stages to control the label quality. Specifically, for each task, we have a detailed instruction to train our labellers, and a labeller is good to start labelling after passing a designed quiz. We will publish all our instructions on our website to benefit the community upon the publication of this paper.

After the labelling stage, we have a review stage, and each reviewer is an experienced labeller had sufficient labelled images (over 500) passed our label quality verification. The reviewer will verify the quality and the coverage of labelled regions. In addition, since we do video labelling, we also have reviewer to visually verify the semantics are temporally consistent in the next frame. Only if an image has passed two reviewers, it could be accepted as a valid ground truth.

Existing Issues. LiDAR scanners could fail on translucent and highly reflective surfaces such as mirrors of buildings. Though we fixed this problem in part of our recorded videos, shown in Fig. 8), we found it is still over laborious to fix every frame in all our videos even with active labelling. Therefore, part of video frames in our current release, pixels without 3D projection or active labelling, e.g., sky and part of building in Fig. 1, are set as *void*, so that they are ignored during the training and evaluation. We leave labelling of these pixels to our future work.

4 Deep Localization and Segmentation

As discussed in introduction (Section 1), ApolloScape contains various ground truth which enables multitask learning. In this paper, we show such a case by creating a deep learning based system for joint localization and semantic segmentation given a semantic 3D map [82], which we call DeLS-3D, as illustrated in Fig. 10. Specifically, at upper part, a pre-built 3D semantic map is available. During testing, an online stream of images and corresponding coarse camera poses from GPS/IMU are fed into the system. Firstly, for each frame, a semantic label map is rendered out given the input coarse camera pose, which is fed into a pose CNN jointly with the respective RGB image. The network calculates the relative rotation and translation, and yields a corrected camera pose. To incorporate the temporal correlations, the corrected poses from pose CNN are fed into a pose RNN to further improves the estimation accuracy in the stream. Last, given the rectified camera pose, a new label map is rendered out, which is fed together with the image to a segment CNN. The rendered label map helps to segment a spatially more accurate and temporally more consistent result for the image stream of video. In this system, since *ApolloScape* contains ground truth for both camera poses and segments, it can be trained with strong supervision at each end of outputs. The code for our system has been released at https://github.

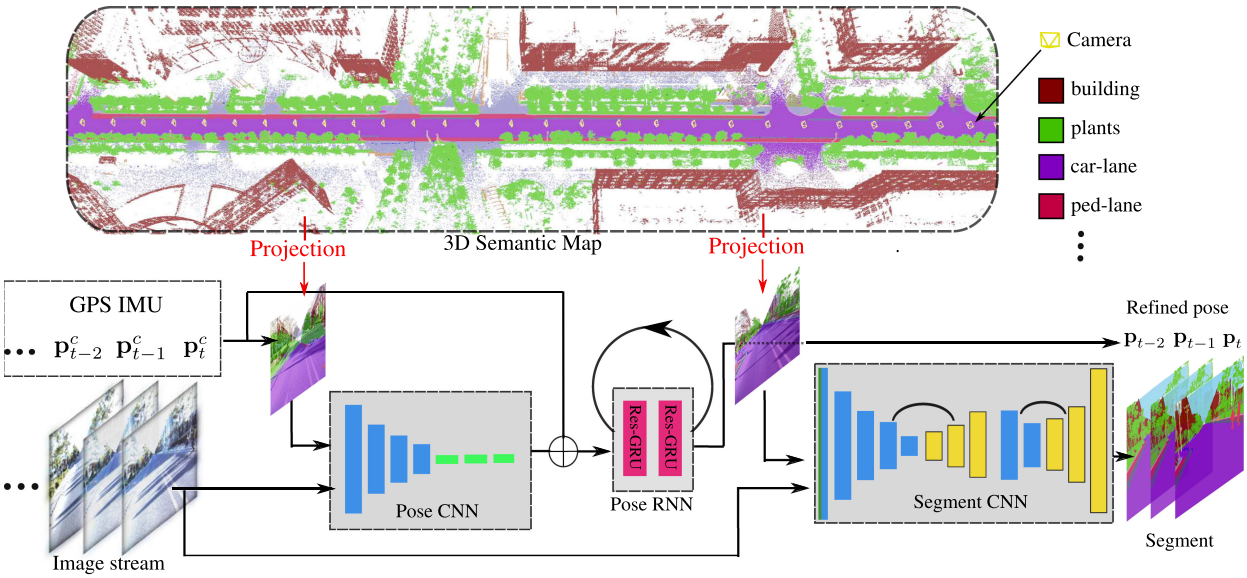


Fig. 10. DeLS-3D overview. The black arrows show the testing process, and red arrows indicate the rendering (projection) operation in training and inference. The yellow frustum shows the location of cameras inside the 3D map. The input of our system contains a sequence of images and corresponding GPS/IMU signals. The outputs are the semantically segmented images, each with its refined camera pose.

com/pengwangucla/DeLS-3D. In the following, we elaborate our network architectures and the loss functions to train the whole system.

4.1 Camera Localization with Motion Prior

Translation Rectification with Road Prior. One common localization priori for navigation is to use the 2D road map, by constraining the GPS signals inside the road regions. We adopt a similar strategy, since once the GPS signal is out of road regions, the rendered label map will be totally different from the street-view, and no correspondence can be found by the network.

To implement this constraint, firstly we render a 2D road map image with a rasterization grid of 0.05 m from our 3D semantic map by using only road 3D points, i.e., points belong to car-lane, pedestrian-lane and bike-lane etc. Then, at each pixel $[x, y] \in \mathbb{Z}^2$ in the 2D map, an offset value $\mathbf{f}(x, y)$ is pre-calculated indicating its 2D offset to the closest pixel belongs to road through the breath-first-search (BFS) algorithm efficiently.

During online testing, given a noisy translation $\mathbf{t} = [t_x, t_y, t_z]$, we can find the closest road points w.r.t. $\mathbf{t}$ using $[t_x, t_y] + \mathbf{f}(\lfloor t_x \rfloor, \lfloor t_y \rfloor)$ from our pre-calculated offset function. Then, a label map is rendered based on the rectified camera pose, which is fed to pose CNN.

CNN-GRU Pose Network Architecture. As shown in Fig. 10, our pose networks contain a pose CNN and a pose GRU-RNN. Particularly, the CNN of our pose network takes as inputs an image $\mathbf{I}$ and the rendered label map $\mathbf{L}$ from corresponding coarse camera pose $\mathbf{p}^c_i$. It outputs a 7 dimension vector $\hat{\mathbf{p}}_i$ representing the relative pose between the image and rendered label map, and we can get a corrected pose w.r.t. the 3D map by $\mathbf{p}_i = \mathbf{p}^c_i + \hat{\mathbf{p}}_i$. For the network architecture of pose CNN, we follow the design of DeMoN [55], which has large kernel to obtain bigger context while keeping the amount of parameters and runtime manageable. The convolutional kernel of this network consists a pair of 1D filters in y and x-direction, and the encoder gradually reduces the spatial resolution with stride of 2 while increasing the number of channels. We list the details of the network in our implementation details at Section 5.

Additionally, since the input is a stream of images, in order to model the temporal dependency, after the pose CNN, a multi-layer GRU with residual connection [83] is appended. More specifically, we adopt a two layer GRU with 32 hidden states as illustrated in Fig. 11. It includes high order interaction beyond nearby frames, which is preferred for improve the pose estimation performance. In traditional navigation applications of estimating 2D poses, Kalman filter [84] is commonly applied by assuming either a constant velocity or acceleration. In our case, because the vehicle velocity is unknown, transition of camera poses is learned from the training sequences, and in our experiments we show that the motion predicted from RNN is better than using a Kalman filter with a constant speed assumption, yielding further improvement over the estimated ones from our pose CNN.

Pose Loss. Following the PoseNet [48], we use the geometric matching loss for training, which avoids the balancing factor between rotation and translation. Formally, given a

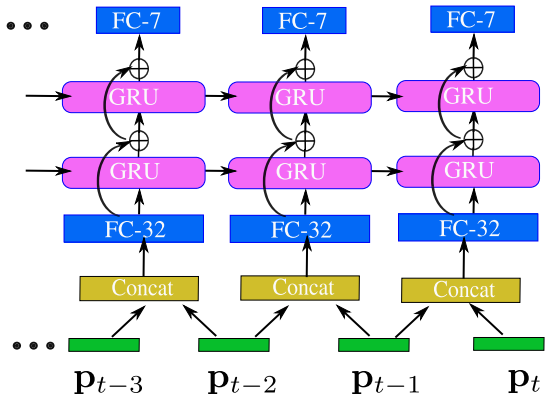


Fig. 11. The GRU RNN network architecture for modeling a sequence of camera poses.

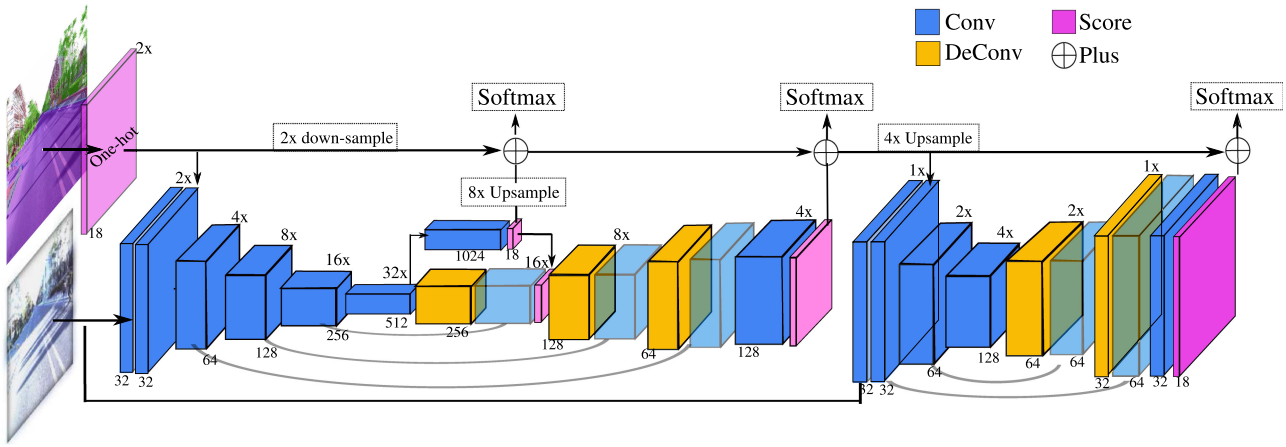


Fig. 12. Architecture of the segment CNN with rendered label map as a segmentation priori. At bottom of each convolutional block, we show the filter size, and at top we mark the downsample rates of each block w.r.t. the input image size. The 'softmax' text box indicates the places a loss is calculated. Details are in Section 4.2.

set of point cloud in 3D $\mathcal{P} = \{\mathbf{x}\}$, and the loss for each image is written as,

$$L(\mathbf{p}, \mathbf{p}^*) = \sum_{\mathbf{x} \in \mathcal{P}} \omega_{l_\mathbf{x}} |\pi(\mathbf{x}, \mathbf{p}) - \pi(\mathbf{x}, \mathbf{p}^*)|_2, \tag{3}$$

where $\mathbf{p}$ and $\mathbf{p}^*$ are the estimated pose and ground truth pose respectively. $\pi()$ is a projective function that maps a 3D point $\mathbf{x}$ to 2D image coordinates. $l_\mathbf{x}$ is the semantic label of $\mathbf{x}$ and $\omega_{l_\mathbf{x}}$ is a weight factor dependent on the semantics. Here, we set stronger weights for point cloud belong to certain classes like traffic light, and find it helps pose CNN to achieve better performance. In [48], only the 3D points visible to the current camera are applied to compute this loss to help the stability of training. However, the amount of visible 3D points is still too large in practical for us to apply the loss. Thus, we pre-render a depth map for each training image with a resolution of 256×304 using the ground truth camera pose, and use the back projected 3D points from the depth map for training.

4.2 Video Parsing with Pose Guidance

Having rectified pose at hand, one may direct render the semantic 3D world to the view of a camera, yielding a semantic parsing of the current image. However, the estimated pose is not perfect, fine regions such as light poles can be completely misaligned. Other issues also exist. For instance, many 3D points are missing due to reflection, e.g., regions of glasses, and points can be sparse at long distance. Last, dynamic objects in the input cannot be represented by the projected label map, yielding incorrect labelling at corresponding regions. Thus, we propose an additional segment CNN to tackle these issues, while taking the rendered label map as segmentation guidance.

Segment Network Architecture. As discussed in Section 2, heavily parameterized networks such as ResNet are not efficient enough for our online application. Thus, as illustrated in Fig. 12, our segment CNN is a light-weight network containing an encoder-decoder network and a refinement network, and both have similar architecture with the corresponding ones used in DeMoN [55] including 1D filters and mirror connections. However, since we have a segment guidance from the 3D semantic map, we add a residual stream (top part of Fig. 12), which encourages the network to learn the differences between the rendered label map and the ground truth. In [85], a full resolution stream is used to keep spatial details, while here, we use the rendered label map to keep the semantic spatial layout.

Another notable difference for encoder-decoder network from DeMoN is that for network inputs, shown in Fig. 12, rather than directly concatenate the label map with input image, we transform the label map to a score map through one-hot operation, and embed the score of each pixel to a 32 dimensional feature vector. Then, we concatenate this feature vector with the first layer output from image, where the input channel imbalance between image and label map is alleviated, which is shown to be useful by previous works [86]. For refinement network shown in Fig. 12, we use the same strategy to handle the two inputs. Finally, the segment network produces a score map, yielding the semantic parsing of the given image.

We train the segment network first with only RGB images, then fine-tune the network by adding the input of rendered label maps. This is because our network is trained from scratch, therefore it needs a large amount of data to learn effective features from images. However, the rendered label map from the estimated pose has on average 70 percent pixel accuracy, leaving only 30 percent of pixels having effective gradients. This could easily drive the network to over fit to the rendered label map, while slowing down the process towards learning features from images. Finally, for segmentation loss, we use the standard softmax loss, and add intermediate supervision right after the outputs from both the encoder and the decoder as indicated in Fig. 12.

5 Experiments

In this section, we first evaluate our online deep localization and segmentation algorithms (DeLS-3D) on two of our released roads, which is a subset of our full data. We compare it against other SOTA deep learning based visual localization, i.e., PoseNet [9], and segmentation algorithms i.e., ResNet38 [76], which shows the benefits of multitask unification.

TABLE 4
Compare the Accuracy of Different Settings for Pose Estimation from the Two Datasets

Data	Method	Trans (m) ↓	Rot (∘)↓	Pix. Acc(%)↑
Zpark	Noisy pose	3.45 ± 0.176	7.87 ± 1.10	54.01 ± 1.5
	Pose CNN w/o semantic	1.355 ± 0.052	0.982 ± 0.023	70.99 ± 0.18
	Pose CNN w semantic	1.331 ± 0.057	0.727 ± 0.018	71.73 ± 0.18
	Noisy pose w KF	2.56 ± 0.16	7.37 ± 1.01	55.1 ± 0.91
	Pose RNN w/o CNN	1.282 ± 0.061	1.731 ± 0.06	68.10 ± 0.32
	Pose CNN w KF	1.281 ± 0.06	0.833 ± 0.03	72.00 ± 0.17
	Pose CNN-RNN	$\mathbf{1.005} \pm 0.044$	$\mathbf{0.719} \pm 0.035$	$\mathbf{73.01} \pm 0.16$
Dlake	Pose CNN w semantic	1.667 ± 0.05	0.702 ± 0.015	87.83 ± 0.017
	Pose RNN w/o CNN	1.385 ± 0.057	1.222 ± 0.054	85.10 ± 0.03
	Pose CNN-RNN	$\mathbf{0.890} \pm 0.037$	$\mathbf{0.557} \pm 0.021$	$\mathbf{88.55} \pm 0.13$

Noisy pose indicates the noisy input signal from GPS, IMU, and 'KF' means kalman filter. The number after ± indicates the standard deviation (S.D.) from 10 simulations. ↓ & ↑ means lower the better and higher the better respectively. We can see the improvement is statistically significant.

Then, we elaborate the benchmarks setup online with ApolloScape and the current leading results, which follows many standard settings such as the ones from KITTI [2] and Cityscapes [3]. These tasks include semantic segmentation, semantic instance segmentation, self-localization, lanemark segmentation. Due to the "DeLS" algorithm proposed in this work does not follow those standard experimental settings, we could not provide its results for the benchmarks. Nevertheless, for each benchmark, we either ran a baseline result with SOTA methods or launched a challenge for other researchers, providing a reasonable estimation of the task difficulties.

5.1 Evaluate DeLS-3D

In this section, we evaluate various settings for pose estimation and segmentation to validate each component in the DeLS-3D system. For GPS and IMU signal, despite we have multiple scans for the same road segments, it is still very limited for training. Thus, follow [53], we simulate noisy GPS and IMU by adding random perturbation ϵ w.r.t. the ground truth pose following uniform distributions. Specifically, translation and rotation noise are set as $\epsilon_t \sim U(0, 7.5\ \mathrm{m})$ and $\epsilon_r \sim U(0^\circ, 15^\circ)$ respectively. We refer to realistic data [87] for setting the noisy range of simulation.

Datasets. Two roads early collected at Beijing in China are used in our evaluation. The first one is inside a technology park, named zhongguancun park (Zpark), and we scanned 6 rounds during different daytimes. The 3D map generated has a road length around $3\ \mathrm{km}$, and the distance between consecutive frames is around $5\ \mathrm{m}$ to $10\ \mathrm{m}$. We use 4 rounds of the video camera images for training and 2 for testing, yielding 2242 training images and 756 testing images. The second one we scanned 10 rounds and 4km near a lake, named daoxianghu lake (Dlake), and the distance between consecutive frames is around $1\ \mathrm{m}$ to $3\ \mathrm{m}$. We use 8 rounds of the video camera images for training and 2 for testing, yielding 17062 training images and 1973 testing images. The existing semantic classes in the two datasets are shown in Table 5, which are subsets from our full semantic classes.

Implementation Details. To quickly render from the 3D map, we adopt OpenGL to efficiently render a label map with the z-buffer handling. A 512×608 image can be generated in 70ms with a single Titan Z GPU, which is also the input size for both pose CNN and segment CNN. For pose CNN, the filter sizes of all layers are $\{32, 32, 64, 128, 256, 1024, 128, 7\}$, and the forward speed for each frame is 9 ms. For pose RNN, we sample sequences with length of 100 from our data for training, and the speed for each frame is 0.9 ms on average. For segment CNN, we keep the size the same as input, and the forward time is 90 ms. Overall, the inference speed is around 240 ms per-image for performing joint localization and segmentation. Both of the network is learned with 'Nadam' optimizer [88] with a learning rate of 10^{-3}. We sequentially train these three models due to GPU memory limitation. Specifically, for pose CNN and segment CNN, we stops at 150 epochs when there is no performance gain, and for pose RNN, we stops at 200 epochs. For data augmentation, we use the imgaug[2] library to add lighting, blurring and flipping variations. We keep a subset from training images for validating the trained model from each epoch, and choose the model performing best for evaluation.

For testing, since input GPS/IMU varies every time, i.e., $\mathbf{p}_t^c = \mathbf{p}^* + \epsilon$, we need to have a confidence range of prediction for both camera pose and image segment, in order to verify the improvement of each component we have is significant. Specifically, we report the standard variation of the results from a 10 time simulation to obtain the confidence range. Finally, we implement all the networks by adopting the MXNet [89] platform.

Evaluation Metrics. We use the median translation offset and median relative angle [9]. For evaluating segment, we adopt the commonly used pixel accuracy (Pix. Acc.), mean class accuracy (mAcc.) and mean intersect-over-union (mIOU) as that from [76].

Pose Evaluation. We first directly compare with the work of PoseNet [9], [48], and use their published code and geometric loss (Eq. (3)) to train a model on Zpark dataset. Due to scene appearance similarity of the street-view, we did not obtain a reasonable model with their methods [9], [48], better than the noisy GPS/IMU signal. Then, we experimented a leading open-source monocular SLAM algorithm, i.e., ORB-SLAM [90], to do self-localization. However, it also provides no better result than provided initial poses, since low-level ORB features fail to match robustly due to many non-diffusion/reflective components in Zpark, such as glass buildings and specular new roads, plus repetitive appearance on trees. Therefore, in Table 4, we majorly list the performance of estimated translation $\mathbf{t}$ and rotation $\mathbf{r}$ from our model variations. At the 1st row, we show the median error of GPS and IMU from our simulation. At the 2nd row, by using our pose CNN with an additional input of projected label map, the model can learn good relative pose between camera and GPS/IMU, which significantly reduces the error (60 percent for $\mathbf{t}$, 85 percent for $\mathbf{r}$). By adding semantic cues, i.e., road priori and semantic weights in Eq. (3), the pose errors are further reduced, especially for rotation (from 0.982 to 0.727 at the 3rd row). In fact, we found the most improvement is from semantic weighting, while the road priori helps marginally. In our future work, we would like to experiment larger noise and more data variations, which will better validate different cues.

When having an video input, we first evaluate a simple baseline which refines the GPS/IMU with Kalman filter [84], i.e., 'Noisy pose w KF', which reasonably reduces the errors.

2. https://github.com/aleju/imgaug

TABLE 5
Compare the Accuracy of Different Segment Networks Setting over Zpark (top) and Dlake (bottom) Dataset

Data	Method	mIOU	Pix. Acc	sky	car-lane	ped-lane	bike-lane	curb	t-cone	t-stack	t-fence	light-pole	t-light	tele-pole	t-sign	billboard	temp-build	building	sec.-stand	plants	object
Zpark	ResNet38 [76]	64.66	95.87	93.6	98.5	82.9	87.2	61.8	46.1	41.7	82.0	37.5	26.7	45.9	49.5	60.0	85.1	67.3	38.0	89.2	66.3
	SegCNN w/o Pose	68.35	95.61	94.2	98.6	83.8	89.5	69.3	47.5	52.9	83.9	52.2	43.5	46.3	52.9	66.9	87.0	69.2	40.0	88.6	63.8
	Render PoseRNN	32.61	73.1	-	91.7	50.4	62.1	16.9	6.6	5.8	30.5	8.9	6.7	10.1	16.3	22.2	70.6	29.4	20.2	73.5	-
	SegCNN w pose GT	79.37	97.1	96.1	99.4	92.5	93.9	81.4	68.8	71.4	90.8	71.7	64.2	69.1	72.2	83.7	91.3	76.2	58.9	91.6	56.7
	SegCNN w Pose CNN	68.6	95.67	94.5	98.7	84.3	89.3	69.0	46.8	52.9	84.9	53.7	39.5	48.8	50.4	67.9	87.5	69.9	42.8	88.5	60.9
	SegCNN w Pose RNN	**69.93**	**95.98**	94.9	98.8	85.3	90.2	71.9	45.7	57.0	85.9	58.5	41.8	51.0	52.2	69.4	88.5	70.9	48.0	89.3	59.5

Data	Method	mIOU	Pix. Acc	sky	car-lane	ped-lane	t-stack	t-fence	wall	light-pole	t-light	tele-pole	t-sign	billboard	building	plants	car	cyclist	motorbike	truck	bus
Dlake	SegCNN w/o Pose	62.36	96.7	95.3	96.8	12.8	21.5	81.9	53.0	44.7	65.8	52.1	87.2	55.5	66.8	94.5	84.9	20.3	28.9	78.4	82.1
	SegCNN w pose GT	73.10	97.7	96.8	97.5	41.3	54.6	87.5	70.5	63.4	77.6	70.5	92.1	69.2	77.4	96.1	87.4	24.5	43.8	80.0	85.7
	SegCNN w pose RNN	**67.00**	**97.1**	95.8	97.2	30.0	37.4	84.2	62.6	47.4	65.5	62.9	89.6	59.0	70.3	95.2	86.8	23.9	34.4	76.8	86.6

t is short for 'traffic' in the table. Here we drop the 10 times S.D. to save space because it is relatively small (≤ 0.1). Our results are especially good at parsing of detailed structures and scene layouts, which is visualized in Fig. 13.

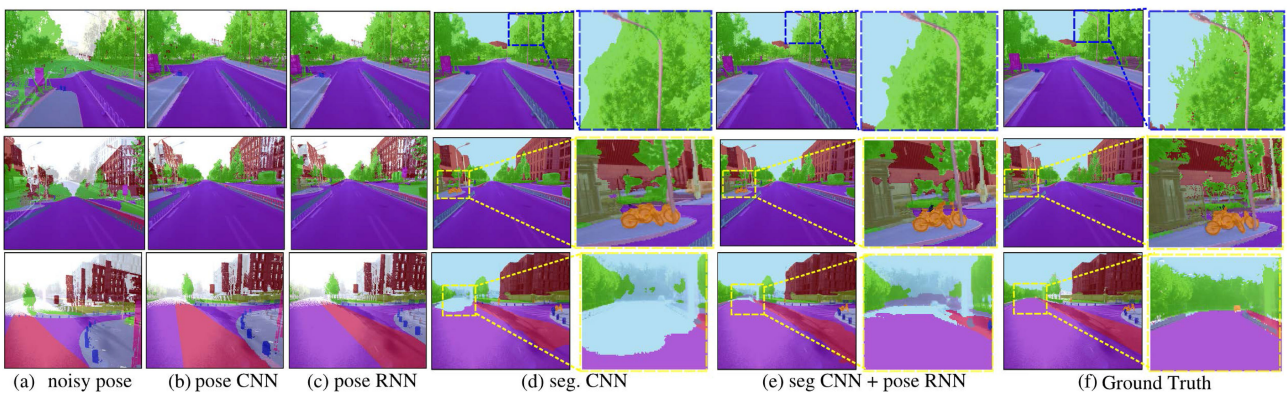


Fig. 13. Results from each intermediate stage out of the system over Zpark dataset. Label map is overlaid with the image. Improved regions are boxed and zoomed out (best in color). More results are shown in the online videos for Zpark and Dlake.

Then, we setup a baseline of performing RNN directly on the GPS/IMU signal, and as shown at 'Pose RNN w/o CNN', the estimated **t** is even better than pose CNN, while **r** is comparably much worse. This meets our expectation since the speed of camera is easier to capture temporally than rotation. Another baseline we adopt is performing to the output from Pose CNN by assuming a constant speed which we set as the averaged speed from training sequences. As shown at 'Pose CNN w KF', it does improve slightly for translation, but harms rotation, which means the filter over smoothed the sequence. Finally when combining pose CNN and RNN, it achieves the best pose estimation both for **t** and **r**. We visualize some results at Fig. 13a, 13b, 13c. Finally at bottom of Table 4, we list corresponding results on Dlake dataset, which draws similar conclusion with that from Zpark dataset.

Segment Evaluation. At top part of Table 5, we show the scene parsing results of Zpark dataset. First, we adopt one of the SOTA parsing network on the CityScapes, i.e., ResNet38 [76], and train it with Zpark dataset. It utilizes pre-trained parameters from the CityScapes [3] dataset, and run with a $1.03s$ per-frame with our resolution. As shown at the 1st row, it achieve reasonable accuracy compare to our segment CNN (2nd row) when there is no pose priori. However, our network is 10x faster. At 3rd row, we show the results of rendered label map with the estimated pose after pose RNN. Clearly, the results are much worse due to missing pixels and object misalignment. At 4th row, we use the rendered label map with ground truth pose as segment CNN guidance to obtain an upper-bound for our segmentation performance. In this case, the rendered label map aligns perfectly with the image, thus significantly improves the results by correct labelling most of the static background. At 5th and 6th row, we show the results trained with rendered label map with pose after pose CNN and pose RNN respectively. We can see using pose CNN, the results just improve slightly compare to the segment CNN. From our observation, this is because the offset is still significant for some detailed structures, e.g., light-pole. However, when using the pose after RNN, better alignment is achieved, and the segment accuracy is improved significantly especially for thin structured regions like pole, as visualized in Fig. 13, which demonstrates the effectiveness of our strategy.

Bottom part of Table 5 shows the results over the larger Dlake dataset with more object labelling, where we see clearer improvement, i.e., from 62.36 to 67.00, and here the rendered label provides a background context for object segmentation, which also improve the object parsing performance. In all classes, we observe the performance of traffic-light drops. In our opinion, the majorly reason is traffic-light only exists in intersection of roads, which happens much fewer than objects such as light-pole, yielding overfitting to the projected label maps from pose. We may fix this issue by training with even larger dataset or better class balancing strategies, which is left to our future work.

In Fig. 13, we visualize several examples from our results at the view of camera. In the figure, we can see the noisy pose (a), is progressively rectified by pose CNN (b) and

TABLE 6
Results of Image Parsing Based on ResNet-38 Network

Group	Class	IoU	
		Cityscapes	ApolloScape
movable object	car	94.67	87.12
	motorcycle	70.51	27.99
	bicycle	78.55	48.65
	person	83.17	57.12
	rider	65.45	6.58
	truck	62.43	25.28
	bus	88.61	48.73
mIoU		77.63	43.07
surface	road	97.94	92.03
	sidewalk	84.08	46.42
infrastructure	fence	61.49	42.08
	traffic light	70.98	67.49
	pole	62.11	46.02
	traffic sign	78.93	79.60
	wall	58.81	8.41
	building	92.66	65.71
nature	vegetation	92.41	90.53

pose RNN (c) from view of camera. Additionally, at (d) and (e), we compare the segment results without and with camera pose respectively. As can be seen at the boxed regions, the segment results with rendered label maps provide better accuracy in terms of capturing region details at the boundary, discovering rare classes and keeping correct scene layout. All of above could be important for applications, e.g., figuring out the traffic signs and tele-poles that are visually hard to detect. For additional visualization, please check our demo videos online.[3,4]

5.2 Benchmarks and Baselines

With various tasks and large amount of labelled data we have proposed, it would be non-practical for us to extensively explore algorithms over all of them. Therefore, we release the data to research community, and set up standard evaluation benchmarks. Currently, four challenges have been set up online for evaluation by withholding part of our labelled results as test set, which include semantic segmentation [20], instance segmentation [21], self-localization [23], lanemark segmentation [22].

For evaluation, in the tasks of semantic segmentation, lanemark segmentation, we adopt mean IoU, and in the task of self-localization, we adopt median translation and rotation offset, which are described in evaluation of DeLS-3D (Section 5.1). For the task of instance segmentation, we use interpolated average precision (AP) [91] under various IoU thresholds which is used for the COCO challenge [36]. Later, we elaborate the split of each dataset, the leading method on each benchmark currently.

Semantic Segmentation. For video semantic segmentation, until now, we haven't receive valid results from the challenge. This probably is due to the extremely large amount of training videos in ApolloScape, making training a model with SOTA deep learning models such as ResNet [59] not-practical. Thus, we select a subset from the whole data for comparison of one model performance between ApolloScape and Cityscapes. Specifically, 5,378 training images and 671 testing images are carefully selected from our 140K labelled semantic video frames for setting up the benchmark, which maintains the diversity and objects appeared of the collected scenes. The selected images will be released at our website [20].

We conducted our experiments using ResNet-38 network [76] that trades depth for width comparing with the original ResNet structure [59]. We fine-tune their released model using our training with initial learning rate 0.0001, standard SGD with momentum 0.9 and weight decay 0.0005, random crop with size 512×512, 10 times data augmentation that includes scaling and left-right flipping, and we train the network for 100 epochs. The predictions are computed with the original image resolution without any post-processing steps such as multi-scale ensemble etc.. Table 6 shows the parsing results of classes in common for these two datasets. Notice that using exactly same training procedure, the test IoU with our dataset are much lower than that from the Cityscapes mostly due to the challenges we have mentioned at Section 3.2, especially for movable objects, where mIoU is 34.6 percent lower than the corresponding one for the Cityscapes.

Here, we leave the training a model with the our full dataset to the research community and our future work.

Instance Segmentation. This task is an extension of semantic object parsing by jointly considering detection and segmentation. Specifically, we select 39212 training images and 1907 testing images, and set up a challenge benchmark online [21] evaluating 7 objects in our dataset (Upper part of Table 6) to collect potential issues within autonomous driving scenario. During the past few month, there are over 140 teams attended our challenge, which reveals our community is much more interested in object level understanding rather than scene segmentation.

The leading results from our participants are shown in Table 8, where we can see in general the reported mAP of winning teams are lower than those reported in Cityscapes benchmarks [92], by using similar strategies [93] modified from MaskRCNN [13]. Based on the challenge reports from the winning team [94], comparing to Cityscapes, ApolloScape contains more tiny and occluded objects (60 percent object has scale less than 32 pixels), which leads to significant drop of performance when transfer models trained on other datasets.

Lanemark Segmentation. Lanemark segmentation task follows the same metric as semantic segmentation, which contains 132189 training images and 33790 testing images. Our in-house challenge benchmark [22] chooses to evaluate 35 most common lane mark types on the road as listed in Table 3.

Until the submission of this paper, we only have one work based on ResNet-38 network [76] evaluated, probably due to the large amount of data (160K+ images). We show the corresponding detailed results in Table 7, where we can see the mIoU of each class are still very limited (40 percent) comparing to the accuracy of leading semantic segmentation algorithms on general classes. We think this is mostly

3. Zpark: https://www.youtube.com/watch?v=fqglYBipNfQ
4. Dlake https://www.youtube.com/watch?v=fqglYBipNfQ

TABLE 7
The IoU Using One SOTA Semantic Segmentation Architecture, i.e., ResNet38 [76]

Method	mIOU	s_w_d	s_y_d	ds_y_dn	b_w_g	b_y_g	s_w_s	s_w_p	c_wy_z	a_w_t	a_w_tl	a_w_tr	a_w_l	a_w_r	a_w_lr	b_n_sr	r_wy_np	om_n_n
ResNet38 [76]	40.0	48.6	53.1	57.8	52.1	22.7	36.4	18.7	59.1	40.4	27.1	49.1	57.4	20.9	0.01	0.9	36.1	40.5

Here the amount of class is less than that in Table 3 due to zero accuracy over some predefined labels.

because the high contrast, dimmed and broken lane marks on the road such as the cases shown in Fig. 3.

Self-Localization. We use the same metrics for evaluating camera pose, i.e., median offset of translation and rotation, as described in Section 5.1. This task contains driving videos in 6 sites from Beijing, Guangzhou, Chengdu and Shanghai in China, under multiple driving scenarios and day times. In total, we provide 153 training videos and 71 testing videos including over 300k image frames, and build an in-house challenge benchmark website [23] most recently.

Currently, we also have few submissions, while the leading one published is from one of the SOTA method for large-scale image based localization [97]. The method is based on image retrieval with learned deep features via various triplet losses. We show their reported number in Table 9, where the localization errors are surprisingly small, i.e., translation is around 15cm and rotation error is around 0.14 degree. Originally, we believe image appearance similarity on the street or highway can fail deep network models. However, from the participant results, especially designed features distinguish minor appearance changes and provide high accurate localization results. Another possibility is that our acquisition vehicle always drives in a roughly constant speed, reducing the issues from speed changing in real applications. In the near future, hopefully, we can add more challenging scenarios with more variations in driving speed and weathers.

In summary, from the dataset benchmarks we set up and evaluated algorithms, we found for low-level localization, the results are impressively good, while for high level semantic understanding, Apolloscape provides additional challenges and new issues, yielding limited accuracy for SOTA algorithms, i.e., best mAP is around 33 percent for instance segmentation, and best mIoU is around 40 percent for lane segmentation. Comparing to human perception, visual based algorithms for autonomous driving definitely need further research to handle extremely difficult cases.

TABLE 8
Results of Top Ranked Instance Segmentation Algorithms in ApolloScape and Cityscapes (Numbers Are Obtained at the Date of Submission)

Dataset	metric	1_{st}	2_{nd}	3_{nd}
ApolloScape	mAP	33.97 [94]	30.22 [95]	25.02 [96]
Cityscapes		38.0	37.2	36.4 [93]

TABLE 9
Detailed Localization Accuracy of the Leading Results on Our Benchmark from [97]

Road ID	Trans (m) ↓	Rot (∘) ↓
Road11	0.0476	0.0452
Road12	0.1115	0.0528
Road14	0.0785	0.0825
Road15	0.0711	0.1240
Road16	0.1229	0.2063
Road17	0.4934	0.3135
mean	0.1542	0.1374

6 Conclusion and Future Work

In this paper, we present the *ApolloScape*, a large, diverse, and multi-task dataset for autonomous driving, which includes high density 3D point cloud map, per-pixel, per-frame semantic image label, lane mark label, semantic instance segmentation for various videos. Every frame of our videos is geo-tagged with high accurate GPS/IMU device. *ApolloScape* is significantly larger than existing autonomous driving datasets, e.g., KITTI [2] and Cityscapes [3], yielding more challenges for computer vision research field. In order to label such a large dataset, we developed an active 2D/3D joint annotation pipeline, which effectively accelerate the labelling process. Back on *ApolloScape*, we developed a joint localization and segmentation algorithm with a 3D semantic map, which fuses multi-sensors, is simple and runs efficiently, yielding strong results in both tasks. We hope it may motivate researcher to develop algorithms handling multiple tasks simultaneously by considering their inner geometrical relationships. Finally, for each individual task, we set up an online evaluation benchmark where different algorithms can compete with a fair platform.

Last but not the least, *ApolloScape* is an evolving dataset, not only in terms of data scale, but also in terms of various driving conditions, tasks and acquisition devices. For example, firstly, we plan to enlarge our dataset to contain more diversified driving environments including snow, and foggy. Secondly, we also released our labelled 3D cars [19], stereo images, 3D humans and tracking [98] of objects in 3D recently. Thirdly, we plan to mount a panoramic camera system, and Velodyne [4] in near future to generate depth maps for objects and panoramic images.

Acknowledgments

This work is supported by Baidu Inc. We also thank the work of Xibin Song, Binbin Cao, Jin Fang, He Jiang, Yu Zhang, Xiang Gu, and Xiaofei Liu for their laborious efforts in organizing data, helping writing label tools, checking labelled results and manage the content of benchmark websites. We thank Alan L. Yuille, Hongdong Li and Andreas Geiger for benchmark suggestions. Xinyu Huang and Peng Wang contributed equally to this work.

REFERENCES

[1] "ApolloScape Website." 2018. [Online]. Available: apolloscape.auto

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Robot. Res.*, vol. 32, pp. 1231–1237, 2013.

[3] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.

[4] Velodyne Lidar, "HDL-64E," 2018. [Online]. Available: http://velodynelidar.com/, Accessed on: Mar. 1, 2018.

[5] A. Kar, C. Häne, and J. Malik, "Learning a multi-view stereo machine," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 365–376.

[6] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "Deepmvs: Learning multi-view stereopsis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2821–2830.

[7] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 767–783.

[8] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 108–125.

[9] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2938–2946.

[10] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," *ISPRS J. Photogrammetry Remote Sens.*, 2018.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6896–6906.

[12] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018. doi: 10.1109/TPAMI.2018.2844175

[14] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4013–4022.

[15] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2014, pp. 75–82.

[16] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1966–1974.

[17] F. Guney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4165–4175.

[18] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3559–3568.

[19] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, and R. Yang, "Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[20] ApolloScape, "Semantic segmentation." 2018. [Online]. Available: http://apolloscape.auto/scene.html

[21] ApolloScape, "Instance segmentation." 2018. [Online]. Available: https://www.kaggle.com/c/cvpr-2018-autonomous-driving

[22] ApolloScape, "Lanemark segmentation." 2018. [Online]. Available: http://apolloscape.auto/lane_segmentation.html

[23] ApolloScape, "Localization." 2018. [Online]. Available: http://apolloscape.auto/self_localization.html

[24] W. Peng, et al., "ApolloScape API." 2018. [Online]. Available: https://github.com/ApolloScapeAuto/dataset-api

[25] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, "Joint semantic segmentation and 3d reconstruction from monocular video," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 703–718.

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

[27] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "Torontocity: Seeing the world with a million eyes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3009–3017.

[28] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 22–29.

[29] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv: 1805.04687*, 2018.

[30] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3234–3243.

[31] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2213–2222.

[32] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.*, 2014, pp. 31–42.

[33] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[34] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, et al.," Benchmarking 6dof outdoor visual localization in changing conditions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, vol. 1, pp. 8601–8610.

[35] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[36] 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 *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[37] "Unity Development Platform." 2005. [Online]. Available: https://unity3d.com/

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

[39] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, no. 5, 2017, Art. no. 6.

[40] Y. Chen, W. Li, and L. Van Gool, "Road: Reality oriented adaptation for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7892–7901.

[41] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *Int. J. Comput. Vis.*, vol. 13, no. 3, pp. 331–356, 1994.

[42] L. Kneip, H. Li, and Y. Seo, "Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 127–142.

[43] P. David, D. Dementhon, R. Duraiswami, and H. Samet, "Softposit: Simultaneous pose and correspondence determination," *Int. J. Comput. Vis.*, vol. 59, no. 3, pp. 259–284, 2004.

[44] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Pose priors for simultaneously solving alignment and correspondence," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 405–418.

[45] D. Campbell, L. Petersson, L. Kneip, and H. Li, "Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, vol. 1, no. 3.

[46] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla, "Are large-scale 3d models really necessary for accurate visual localization?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6175–6184.

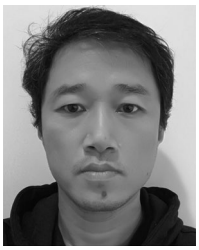
[47] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[48] A. Kendall, R. Cipolla, et al., "Geometric loss functions for camera pose regression with deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 3, Art. no. 8.

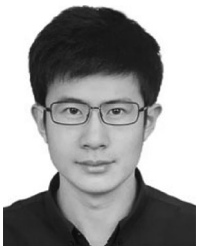
[49] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using lstms for structured feature correlation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 627–637.

[50] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 3, pp. 1–9.

[51] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long short-term memory kalman filters: Recurrent neural estimators for pose regularization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5525–5533.
[52] K.-N. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler, "Vso: Visual semantic odometry," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 234–250.
[53] K. Vishal, C. Jawahar, and V. Chari, "Accurate localization by fusing images and gps signals," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2015, pp. 17–24.
[54] Z. Laskar, I. Melekhov, S. Kalia, and J. Kannala, "Camera relocalization by computing pairwise relative poses using convolutional neural network," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 920–929.
[55] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 5, 2017, Art. no. 6.
[56] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6230–6239.
[57] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr, "Higher order conditional random fields in deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 524–540.
[58] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, "Scene labeling with lstm recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3547–3555.
[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[60] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016.
[61] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 405–420.
[62] A. Kundu, V. Vineet, and V. Koltun, "Feature space optimization for semantic video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3168–3175.
[63] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[64] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4453–4462.
[65] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2349–2358.
[66] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys, "Joint 3d scene reconstruction and class segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 97–104.
[67] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 2, pp. 6565–6574.
[68] RIEGL, "VMX-1HA." 2000. [Online]. Available: http://www.riegl.com/
[69] TuSimple, "Lanemark segmentation." 2017. [Online]. Available: http://benchmark.tusimple.ai/#/
[70] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *Int. J. Comput. Vis.*, vol. 120, no. 3, pp. 300–323, 2016.
[71] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding," *arXiv: 1810.06125*, 2018.
[72] J. Xie, M. Kiefel, M.-T. Sun, and A. Geiger, "Semantic instance annotation of street scenes by 3d to 2d label transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3688–3697.
[73] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter, "Object partitioning using local convexity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 304–311.
[74] "Point Cloud Library." 2010. [Online]. Available: http://pointclouds.org.
[75] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.
[76] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv:1611.10080*, 2016.
[77] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[78] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Joint object and part segmentation using deep learned potentials," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1573–1581.
[79] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," in *Proc. Workshops 26th AAAI Conf. Artif. Intell.*, 2012, vol. 1, no. 2.
[80] A. Kovashka, O. Russakovsky, L. Fei-Fei, K. Grauman, et al., "Crowdsourcing in computer vision," *Found. Trends® Comput. Graph. Vis.*, vol. 10, no. 3, pp. 177–243, 2016.
[81] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.
[82] P. Wang, R. Yang, B. Cao, W. Xu, and Y. Lin, "Dels-3d: Deep localization and segmentation with a 3d semantic map," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5860–5869.
[83] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016.
[84] R. E. Kalman, et al., "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.
[85] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3309–3318.
[86] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2650–2658.
[87] B.-H. Lee, J.-H. Song, J.-H. Im, S.-H. Im, M.-B. Heo, and G.-I. Jee, "Gps/dr error estimation for autonomous vehicle localization," *Sens.*, vol. 15, no. 8, pp. 20 779–20 798, 2015.
[88] T. Dozat, "Incorporating nesterov momentum into adam," 2016.
[89] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015.
[90] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[91] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 297–312.
[92] "Cityscapes instance segmentation benchmark." [Online]. Available: https://www.cityscapes-dataset.com/benchmarks/#instance-level-scene-lab eling-task
[93] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8759–8768.
[94] Z. Yueqing, L. Zeming, and G. Yu, "Find tiny instance segmentation," 2018. [Online]. Available: http://www.skicyyu.org/WAD/wad_final.pdf,
[95] Smart_Vision_SG, "Wad instance segmentation 2nd place," 2018. [Online]. Available: https://github.com/Computational-Camera/Kaggle-CVPR-2018-WAD-Video-Segm entation-Challenge-Solution
[96] SZU_N606, "Wad instance segmentation 3rd place," 2018. [Online]. Available: https://github.com/wwoody827/cvpr-2018-autonomous-driving-autopilot-sol ution
[97] L. Liu, H. Li, and Y. Dai, "Deep stochastic attraction and repulsion embedding for image based localization," *arXiv: 1808.08779*, 2018.
[98] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," *Proc. AAAI*, 2019.

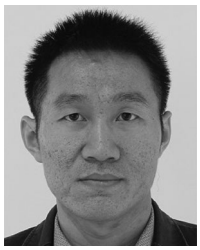


Xinyu Huang received the BS degree from Huazhong University of Science and Technology and the PhD degree from the University of Kentucky. He is currently a senior research scientist with Baidu Research and an associate professor with the North Carolina Central University. His research areas include computer vision, image processing, pattern recognition, and machine learning.



Peng Wang received the BS and MS degrees from Peking University, China, and the PhD degree from the University of California, Los Angeles, advised by Prof. Alan Yuille. He is a senior research scientist with Baidu USA LLC. His research interests include image parsing and 3D understanding, and vision based autonomous driving system. He has around 30 published papers in ECCV/CVPR/ICCV/NIPS.

Xinjing Cheng is a research scientist with Robotics and Autonomous Driving Lab, Baidu Research, Baidu Inc., Beijing, China. Before that, he was a research assistant with the Intelligent Bionic Center, Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS), Shenzhen, China. His current research interests include computer vision, deep learning, robotics and autonomous driving.



Dingfu Zhou received the BE and ME degrees in signal and information processing from Northwestern Polytechnical University, Xian, China, and the PhD degree in system and control from Sorbonne Universités, Université de Technologie de Compiègne, Compiègne, France, in 2014. He a senior researcher with Robotics and Autonomous Driving Laboratory (RAL) of Baidu. Before joining in Baidu, he worked as a PostDoc researcher with the Research School of Engineering, Australian National University, Canberra, Australia. His research interests include simultaneous localization and mapping, structure from motion, classification and their application in autonomous driving.

Qichuan Geng received the BS degree from Beihang University, in 2012. He is working toward the PhD degree in State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing, China. His main research interests include computer vision, semantic segmentation and scene geometry recovery.

Ruigang Yang is the chief scientist for 3D vision with Baidu. He is also a full professor with the University of Kentucky (on leave). His research interests include 3D computer vision and 3D computer graphics, in particular 3D modeling and 3D data analysis. He has published more than 100 papers with an H-index of 48. He is an associate editor for the *IEEE Transactions on Pattern Analysis and Machine*. He has been a program co-chair for 3DIMPVT (now 3DV) 2011 and WACV 2014, and he has been area chairs for both ICCV and CVPR multiple times.