# DeLS-3D: Deep Localization and Segmentation with a 3D Semantic Map

Peng Wang, Ruigang Yang, Binbin Cao, Wei Xu, Yuanqing Lin
Baidu Research
National Engineering Laboratory for Deep Learning Technology and Applications
{wangpeng54, yangruigang, caobinbin, wei.xu, linyuanqing}@baidu.com

## Abstract

*For applications such as augmented reality, autonomous driving, self-localization/camera pose estimation and scene parsing are crucial technologies. In this paper, we propose a unified framework to tackle these two problems simultaneously. The uniqueness of our design is a sensor fusion scheme which integrates camera videos, motion sensors (GPS/IMU), and a 3D semantic map in order to achieve robustness and efficiency of the system. Specifically, we first have an initial coarse camera pose obtained from consumer-grade GPS/IMU, based on which a label map can be rendered from the 3D semantic map. Then, the rendered label map and the RGB image are jointly fed into a pose CNN, yielding a corrected camera pose. In addition, to incorporate temporal information, a multi-layer recurrent neural network (RNN) is further deployed improve the pose accuracy. Finally, based on the pose from RNN, we render a new label map, which is fed together with the RGB image into a segment CNN which produces per-pixel semantic label. In order to validate our approach, we build a dataset with registered 3D point clouds and video camera images. Both the point clouds and the images are semantically-labeled. Each video frame has ground truth pose from highly accurate motion sensors. We show that practically, pose estimation solely relying on images like PoseNet [25] may fail due to street view confusion, and it is important to fuse multiple sensors. Finally, various ablation studies are performed, which demonstrate the effectiveness of the proposed system. In particular, we show that scene parsing and pose estimation are mutually beneficial to achieve a more robust and accurate system.*

## 1. Introduction

In applications like robotic navigation [34] or augment reality [2], visual-based 6-DOF camera pose estimation [5, 31, 25, 10], and concurrently parsing each frame of a video into semantically meaningful regions [48, 46, 6] efficiently are the key components, which are attracting much attention in computer vision.

Currently, most state-of-the-art (SOTA) algorithms try to solve both tasks using solely visual signals. For camera localization, geometric based methods are relying on visual feature matching, *e.g.* systems of Perspective-n-Points (PnP) [19, 26, 5] when a 3D map and an image is provided, or systems of SLAM [15, 32, 33] when there is a video. Such systems are dependent on local appearance, which could fail when confronted with low-texture environments. Most recently, deep learning based methods, *e.g.* for either images [25] or videos [8], have been developed, which not only consider hierarchical features, while yielding real-time performance. Nevertheless, those methods are good for environments with rich distinguishable features, such as these in the Cambridge landmarks dataset [25]. They could fail for common street views with very similar appearances or even repetitive structures.

For scene parsing, approaches [48, 6] based on deep fully convolutional network (FCN) with ResNet [21] are the best-performing algorithms for single image input. When the input is video, researchers [28, 49] incorporate the optical flow between consecutive frames, which not only accelerates the parsing, but also improve temporal consistency. Furthermore, for static background, one may use structure-from-motion (SFM) techniques [44] to jointly parse and reconstruct [27]. However, these methods could be either time-consuming or hard to generalize for applications asking online real-time performance.

In this paper, we aim to solve this camera localization and scene parsing problem jointly from a more practical standpoint. 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 deep-learning based 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. Our setting is considered on par with the widely used mobile navigation systems, whereas the 2D labeled map is replaced with a 3D semantic map, and we are able to virtually place the navigation signals into the images with more accurate self-localization

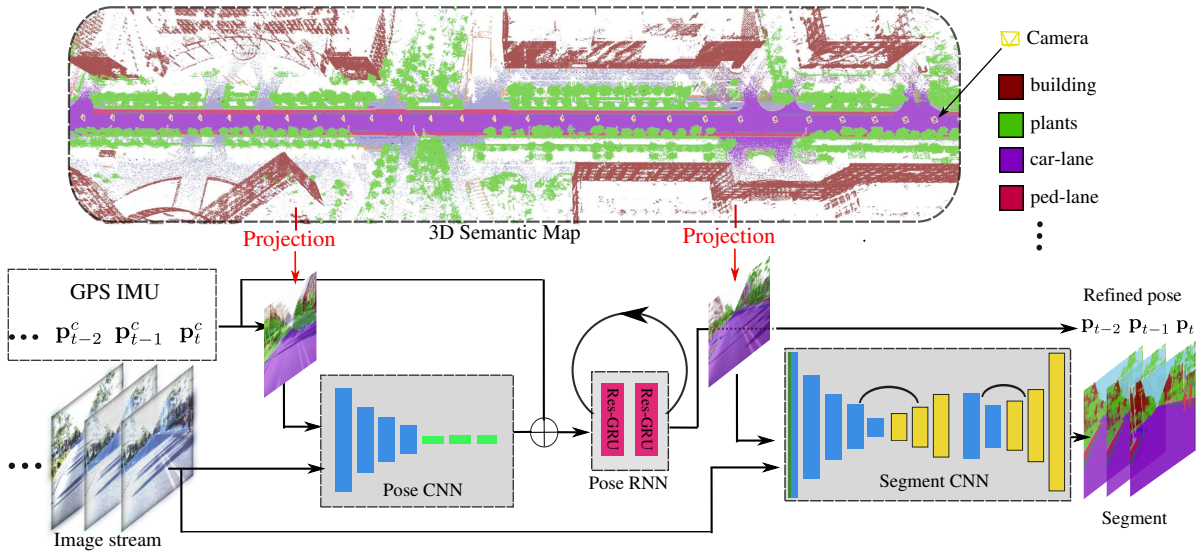



Figure 1: System overview. The black arrows show the testing process, and red arrows indicate the rendering (projection) operation in training and inference. The yellow frustum shows the location of cameras inside the 3D map. The input of our system contains a sequence of images and corresponding GPS/IMU signals. The outputs are the semantically segmented images, each with its refined camera pose.

and scene parsing on-the-fly. Promisingly, with the accelerated development of autonomous driving, city-scale 3D semantic maps are being collected and built (such as the TorontoCity dataset [43]). Here, we constructed our own data with high quality 3D semantic map, which is captured via a high-accuracy mobile LIDAR device from *Riegl*[1].

Last but not least, within our deep learning 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 $512 \times 608$ in within 90ms with pixel accuracy around 96% without model compression, which demonstrates its efficiency and effectiveness.

In summary, the contributions of this paper are:

- We propose a deep learning based system for fusing multiple sensors, *i.e.* RGB images, customer-grad GPS/IMU, and 3D semantic maps, which improves the robustness and accuracy for camera localization and scene parsing.
- Camera poses and scene semantics are designed to handle jointly in a unified framework.
- We create a dataset from real scenes to fully evaluate our approach. It includes dense 3D semantically labelled point clouds, ground truth camera poses and pixel-level semantic labels of video camera images, which will be released in order to benefit related researches.

The structure of this paper is organized as follows. We first give an overview of our system in Sec. 1.1 and talk about related works in Sec. 2. In Sec. 3, we describe the uniqueness of our data from the existing outdoor datasets, and introduce our collection and labelling process. Then, Sec. 4 presents details of our system. We perform full evaluation quantitatively for both pose estimation and scene parsing in Sec. 5, and Sec. 6 concludes the paper and points out future directions.

### 1.1. Framework

The framework of our system is illustrated in Fig. 1. 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 our data contains ground truth for both camera poses and segments, it can be trained with strong supervision at each end of outputs.

[1] http://www.rieglusa.com/index.html

## 2. Related Work

Estimating camera pose and semantic parsing given a video or a single image have long been center problems for computer vision. Here we summarize the related works in several aspects without enumerating them all due to space limitation. Notice that our application is autonomous driving and navigation, we therefore focus on outdoor cases with street-view input. Localization and general parsing in the wild is beyond the scope of this paper.

**Camera pose estimation.** Traditionally, localizing an image given a set of 3D points is formulated as a Perspective-$n$-Point (P$n$P) problem [19, 26] 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 [11, 31]. Campbell *et al.* [5] propose a global-optimal solver which leverage the prior. In the case that geo-tagged images are available, Sattler *et al.* [39] 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 [15] 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 [25, 24] takes a low-resolution image as input, which can estimate pose in 10ms w.r.t. a feature rich environment composed of distinguished landmarks. LSTM-PoseNet [20] further captures a global spatial context after CNN features. Given an video, later works incorporate Bi-Directional LSTM [8] or Kalman filter LSTM [10] to obtain better results with temporal information. 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 [42], 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 [29, 41] 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.

**Scene parsing.** For parsing a single image of street views (e.g., these from CityScapes [9]), most state-of-the-arts (SOTA) algorithms are designed based on a FCN [46] and a multi-scale context module with dilated convolution [6], pooling [48], CRF [1], or spatial RNN [4]. However, they are dependent on a ResNet [21] with hundreds of layers, which is too computationally expensive for applications that require real-time performance. Some researchers apply small models [35] or model compression [47] for acceleration, with the cost of reduced accuracy. When the input is a video, spatial-temporal informations are jointly considered, Kundu *et al.* [28] use 3D dense CRF to get temporally consistent results. Recently, optical flow [12] between consecutive frames is computed to transfer label or features [16, 49] 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 [41] for inference efficiency.

**Joint 2D-3D for video parsing.** Our work is also related to joint reconstruction, pose estimation and parsing [27, 18] through embedding 2D-3D consistency. Traditionally, reliant on structure-from-motion (SFM) [18] 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 [40] 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. Dataset

**Motivation.** As described in the Sec. 1.1, our system is designed to work with available 3D motion sensors and a semantic 3D map. However, public outdoor datasets such as KITTI and CityScapes do not contain such information, in particular the 3D map. The TortoroCity dataset [43] may be used while is not open to public yet. As summarized in Tab. 1, we list several key requirements to perform our experiments, which none of current existing datasets fully satisfy.

| Dataset | Real data | Camera pose | 3D semantic map | Video per-pixel label |
|---|---|---|---|---|
| CamVid [3] | ✓ | - | - | - |
| KITTI [17] | ✓ | ✓ | sparse points | - |
| CityScapes [9] | ✓ | - | - | selected frames |
| Toronto [43] | ✓ | ✓ | 3d building & road | selected pixels |
| Synthia [38] | - | ✓ | - | ✓ |
| P.F.B. [37] | - | ✓ | - | ✓ |
| Ours | ✓ | ✓ | dense point cloud | ✓ |

Table 1: Compare our dataset with the other related outdoor street-view datasets for our task. 'Real data' means whether the data is collected from the physical world. '3D semantic map' means whether it contains a 3D map of scenes with semantic label. 'Video per-pixel label' means whether it has per-pixel semantic label.

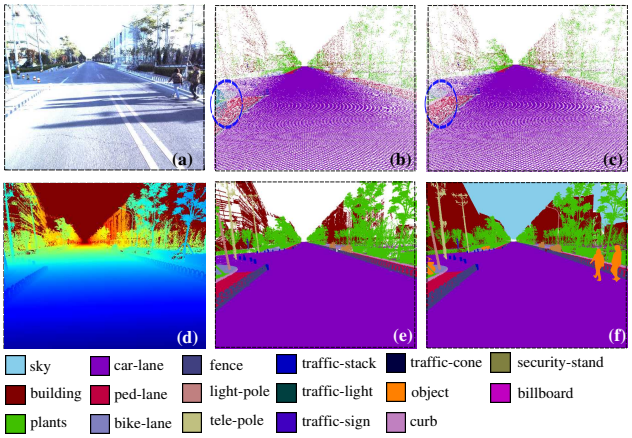



Figure 2: An example of our collected street-view dataset. (a) Image. (b) Rendered label map with 3D point cloud projection, with an inaccurate moving object (rider) circled in blue. (c) Rendered label map with 3D point cloud projection after points with low temporal consistency being removed. (d) & (e) Rendered depth map of background and rendered label map after class dependent splatting in 3D point clouds (Sec. 4.1). (f) Merged label map with missing region in-painted, moving objects and sky.

**Data collection.** We use a mobile LIDAR scanner from *Riegl* to collect point clouds of the static 3D map with high granularity. As shown in Fig. 2(a). The captured point cloud density is much higher than the Velodyne[2] used by KITTI [17]. Different from the sparse Velodyne LIDAR, our mobile scanner utilizes two laser beams to scan vertical circles. As the acquisition vehicle moves, it scans its surroundings as a push-broom camera. However, moving objects, such as vehicles and pedestrians, could be compressed, expanded, or completely missing in the captured point clouds. In order to eliminate these inaccurate moving objects (circled in blue at Fig. 2(b)), we conduct three steps: 1) scan the same road segment multiple rounds; 2) align and fuse those point clouds; 3) remove the points with low temporal consistency. Formally, the condition to kept a point $\mathbf{x}$ in round $j$ is,

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

where $\delta = 0.6$ and $\epsilon_d = 0.025m$ in our setting, and $\mathbb{1}()$ is an indicator function. We keep the remained point clouds as a static background $\mathcal{M}$ for further labelling.

For video capturing, we use two frontal cameras with a resolution of $2048 \times 2432$. The whole system including the LIDAR scanner and cameras is well calibrated.

**2D and 3D labeling** In order to have semantic labelling of each video frame, we handle static background, static objects (*e.g.*parked vehicles that could be well recognized in point clouds), and moving objects separately. Firstly, for static background, we directly perform labelling on the 3D point clouds $\mathcal{M}$ which are then projected to images, yielding labelled background for all the frames. Specifically, we over-segment the point clouds into point clusters based on spatial distances and normal directions, and then label each cluster of points manually. Second, for static objects in each round, we prune out the points of static background, and label the remaining points of the objects. Thirdly, after 3D-2D projection, only moving objects remain unlabeled. Here, we adopt an active labelling strategy, by first training an object segmentation model using a SOTA algorithm [46], and then refining the masks of moving objects manually.

As shown in Fig. 2(c), the labels obtained from above three steps are still not perfect. There are some unlabeled pixels that could be caused by missing points or reflection. We handle such issues by using the splatting technique in computer graphics, which turns each point into a small square as discussed in Sec. 4.1 (Fig. 2(e)). The results are further refined to generate the final labels (Fig. 2(f)). With such a strategy, we can greatly increase labelling efficiency and accuracy for video frames. For example, it could be very labor-intensive to label texture-rich regions like trees and poles, especially when occlusion happens. We provide the labelled video in our supplementary materials for readers who are interested.

Finally, due to space limitation, we elaborate the whole acquisition system, data collection and labelling process with an extended dataset paper [22], called ApolloScape, where a larger dataset with more labelled objects is collected and organized after our submission. Nevertheless, in this paper, we experimented with a preliminary version of that dataset, which will be released separately for reproducibility of our results.

## 4. Localizing camera and scene parsing.

As shown in Sec. 1.1, our full system is based on a semantic 3D map and deep networks. In the following, we will first describe how a semantic label map is rendered from the 3D, then discuss the details of our network architectures and the loss functions to train the whole system.

### 4.1. Render a label map from a camera pose.

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.

In our setting, the 3D map is a point cloud based environment. Although the density of the point cloud is very high (one point per 25mm 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. 2(c).

[2]http://www.velodynelidar.com/

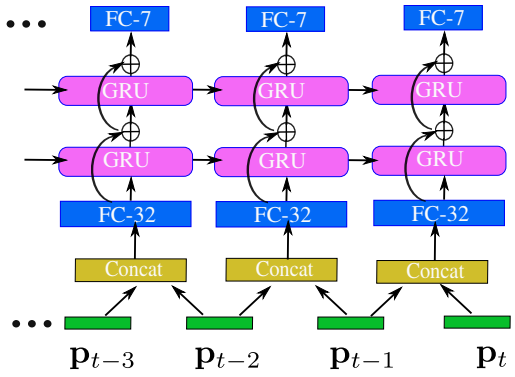



Figure 3: The GRU RNN network architecture for modeling a sequence of camera poses.

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, 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. 2(e), invalid values in-between those projected points are well in-painted, meanwhile the boundaries separating different semantic classes are also well preserved. Later, we insert such a rendered label map for the pose CNN and segment CNN, which guides the network to localize the camera and parse the image.

### 4.2. 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 of camera, 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 $0.05m$ from our 3D semantic map by using only road 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. 1, 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 $\mathbf{p}_i^c$. 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 $\mathbf{p}_i = \mathbf{p}_i^c + \hat{\mathbf{p}}_i$. For the network architecture of pose CNN, we follow the design of DeMoN [41], 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 Sec. 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 [45] is appended. More specifically, we adopt a two layer GRU with 32 hidden states as illustrated in Fig. 3. 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 [23] 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 [24], we use the geometric matching loss for training, which avoids the balancing factor between rotation and translation. Formally, given a 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

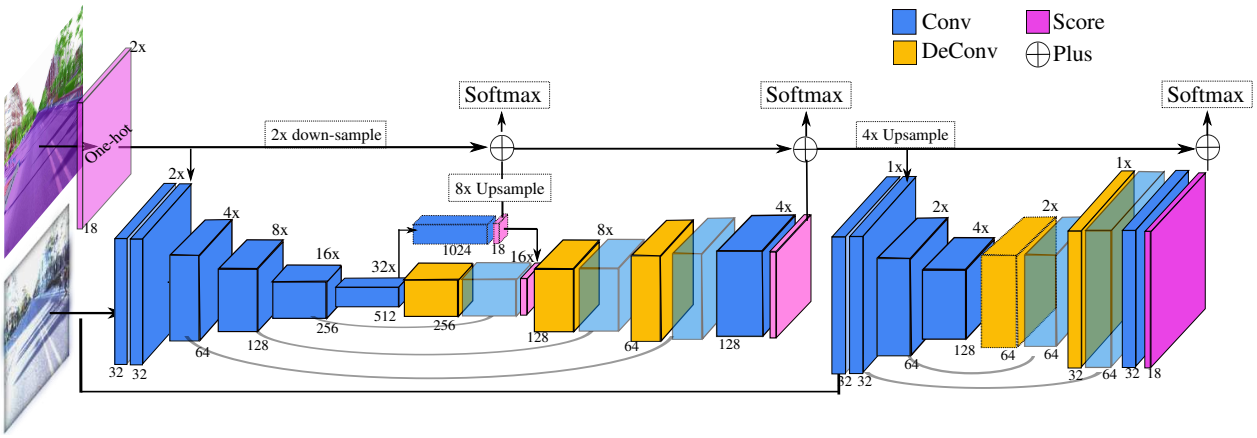



Figure 4: Architecture of the segment CNN with rendered label map as a segmentation priori. At bottom of each convolutional block, we show the filter size, and at top we mark the downsample rates of each block w.r.t. the input image size. The 'softmax' text box indicates the places a loss is calculated. Details are in Sec. 4.3.

achieve better performance. In [24], only the 3D points visible to the current camera are applied to compute this loss to help the stableness 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 $256 \times 304$ using the ground truth camera pose, and use the back projected 3D points from the depth map for training.

### 4.3. 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 Sec. 2, heavily parameterized networks such as ResNet are not efficient enough for our online application. Thus, as illustrated in Fig. 4, 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 [41] 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. 4), which encourages the network to learn the differences between the rendered label map and the ground truth. In [36], 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. 4, 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 [14]. For refinement network shown in Fig. 4, 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% pixel accuracy, leaving only 30% 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. 4.

## 5. Experiments

We perform all experiments using our collected dataset, and evaluate multiple system settings for pose estimation and segmentation to validate each component. For GPS and IMU signal, despite we have multiple scans for the same road segments, it is still very limited for training. Thus, follow [42], we simulate noisy GPS and IMU by adding random perturbation $\epsilon$ w.r.t. the ground truth pose following uniform distributions. Specifically, translation and rotation noise are set as $\epsilon_t \sim U(0, 7.5m)$ and $\epsilon_r \sim U(0°, 15°)$ respectively. We refer to realistic data [30] for setting the noisy range of simulation.

**Datasets.** In this paper, our acquisition vehicle scans two sites at Beijing in China yielding two datasets. 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 $3km$, and the distance between consecutive frames is around $5m$ to $10m$. 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 $1m$ to $3m$. We use 8 rounds of the video camera images for training and 2 for testing, yielding 17062 training images and 1973 testing images. The semantic classes of the two datasets are shown in Tab. 3. We will release the two datasets separately.

**Implementation details.** To quickly render from the 3D map, we adopt OpenGL to efficiently render a label map with the z-buffer handling. A $512 \times 608$ 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 9ms. For pose RNN, we sample sequences with length of 100 from our data for training, and the speed for each frame is 0.9ms on average. For segment CNN, we keep the size the same as input, and the forward time is 90ms. Both of the network is learned with 'Nadam' optimizer [13] 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[3] 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 [7] platform.

For pose evaluation, we use the median translation offset and median relative angle [25]. 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 [46].

**Pose Evaluation.** In Tab. 2, we show the performance of estimated translation $\mathbf{t}$ and rotation $\mathbf{r}$ from different model variations. We first directly follow the work of PoseNet [25, 24], 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, *i.e.* results better than the noisy GPS/IMU signal. 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, the model can learn good relative pose between camera and GPS/IMU, which significantly reduces the error (60% for $\mathbf{t}$, 85% 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.

| 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 |
| | 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 | **1.005** ± 0.044 | **0.719** ± 0.035 | **73.01** ± 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 | **0.890** ± 0.037 | **0.557**± 0.021 | **88.55** ± 0.13 |

Table 2: Compare the accuracy of different settings for pose estimation from the two datasets. 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.

For evaluating an video input, we setup a baseline of performing RNN directly on the GPS/IMU signal, and as shown at 'Pose RNN w/o CNN', the estimated $\mathbf{t}$ is even better than pose CNN, while $\mathbf{r}$ 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 Kalman filter [23] 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 $\mathbf{t}$ and $\mathbf{r}$. We visualize some results at Fig. 5(a-c). Finally at bottom of Tab. 2, we list corresponding results on Dlake dataset, which draws similar conclusion with that from Zpark dataset.

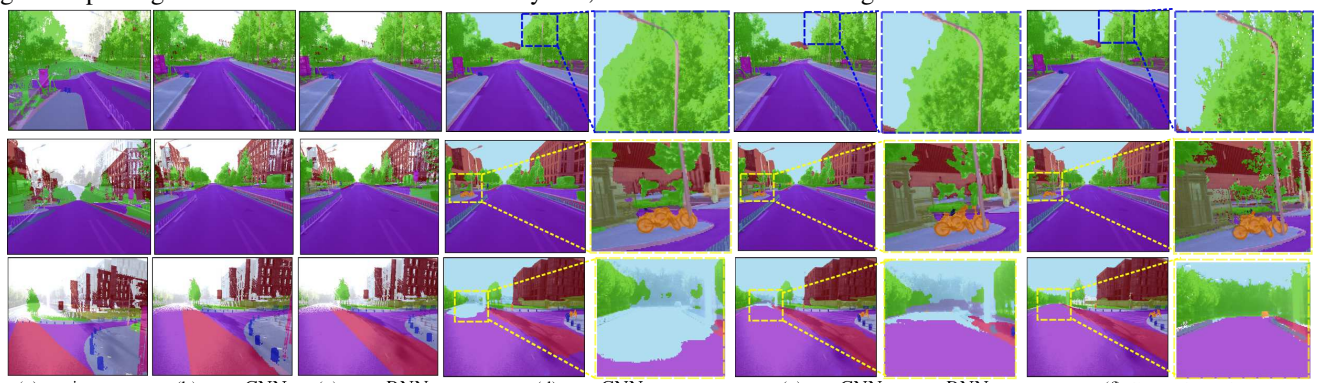
**Segment Evaluation.** At top part of Tab. 3, we show the scene parsing results of Zpark dataset. Firstly, we adopt one of the SOTA parsing network on the CityScapes, *i.e.* ResNet38 [46], and train it with Zpark dataset. It utilizes pre-trained parameters from the CityScapes [9] dataset, and run with a $1.03s$ per-frame with our resolution. As shown

[3]https://github.com/aleju/imgaug

| 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 [46] | 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 |
| | 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/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 |
| | 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 |

Table 3: Compare the accuracy of different segment networks setting over Zpark (top) and Dlake (bottom) dataset. $t$ is short for 'traffic' in the table. Here we drop the 10 times S.D. to save space because it is relatively small. Our results are especially good at parsing of detailed structures and scene layouts, which is visualized in Fig. 5.



(a) noisy pose (b) pose CNN (c) pose RNN (d) seg. CNN (e) seg CNN + pose RNN (f) Ground Truth

Figure 5: 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 available in the supplementary material.

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 from 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. 5, which demonstrates the effectiveness of our strategy. We list the results over Dlake dataset with more object labelling at bottom part of Tab. 3, and here the rendered label provides a background context for object segmentation, which also improve the object parsing performance.

In Fig. 5, 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 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.

## 6. Conclusion

In this paper, we present a deep learning based framework for camera self-localization and scene parsing with a given 3D semantic map for online videos, for the applications of visual-based outdoor robotic navigation. The algorithm fuses multi-sensors, is simple and runs efficient, meanwhile yields strong results in both of the tasks. More importantly, in our system, we show that the two information are mutually beneficial, where pose helps give good priori for segmentation and semantic guides a better localization. To perform the experiments, we created a dataset which contains a point-cloud based semantic 3D map and videos with ground truth camera pose and per-frame semantic labelling.

## References


[1] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*, 2016. 3

[2] G. Bhorkar. A survey of augmented reality navigation. *CoRR*, abs/1708.05006, 2017. 1

[3] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *PR*, 30(2):88–97, 2009. 3

[4] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, pages 3547–3555, 2015. 3

[5] D. Campbell, L. Petersson, L. Kneip, and H. Li. Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. *ICCV*, 2017. 1, 3

[6] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 1, 3

[7] 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*, abs/1512.01274, 2015. 7

[8] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. Vidloc: 6-dof video-clip relocalization. *CVPR*, 2017. 1, 3

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

[10] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. *ICCV*, 2017. 1, 3

[11] P. David, D. Dementhon, R. Duraiswami, and H. Samet. Softposit: Simultaneous pose and correspondence determination. *IJCV*, 59(3):259–284, 2004. 3

[12] 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 *ICCV*, pages 2758–2766, 2015. 3

[13] T. Dozat. Incorporating nesterov momentum into adam. 2016. 7

[14] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 6

[15] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 1, 3

[16] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. *ICCV*, 2017. 3

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 3, 4

[18] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3d scene reconstruction and class segmentation. In *CVPR*, pages 97–104, 2013. 3

[19] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *IJCV*, 13(3):331–356, 1994. 1, 3

[20] F. W. C. Hazirbas, L. L.-T. T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. *ICCV*, 2017. 3

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 1, 3

[22] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The apolloscape dataset for autonomous driving. *arXiv preprint arXiv: 1803.06184*, 2018. 4

[23] R. E. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 5, 7

[24] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. *CVPR*, 2017. 3, 5, 6, 7

[25] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 1, 3, 7

[26] L. Kneip, H. Li, and Y. Seo. Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability. In *ECCV*, 2014. 1, 3

[27] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*, pages 703–718. Springer, 2014. 1, 3

[28] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, pages 3168–3175, 2016. 1, 3

[29] Z. Laskar, I. Melekhov, S. Kalia, and J. Kannala. Camera relocalization by computing pairwise relative poses using convolutional neural network. *CVPR*, 2017. 3

[30] 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. *Sensors*, 15(8):20779–20798, 2015. 6

[31] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose priors for simultaneously solving alignment and correspondence. *ECCV*, pages 405–418, 2008. 1, 3

[32] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1

[33] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: dense tracking and mapping in real-time. In *ICCV*, 2011. 1

[34] K. Ohno, T. Tsubouchi, B. Shigematsu, S. Maeyama, and S. Yuta. Outdoor navigation of a mobile robot between buildings based on dgps and odometry data fusion. In *ICRA*, 2003. 1

[35] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. 3

[36] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *CVPR*, 2017. 6

[37] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *ICCV*, 2017. 3

[38] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 3

[39] 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 *CVPR*, 2017. 3

[40] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. *CVPR*, 2017. 3

[41] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. *CVPR*, 2017. 3, 5, 6

[42] K. Vishal, C. Jawahar, and V. Chari. Accurate localization by fusing images and gps signals. In *CVPRW*, pages 17–24, 2015. 3, 6

[43] 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. *arXiv preprint arXiv:1612.00423*, 2016. 2, 3

[44] C. Wu et al. Visualsfm: A visual structure from motion system. 2011. 1

[45] 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 preprint arXiv:1609.08144*, 2016. 5

[46] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *CoRR*, abs/1611.10080, 2016. 1, 3, 4, 7, 8

[47] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. *CoRR*, abs/1704.08545, 2017. 3

[48] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CVPR*, 2017. 1, 3

[49] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. *CVPR*, 2017. 1, 3