L^3-Net: Towards Learning based LiDAR Localization for Autonomous Driving

Weixin Lu* Yao Zhou* Guowei Wan Shenhua Hou Shiyu Song†
Baidu Autonomous Driving Business Unit (ADU)
{luweixin, zhouyao, wanguowei, houshenhua, songshiyu}@baidu.com

Abstract

We present L^3-Net - a novel learning-based LiDAR localization system that achieves centimeter-level localization accuracy, comparable to prior state-of-the-art systems with hand-crafted pipelines. Rather than relying on these hand-crafted modules, we innovatively implement the use of various deep neural network structures to establish a learning-based approach. L^3-Net learns local descriptors specifically optimized for matching in different real-world driving scenarios. 3D convolutions over a cost volume built in the solution space significantly boosts the localization accuracy. RNNs are demonstrated to be effective in modeling the vehicle's dynamics, yielding better temporal smoothness and accuracy. We comprehensively validate the effectiveness of our approach using freshly collected datasets. Multiple trials of repetitive data collection over the same road and areas make our dataset ideal for testing localization systems. The SunnyvaleBigLoop sequences, with a year's time interval between the collected mapping and testing data, made it quite challenging, but the low localization error of our method in these datasets demonstrates its maturity for real industrial implementation.

1. Introduction

In recent years, autonomous driving has gained substantial interest in both academia and in the industry. A precise and reliable localization module that estimates the position and orientation of the autonomous vehicle is highly critical. In an ideal situation, the positioning accuracy has to be specific to the level of centimeters and reach sub-degree attitude accuracy universally. In the last decade, several methods have been proposed to achieve this goal with the help of 3D light detection and ranging (LiDAR) scanners [19, 20, 18, 36, 37, 17, 9, 32]. A classic localization pipeline typically involves several steps with certain variations as shown in Figure 1. They are a feature representation method (e.g., points[2], planes, poles, Gaussian bars over 2D grids [20, 37, 32]), a matching algorithm, an outlier rejection step (optional), a matching cost function, a spatial searching or optimization method (e.g., exhaustive or coarse-to-fine searching, Monte Carlo sampling or iterative gradient-descent minimization) and a temporal optimization or filtering framework. Although some of them have shown excellent performance in terms of the accuracy and robustness across different scenarios, it usually requires significant engineering effort in tuning each module in the pipeline and designing the hard-coded featuring and matching methods. Moreover, these handcrafted systems typically have strong preferences over the running scenarios. Making a universal localization system that is adaptive to all challenging scenarios requires tremendous engineering efforts, which is not feasible.

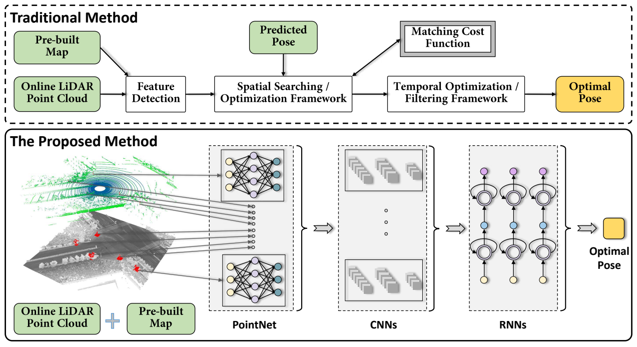

Figure 1: Architectures of the traditional and the proposed learning based method. In our method, L^3-Net takes online LiDAR scans, a pre-built map and a predicted pose as inputs, learns features by PointNet, constructs cost volumes over the solution space, applies CNNs and RNNs to estimate the optimal pose.

The learning based method opens a completely new window to solving the above problems in a data-driven fashion. The good part is that it's easy to collect a huge amount of training data for a localization task because the ground truth trajectory typically can be obtained automatically or semi-automatically using offline methods. In this way, human labeling efforts can be minimized making it more attractive and more cost-effective. Thus, localization or other 3D

*Authors with equal contributions
†Author to whom correspondence should be addressed

related geometry problems are naturally well suited to be solved using the data-driven techniques. Conversely, learning based methods have shown excellent performance in redundant tasks, such as classification, detection or segmentation. These examples have demonstrated that deep neural networks are very efficient for understanding semantics. But this has not been the case for tasks related to the 3D geometry, for example, the localization problem.

In this paper, we propose a deep neural network architecture to accurately estimate the vehicle's position and orientation using LiDAR scans. Modules in the traditional handcrafted pipelines are replaced with deep neural networks. We first extract a set of keypoints evaluated by their linearities and scattering defined with the eigenvalues of the neighbors of a 3D point [34]. Inspired by [7], a single mini-PointNet [27] is used to extract feature descriptors. They encode certain statistical properties of the points and are trained to optimize the matching robustness especially in different scenarios through our network architecture. The key design of our network is to significantly improve the localization accuracy, yielding comparable results to handcrafted pipelines, in a fully differentiable cost volume over $x \times y \times yaw$ dimensions regularized by 3D convolutions inspired by recent learning based stereo [16]. Finally, we calculate the matching probabilities of all the dimensions and obtain the optimal estimation. The temporal motion dynamics, which are typically modeled by filtering methods, such as a particle filter, are implicitly encapsulated by deep Recurrent Neural Networks (RNNs).

To summarize, our main contributions are:

- To the best of our knowledge, this is the first learning based LiDAR localization framework for autonomous driving that directly processes point clouds and accurately estimates the vehicle's position and orientation, yielding comparable results to prior state-of-the-art handcrafted ones.
- Novel use of 3D convolutions for learning how to regularize the cost volume over $x \times y \times yaw$ dimensions which boosts the localization accuracy.
- Rigorous tests in various urban roads together with the release of a dataset including more than 380km real traffic driving and multiple trials traveling at different times on the same roads, that is well suitable for the localization task.

2. Related Work

LiDAR-based localization for autonomous vehicles has been under study for quite some time because of its precision and reliability compared to other sensors. In this paper we summarize related work. Note that the generic point cloud registration methods potentially available for our application are beyond the scope of this paper.

2.1. Methods based on Geometry

Traditional LiDAR localization methods are based on geometry. They rely heavily on geometric constraints to estimate the motion of the vehicle. The most straightforward method is to apply point-cloud registration algorithms to solve the motion problem. Iterative Closest Point (ICP) [2], G-ICP [30] and Normal Distributions Transform (NDT) [3] are common registration algorithms that can be considered. K. Yoneda and S. Mita [40] localized their vehicle by using ICP to align their online point cloud to the pre-recorded point cloud map. Another example that can be considered is, S. Kato et al. [13] provided an open-source platform, named Autoware. It provides a rich set of self-driving modules including localization where registration methods, such as ICP and NDT are supported. However, it is known that these registration methods are very sensitive to the initial guess. They fail in scenes without abundant 3D features, such as highways or other open spaces.

A more pervasive strategy is to utilize the LiDAR intensity and limit the solution space to only three degrees of freedom (x, y, and yaw) = (x, y, ψ) as other elements (z, roll and pitch) = (z, ϕ, θ) can be estimated from an inertial measurement unit (IMU) and a digital elevation model (DEM) of the ground plane. J. Levinson and S. Thrun [19, 20] propose a LiDAR intensity-based localization method. LiDAR intensity provides more texture information of the environment, for example, the road lane markers, as valuable additional cues compared to the system solely based on the geometry information of the point cloud. Several recent works [36, 37, 17, 32] combine both intensity and altitude cues to achieve more robust and accurate results. G. Wan and S. Song [32] achieve 5-7cm RMS horizontal and longitudinal accuracy by adaptively fusing the intensity and altitude cues. The system has been demonstrated to be more robust to environmental changes, such as road construction. R. Wolcott and R. Eustice [36, 37] uses a Gaussian mixture map that captures both the intensity and the altitude. The system performs well in severe weathers, such as snow.

Some works [5, 23] localize the vehicle with low-end 2D LiDARs. But as the retail price of 3D LiDAR scanners keeps falling, the cost advantage that current 2D LiDARs have over 3D will no longer exist. Other works [24, 29, 35] focus more on Advanced Driver-assistance Systems (ADAS) with Ibeo 3D LiDARs, but those are currently beyond the scope of this paper.

2.2. Methods based on Learning

Deep learning is a machine learning technique inspired by the structure and function of the human brain. It has shown excellent performance in semantic tasks, for example, detection, classification or segmentation. However, they typically are not considered as effective approaches to geometric problems as humans are not good at accurate distance measuring without the assistance of tools, either.

To the our best knowledge, there are few prior existing ways of tackling these problems. But none of them have achieved state-of-the-art performance in terms of accuracy, compared to a well-designed handcrafted pipeline. PoseNet [15] and its variants [14, 25, 31] attempt to solve the visual re-localization problem, in which an accurate solution is not the goal. Improvements [6, 4] over PoseNet incorporate relative geometric constraints between frames or integrate temporal information through a bidirectional LSTM. The pose trajectories are smoothed and localization errors are reduced, but they still do not meet the needs of autonomous driving applications yet. DeLS-3D [33] applies the network architecture of PoseNet to solve the visual localization problem, but the translational error is about 0.9-1.3m. Some existing methods [41, 7, 10] could also be applied to solve the re-localization problem with point clouds. But, in order to obtain accurate matching results, methods, such as ICP, are still necessary for registration refinement. [39] applies a semi-handcrafted deep neural network, LocNet, to globally re-localize the vehicle and ICP registration is again used to obtain accurate localization results. Most recently, I. Barsan et al. [1] proposed a learning based localization method, using LiDAR intensity images similar to [19, 20, 32]. Compared to processing point clouds directly, it arguably loses important information that potentially could be learned and encoded by neural networks. In this paper, we propose a novel learning based LiDAR localization system that processes point clouds directly. It has the capability to match the performance of the state-of-the-art handcrafted localization pipeline.

3. Problem Statement

We have designed a deep learning framework for LiDAR-based localization that consumes an online LiDAR point cloud and a pre-built 3D point cloud map. The online LiDAR point cloud can be a single or several consecutive frames from a LiDAR device that is mounted on a vehicle, accumulated from multiple LiDAR scans taking motion compensation into consideration. It is represented as a set of 3D points $\{P_i | i = 1, ..., n\}$, where each point P_i is a vector of (x, y, z, r) including its coordinates and reflection intensity in the local vehicle or LiDAR coordinate system. The pre-built 3D point cloud map is a collection of LiDAR points with global coordinates collected by surveying or mapping vehicles. For better storage efficiency, the 3D point cloud map is down-sampled using a voxel grid filter. Furthermore, we perform semantic segmentation using PointNet++ [28] to remove dynamic objects like vehicles, bicycles, pedestrians, etc., in the point cloud map.

Besides the online point cloud and the pre-built map, the input to our localization framework also includes a predicted pose usually generated by an inertial measurement unit (IMU), or the vehicle dynamics (motion model). It measures the incremental motion between consecutive LiDAR frames. Therefore, the task is to seek an optimal offset between the final and predicted poses by minimizing the matching cost between the online point cloud and the 3D map. For better efficiency and robustness, we follow state-of-the-art localization systems, and estimate the 2D horizontal and heading offset (Δx, Δy, $\Delta \psi$) only.

4. L^3-Net

This section describes the architecture of the proposed network designed for the Learning based LiDAR Localization problem in detail, the so-called L^3-Net, shown in Figure 2.

4.1. Keypoint Features

The first step is extracting local feature descriptors from a set of local patches, which we call keypoints. The selection of keypoints considers several local and global geometry characteristics from different aspects. After the keypoint selection, feature descriptors are extracted using a mini-version of PointNet [27].

Keypoint Selection Given online LiDAR point cloud, we extract a fixed number of keypoints considering some factors including density, geometric characteristic, and distribution. Firstly, we traverse all the points and find the candidates with enough point density in their neighborhood. Secondly, we then evaluate the linearity and scattering of each candidate keypoint using the well-known 3D structure tensor [34]. Features with strong linear and scattering structures are considered to be suitable for the localization task because of their uniqueness and richness in common road scenes. Thirdly, we sort the candidate keypoints by their combinatorially geometric characteristic taking their linearity and scattering into consideration. From the most significant to the least, we try to select a minimum number of keypoints, and confirm that the newly selected keypoints has maintained enough distance from the existing ones. The parameters and thresholds of the implementation are discussed in-detail in Section 5.1.

Descriptor Extraction Once all the qualified keypoints have been selected, we extract meaningful feature descriptors for them. Conventionally, simple geometric or statistical features are used to describe the similarity between point clouds using features learned by deep networks. In the proposed method, we extract feature descriptors by applying PointNet [27], which is a pioneer work addressing the issue of consuming unordered points in a network architecture.

For each keypoint, we collect 64 neighboring points. For each neighboring point, the relative coordinate to keypoint and its reflection intensity (x, y, z, r) is used for descriptor

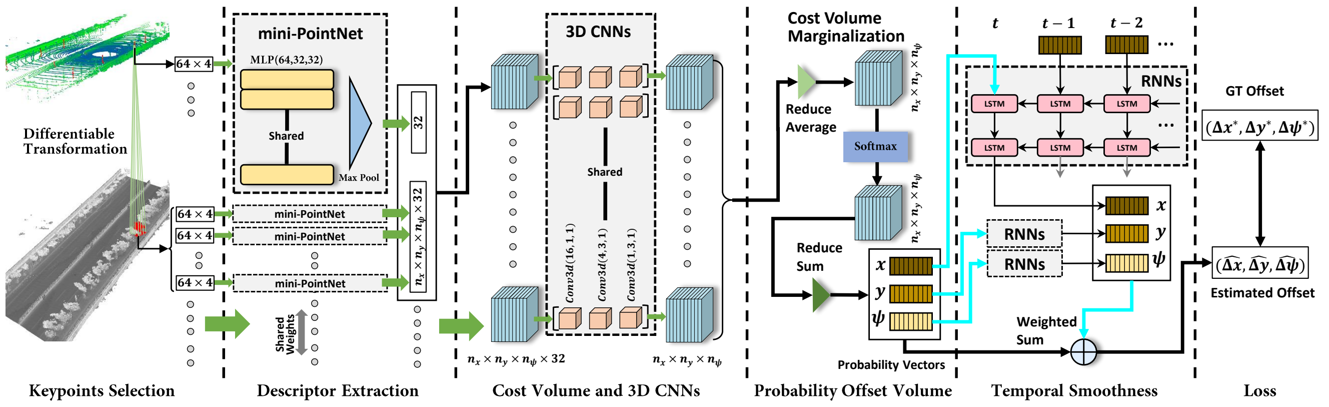

Figure 2: Architecture of the proposed Learning-based LiDAR Localization Network, L^3-Net. During the first training stage, only the black arrows are involved, which include the keypoint selection, the mini-PointNet feature extraction and the regularization based on the 3D CNNs. The cyan arrows indicate the second training stage where the temporal smoothness based on the RNNs is added.

extraction. Therefore, the input of the mini-PointNet network is a 64×4 tensor, and the output is a 32-dimensional vector representing local feature of the keypoint patch. To be more specific, the mini-version PointNet as shown in Figure. 2 includes: a multi-layer perceptron (MLP) of 3 stacking fully connected layers and a max-pooling layer to aggregate and obtain the feature descriptor. We use a parameter-shared mini-PointNet structure for feature extraction of both the online point cloud and the offline map.

4.2. Cost Volume and 3D CNNs

The next step is building a network to accurately infer the localization offset (Δx, Δy, $\Delta\psi$). This is done by constructing a cost volume in the solution space (x, y, ψ), and regularize it with 3D convolutional neural networks (3D CNNs). First of all, we divide the solution space into discrete spaces in x, y and ψ dimensions, and denote n_x, n_y, n_ψ as the size in each dimension. In the following, we denote $\{f_1, ..., f_N\}$ as the keypoint descriptors of the online LiDAR point cloud. Therefore, the cost volume is $N \times n_x \times n_y \times n_\psi$. Each cell represents the matching cost between the corresponding keypoint and the 3D map point with the given offset.

Differentiable Transformation Given the predicted pose, all the local keypoints of the online point cloud are transformed to their global coordinates. Then, we divide the neighborhood of the predicted pose in x, y and yaw dimensions, denoted as $\{(\Delta x_i, \Delta y_j, \Delta\psi_k) | 1 \le i \le n_x, 1 \le j \le n_y, 1 \le k \le n_\psi\}$. The corresponding coordinates in the 3D map can be computed using a transform expressed by a 2×2 rotation matrix and a $2d$ translation vector:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\Delta\psi_k & -\sin\Delta\psi_k \\ \sin\Delta\psi_k & \cos\Delta\psi_k \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x_i \\ \Delta y_j \end{pmatrix}. \quad (1)$$

Then again, the neighboring points of the computed corresponding coordinates in the 3D map are used to extract its feature descriptor through a mini-PointNet structure. Every cell in the cost volume is related to an original keypoint from the online point cloud with its feature descriptor, a transformation, and also a corresponding feature descriptor from the map. Furthermore, a bilinear interpolation filter is also applied to refine the corresponding feature descriptor from the map with its four neighbors in x and y dimensions. As a core step to bridge the keypoint features and the regularization network, the transformation, and the bilinear interpolation are differentiable, enabling the feature learning in the mini-PointNet structure through backpropagation during the training stages. With the descriptor pair from the online point cloud and the map, we can form a $N \times n_x \times n_y \times n_\psi$ volume in the offset solution space, by computing the metric distance between them, which is the input of the regularization network. Metric distance is a 32-dimensional vector, in which each element is calculated by $L2$ distance from the corresponding one in the descriptor pair.

Regularization Given the above input, we hope to learn a regularization function which is able to take into account the context in this volume and refine matching costs. The matching costs in the offset space are calculated independently for each keypoint, so they can never be perfect, even if they were using deep feature representations.

Inspired by recent learning based stereo methods [16, 38, 21, 11], we apply 3D convolutions for volume regularization. In Section 6.3, we show the effectiveness of the 3D CNNs and how they help improve the localization precision significantly. Our 3D CNNs consists of three layers; The first two 3D convolutional layers use ReLU unit and batch normalization where a batch includes all keypoints from a single frame. The last convolutional layer directly sends its output, omitting the normalization and activation operations. 3D CNNs is performed on each $n_x \times n_y \times n_\psi$ subvolumes and they share the same parameters, which significantly increases the speed of convergence and effectively avoids over-fitting.

4.3. Probability Offset Volume

In Section 4.2, we calculate the matching costs of all offset configurations $\{\Delta x_i, \Delta y_j, \Delta\psi_k\}$ for each keypoint independently. In this section, we introduce a probability offset volume to represent the consensus of all keypoints in the offset space, which is a $n_x \times n_y \times n_\psi$ volume. It represents the overall matching cost between the online point cloud and the 3D map given the offset.

Marginalization Suppose that all keypoints are independent of each other, then the matching probability of an offset $\Delta T = (\Delta x_i, \Delta y_j, \Delta\psi_k)$ can be calculated by the following: $\prod_{i=1}^{N} P_i(\Delta T)$, where $P_i(\Delta T)$ represents the matching probability of i-th keypoint at offset ΔT.

Because the product can easily cause overflow, the above equation is converted into log-likelihood:

$$C(\Delta T) \propto \log(\prod_{i=1}^{N} P_i(\Delta T)) = \sum_{i=1}^{N} (\log(P_i(\Delta T)), \qquad (2)$$

where $C(\Delta T)$ represents the overall matching cost at offset ΔT between the online point cloud and the 3D map.

In our implementation, we take the above cost $\log(P_i(\Delta T))$ as input, and then marginalize it into a $n_x \times n_y \times n_\psi$ cost volume across the keypoint dimension by applying a $reduce\ average$ operation, which corresponds to the overall matching costs $C(\Delta T)$.

Probability The value of each cell in the marginalized cost volume is the overall matching cost of the corresponding offset. We apply the $softmax$ operation along x, y, and yaw dimensions to convert the matching costs $C(\Delta T)$ into normalized values, interpreted as probabilities $P(\Delta T)$. In Section 6.3, we visualize the distributions of the matching cost and the probability offset volume in x-y dimensions with given yaws. Finally, we marginalize the probability offset volume $P(\Delta T)$ into probability vectors across x, y, and ψ dimensions by applying a $reduce\ sum$ operation: $P_i(\Delta x_i) = \sum_{y,\psi} P(\Delta T)$, $P_j(\Delta y_j) = \sum_{x,\psi} P(\Delta T)$ and $P_k(\Delta\psi_k) = \sum_{x,y} P(\Delta T)$.

4.4. Temporal Smoothness

The above sections introduce the spatial matching between the online point cloud and the map. The probability offset volumes of sequential frames are therefore independent of each other. However, the localization task is a sequential process, so the poses of sequential frames should be considered jointly. In traditional methods [19, 20, 32], the historical distributions within the histogram filter are propagated to estimate the current matching distribution, which ensures the temporal smoothness of the output. Following this spirit, we introduce the recurrent neural networks (RNNs) to achieve similar temporal smoothness. To be more specific, we use LSTM [8] in our network as shown in Figure 2. The probability vector of each dimension (x, y, ψ) from the probability offset volume is treated as the input of each parameter independent RNNs unit. Through learning of historical information by RNNs, the trajectory of localization result is smoother and more accurate as shown in Table 3 of Sec. 6.

4.5. Loss

Unlike prior work [41, 7] using the feature space distance as a loss, we directly define the loss as squared L2 distance between the estimated offset $\hat{\Delta T} = (\hat{\Delta x}, \hat{\Delta y}, \hat{\Delta \psi})$ and the ground truth $\Delta T^* = (\Delta x^*, \Delta y^*, \Delta\psi^*)$. The estimated offset can be calculated by:

$$\hat{\Delta T} = (\sum_{i=1}^{n_x} P_i(\Delta x_i)\cdot\Delta x_i, \sum_{j=1}^{n_y} P_j(\Delta y_j)\cdot\Delta y_j, \sum_{k=1}^{n_\psi} P_k(\Delta\psi_k)\cdot\Delta\psi_k) \qquad (3)$$

The loss function is defined as below:

$$Loss = \alpha\cdot(\|\hat{\Delta x} - \Delta x^*\|^2 + \|\hat{\Delta y} - \Delta y^*\|^2) + \|\hat{\Delta\psi} - \Delta\psi^*\|^2, \qquad (4)$$

where α is the balancing factor.

5. Implementation Details

5.1. Hyperparameters

During the keypoint selection, we choose 128 keypoints within a frame of the LiDAR point cloud. The solution space of the cost volume is set as $11 \times 11 \times 11$, and the steps in x, y and ψ dimensions are 0.25m, 0.25m and $0.5°$, respectively. Therefore, the maximum affordable offset of the predicted pose is about $(0.25 \times \frac{11-1}{2} = 1.25$m, 1.25m, $2.5°)$ which is sufficient for our application. In our implementation, the mini-PointNet structure is $64 \times 32 \times 32$ MLP, 3D CNNs is Conv3d $(16, 1, 1)$ - Conv3d $(4, 3, 1)$ - Conv3d $(1, 3, 1)$, and RNNs is a two layer LSTM with 11 hidden states.

5.2. Training

We adopt a 2-step strategy during the training stage. In the first step, we only train the mini-PointNet structure and the 3D CNNs. In order to achieve this, we first remove RNNs in the network architecture and calculate the loss directly from the probability vectors inferred from the probability offset volume. The batch size and the learning rate are set to be 1 and 0.01, respectively. In order to make the extracted features more robust, we add a uniformly distributed random noise of $[0 \sim 1.0]m$ in x-y dimension, and a random error of $[0 \sim 2.0]^{\circ}$ in yaw dimension to the input predicted pose. In the second step, we train the parameters of RNNs with those fixed in the mini-PointNet structure and the 3D CNNs. The batch size and the learning rate are set to be 1 and 0.001, respectively. We sample the sequences with a length of 10 during the RNNs training. Given that the frequency of LiDAR frames is 10hz, the actual receptive field of RNNs is about 1.0 second. In these two steps, we randomly divide the dataset into the training and validation set, yielding the ratio of training to validation as 4 to 1. We decide to stop at 100 epochs for these two steps when there is no performance gain.

6. Experiments

6.1. The Apollo-SouthBay Dataset

The common sensor in a LiDAR-based localization system for autonomous driving applications is a 360° 3D LiDAR. To build the map and test the system, we also need multiple trials of data collection on the same road. To the best of our knowledge, no public datasets meet this requirement, as summarized in Table 1. Therefore, we needed to collect data using our own vehicle by driving through different areas in southern San Francisco Bay Area covering different scenarios including but not limited to residential areas, urban downtown areas, and highways, and build a new dataset, Apollo-SouthBay Dataset. We equipped a standard Lincoln MKZ sedan with a Velodyne HDL-64E LiDAR, and an integrated navigation system for data collection. We have collected data for multiple trials over a period of days, weeks, or even a year to meet the needs of mapping, training and testing the vehicle. And this historical data is what is currently being used for map building and testing for a localization system for autonomous driving applications. We currently use the high-end integrated navigation system, NovAtel ProPak6, a triple-frequency GNSS RTK receiver, together with the IMU-ISA-100C, a near navigation-grade IMU. The GNSS RTK/INS integrated solution using post-processing software, such as the NovAtel Inertial Explorer, is employed as the ground truth. In total, our dataset covers a driving distance of 380.5 km and contains a set of $506,679$ LiDAR frames with high-quality ground truth. The Apollo-SouthBay Dataset is to be released soon.

6.2. Performance

Training and Testing Setup Our dataset involves six different routes as shown in Table 2. In the first five of these routes, BaylandsToSeafood, $\cdots$, SanJoseDowntown, the collection time interval between the mapping/training and testing data is approximately a week. The sixth one, SunnyvaleBigLoop, which is the longest and covers different scenarios including residential areas, urban roads, and the highway, is excluded from the training dataset deliberately. Please note that SunnyvaleBigLoop has a wide time interval of as long as a year between the mapping and test data collection, which is very challenging for localization. The mapping procedure requires multiple repetitions of data collection to ensure good data density depending on the vehicle's speed. That's the reason why the data length of the mapping dataset is often longer than the others.

Datasets	Length	Ground Truth	360° LiDAR	Multiple Trials
Ford Campus[26]	5.1km	✓	✓	×
KITTI[12]	39.2km	✓	✓	×
Oxford RobotCar[22]	1000.0km	✓	×	✓
Ours	380.5km	✓	✓	✓

Table 1: Our dataset compared to other available related datasets. As seen above, only our dataset fully meets the requirement for map building and testing for a localization system for autonomous driving applications.

Route	Mapping		Training		Testing	
	Dist.(km)	Fram.	Dist.(km)	Fram.	Dist.(km)	Fram.
BaylandsToSeafood	24.91	36,304	4.15	5,551	5.73	6,445
ColumbiaPark	44.83	69,552	13.9	19,705	8.55	13,685
Highway237	29.51	19,625	4.82	2,057	4.34	1,717
MathildaAVE	40.99	50,638	8.83	10,596	9.07	9,483
SanJoseDowntown	23.23	59,774	5.69	14,849	6.17	16,591
SunnyvaleBigLoop	108.1	128,937	-	-	37.7	41,170

Table 2: Routes and their usage for mapping, training and testing purposes. The time interval between the mapping, training and testing data is approximately one year (SunnyvaleBigLoop) and a week (others).

Quantitative Analysis Our proposed learning-based localization system L^3-Net has been extensively tested in real-world driving scenarios. The localization performance is compared against several state-of-the-art LiDAR-based localization methods, such as Levinson et al. [20], and Wan et al. [32]. The pre-built map resolutions of Levinson et al., Wan et al. and ours are all 12.5cm. The input predicted poses are generated from the built-in tightly-coupled GNSS/IMU integrated solution in NovAtel with the RTK disabled, which is the same as the usage of the LiDAR localization module in [20], although currently, the multi-sensor fusion system is not our focus in this paper. The 2-Systems mode is used in [32] since our focus is on the LiDAR-based localization task.

In Table 3, we give a quantitative analysis of each available method. It further demonstrates that the localization performance of our learning based L^3-Net is comparable to the state-of-the-art handcrafted method [32] for real-world driving scenarios. In addition, note our vast performance improvement over [20]. The low localization error of our system in SunnyvaleBigLoop demonstrates that our net-

Route	Methods	Horiz. RMS	Horiz. Max	Long. RMS	Lat. RMS	< 0.1m Pct.	< 0.2m Pct.	< 0.3m Pct.	Yaw. RMS	Yaw. Max	< 0.1° Pct.	< 0.3° Pct.	< 0.6° Pct.
BaylandsToSeafood	Levinson et al.[20]	0.148	1.501	0.115	0.074	54.62%	82.41%	91.10%	-	-	-	-	-
	Wan et al.[32]	**0.036**	**0.203**	**0.026**	**0.019**	**98.88%**	**99.98%**	100.0%	0.054	0.372	86.82%	99.86%	100.0%
	Ours.(WithoutRNN)	0.054	0.328	0.041	0.026	94.49%	99.77%	99.95%	0.029	0.294	98.56%	100.0%	100.0%
	Ours.(WithRNN)	0.050	0.209	0.039	0.024	96.48%	99.89%	100.0%	**0.020**	**0.179**	**99.35%**	100.0%	100.0%
ColumbiaPark	Levinson et al.[20]	0.063	0.202	0.045	0.034	87.30%	99.99%	100.0%	-	-	-	-	-
	Wan et al.[32]	0.046	0.160	0.034	0.024	96.46%	100.0%	100.0%	0.081	0.384	67.27%	99.74%	100.0%
	Ours.(WithoutRNN)	0.047	0.161	0.034	0.025	95.82%	100.0%	100.0%	0.049	0.322	92.57%	99.99%	100.0%
	Ours.(WithRNN)	**0.043**	**0.159**	**0.032**	**0.023**	**98.02%**	100.0%	100.0%	**0.028**	**0.190**	**99.50%**	**100.0%**	100.0%
Highway237	Levinson et al.[20]	0.161	0.622	0.138	0.061	37.05%	69.90%	86.09%	-	-	-	-	-
	Wan et al.[32]	0.049	0.196	0.038	0.022	93.27%	100.0%	100.0%	0.069	0.302	78.12%	99.94%	100.0%
	Ours.(WithoutRNN)	0.053	0.257	0.046	**0.019**	92.05%	99.77%	100.0%	0.048	0.211	94.51%	100.0%	100.0%
	Ours.(WithRNN)	**0.045**	**0.190**	**0.034**	0.023	**99.01%**	100.0%	100.0%	**0.038**	**0.112**	**99.30%**	100.0%	100.0%
MathildaAVE	Levinson et al.[20]	0.106	0.779	0.086	0.044	65.20%	90.43%	94.83%	-	-	-	-	-
	Wan et al.[32]	**0.040**	0.179	**0.030**	**0.020**	98.72%	100.0%	100.0%	0.060	0.453	82.91%	99.74%	100.0%
	Ours.(WithoutRNN)	0.054	0.379	0.040	0.028	96.82%	99.91%	99.99%	0.033	0.674	97.56%	99.83%	99.97%
	Ours.(WithRNN)	0.051	**0.154**	0.040	0.025	**98.87%**	100.0%	100.0%	**0.019**	**0.176**	**99.31%**	**100.0%**	100.0%
SanJoseDowntown	Levinson et al.[20]	0.103	0.586	0.075	0.055	58.20%	88.39%	97.75%	-	-	-	-	-
	Wan et al.[32]	0.058	0.290	0.039	0.034	87.72%	**99.55%**	100.0%	0.052	0.246	87.82%	100.0%	100.0%
	Ours.(WithoutRNN)	0.057	**0.288**	0.037	0.037	89.81%	98.93%	100.0%	**0.033**	0.274	**99.02%**	100.0%	100.0%
	Ours.(WithRNN)	**0.055**	0.294	**0.036**	0.034	**91.32%**	99.20%	100.0%	0.034	**0.221**	98.86%	100.0%	100.0%
SunnyvaleBigLoop	Levinson et al.[20]	0.132	1.423	0.097	0.070	43.95%	87.51%	94.99%	-	-	-	-	-
	Wan et al.[32]	0.069	0.368	0.050	0.038	80.86%	99.08%	**99.96%**	0.081	0.679	69.51%	98.60%	100.0%
	Ours.(WithoutRNN)	0.060	0.451	0.039	0.037	88.24%	98.99%	99.85%	0.046	0.405	91.32%	99.98%	100.0%
	Ours.(WithRNN)	**0.055**	**0.347**	**0.037**	**0.032**	**92.42%**	**99.14%**	99.94%	**0.033**	**0.262**	**96.44%**	**100.0%**	100.0%

Table 3: Comparison with other LiDAR-based localization systems. Note that we nearly match even the state-of-the-art system [32] that has an elaborate handcrafted pipeline. Our wide improvement over other systems [20] is notable.

work can generalize decently in new road scenarios. In most routes, the temporal smoothness by the RNNs gives us better performance, which illustrates its effectiveness.

Methods	Levinson[20]	Wan[32]	Ours
Runtime (ms)	73.2	61.6	121.3
Map Size (MB/km)	10	5	14

Table 4: Runtime performance analysis: to illustrate the processing time and the storage size of the map.

Run-time Analysis We evaluated the runtime performance of our platform with a GTX 1080 Ti GPU, Core i7-9700K CPU, and 16GB Memory as shown in Table 4. It takes 31.0ms, 22.7ms and 67.6ms in the keypoint selection, differential transformation and forward pass steps, respectively. The total end-to end processing time of each frame is 121.3ms, yielding a real-time system.

6.3. Ablations and Visualization

We use the same training and test data introduced in Section 6.2 to better evaluate the proposed network.

Feature Descriptor Comparison We substitute our mini-PointNet structure for PointNet [27] and PPFNet [7] in the descriptor extraction step as shown in Table 5. We note that our mini-PointNet structure outperforms PointNet [27] and PPFNet [7] significantly. In PPFNet, global features are concatenated to local features to introduce the spatial relations of the local features. Intuitively, our input predicted pose is already accurate enough for the network to focus on local matching, therefore the global features are not necessary in our task. Moreover, the descriptor size and the structure of the mini-PointNet in our design are much smaller than PointNet to let them further focus on local feature learning, yielding better performance in localization task rather than semantic tasks, such as classification and segmentation.

Methods	Horiz. RMS	Horiz. Std	<10cm Pct.	Yaw. RMS	Yaw. Std	< 0.1° Pct.
PointNet	0.130	0.278	74.33%	0.12	0.40	81.57%
PPFNet	0.085	0.122	79.68%	0.07	0.17	82.61%
Ours	**0.069**	**0.057**	**84.22%**	**0.06**	**0.05**	**83.29%**

Table 5: Comparison with various network structures. The benefits of our proposed mini-PointNet that focuses more on local features are clearly visible.

Methods	Horiz. RMS	Horiz. Std	<10cm Pct.	Yaw. RMS	Yaw. Std	< 0.3° Pct.
3D CNNs $\times$	0.134	0.072	33.67%	0.089	0.073	65.09%
3D CNNs $\downarrow$	0.137	0.108	37.35%	0.085	0.073	84.03%
3D CNNs $\uparrow$	**0.060**	0.057	**89.48%**	**0.047**	**0.041**	**90.94%**
Ours	0.065	**0.056**	86.49%	0.056	0.048	83.74%

Table 6: Comparison with various 3D CNNs setups. Note that decreasing or removing the CNN layers give much worse results. The importance of 3D CNNs is clear.

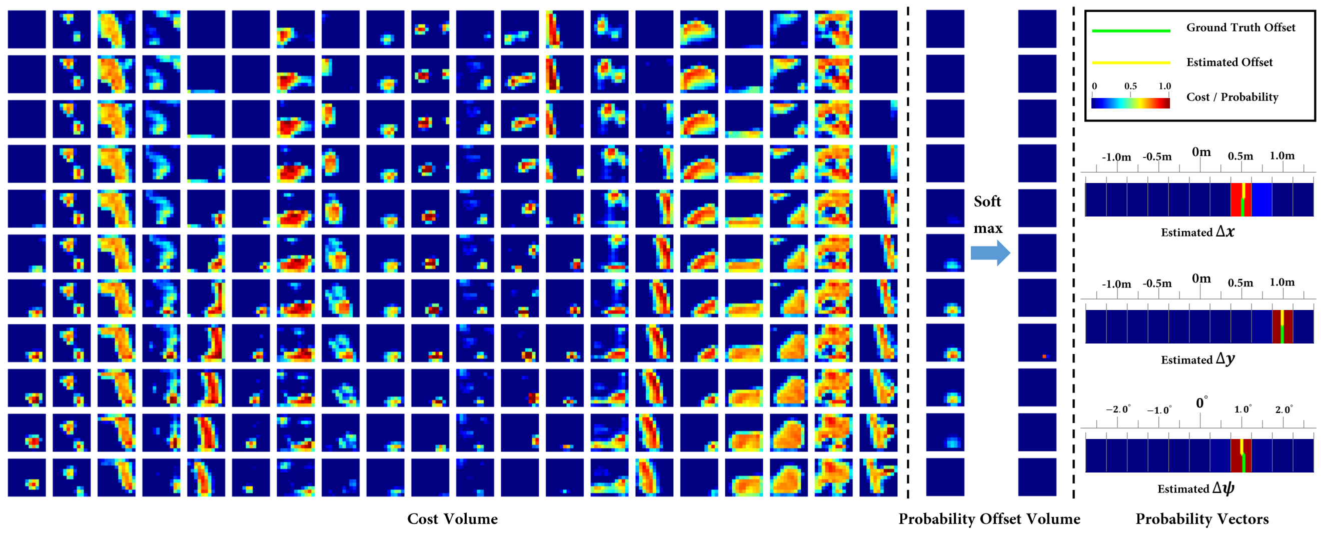

Figure 3: Visualization of network outputs in different stages in Section 4. The cost volume is visualized over $(x,\ y)$ dimensions with fixed yaw values and selected keypoints. The matching response is clearly significant after we accumulate the matching costs from all the keypoints as shown in the probability offset volume in the middle. The final estimated offsets (0.538m, 0.993m, 1.001°) and their ground truths (0.524m, 0.994m, 1.044°) are shown in the right.

3D CNNs In order to verify the importance of 3D CNNs, we conduct the following experiments: removing 3D CNNs, decreasing layers from 3 to 1 and increasing layers from 3 to 4, denoted as 3D CNNs $\times$, 3D CNNs $\downarrow$ and 3D CNNs $\uparrow$. Our results are shown in Table 6, where the localization accuracy drops heavily using 3D CNNs $\times$ and 3D CNNs $\downarrow$. This shows that 3D CNNs can learn the $real$ feature distance and effectively regularize the output in the solution space as compared to directly applying the $L2$ distance between descriptor pairs (No 3D CNNs). Larger-capacity 3D CNNs leads to better localization accuracy, however, more data is also required empirically in training. We use 3 layers as a default setup in our method.

Visualization To have better insights of the mechanism of the network, we visualize the cost volume, the probability offset volume, and the probability vectors discussed in Section 4. In Figure 3, on the left is the cost volume visualized in x, y dimensions given 11 different yaws from 20 out of 128 keypoints after being regularization with 3D CNNs. On the left, marginalized cost volume is displayed, which is obtained from the cost volume of all keypoints, and the probability offset volume after the $softmax$ operation on the central columns of the figure. The right side shows the probability vectors marginalized from the probability offset volume, the estimated offsets, and the ground truth offsets. It is seen that the matching cost estimation from a single keypoint is not reliable due to the insufficient geometric uniqueness over the solution space. However, the matching response in the probability offset volume is absolutely clear after all the matching costs are accumulated from all the keypoints.

7. Conclusion

We have presented a novel learning-based LiDAR localization framework, designed for autonomous driving applications. Elaborately hand-crafted modules in traditional localization pipelines are substituted with learning-based deep neural networks. Our system achieves comparable localization accuracy to prior state-of-the-art systems, and is ready for industrial use. The probability offset volume implies the matching confidence over the solution space making it ready to be deployed in a multi-sensor fusion based localization framework. A 360° 3D LiDAR sensor, a high-end integrated navigation system and the data which includes multiple trials of driving over the same road areas in southern San Francisco Bay make our dataset ideal for benchmarking localization systems.

ACKNOWLEDGMENT

This work is supported by Baidu Autonomous Driving Business Unit (Baidu ADU) in conjunction with the Apollo Project (http://apollo.auto/). We would like to thank our colleagues for their kind help and support throughout the project. Natasha Dsouza helped with the text editing and proof reading. Yong Xiao and Runxin He helped with data collection. Yu Liu processed mapping and testing data.

References

[1] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to localize using a LiDAR intensity map. In *Conference on Robot Learning*, pages 605–616, 2018. 3

[2] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992. 1, 2

[3] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2743–2748, 2003. 2

[4] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-aware learning of maps for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2616–2625, 2018. 3

[5] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1554–1559, May 2013. 2

[6] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. VidLoc: A deep spatio-temporal model for 6-DoF video-clip relocalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 3

[7] H. Deng, T. Birdal, and S. Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 3, 5, 7

[8] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, Apr. 2017. 5

[9] R. Dub, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto. Incremental-segment-based localization in 3-D point clouds. *IEEE Robotics and Automation Letters*, 3(3):1832–1839, July 2018. 1

[10] G. Elbaz, T. Avraham, and A. Fischer. 3D point cloud registration for localization using a deep neural network autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, July 2017. 3

[11] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016. 4

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, June 2012. 6

[13] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, Nov 2015. 2

[14] A. Kendall, R. Cipolla, et al. Geometric loss functions for camera pose regression with deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 8, 2017. 3

[15] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, Dec 2015. 3

[16] A. Kendall, H. Martirosyan, S. Dasgupta, and P. Henry. End-to-end learning of geometry and context for deep stereo regression. In *IEEE International Conference on Computer Vision (ICCV)*, pages 66–75, Oct 2017. 2, 4

[17] H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung. Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area. *IEEE Robotics and Automation Letters*, 2(3):1518–1524, July 2017. 1, 2

[18] R. Kummerle, D. Hahnel, D. Dolgov, S. Thrun, and W. Burgard. Autonomous driving in a multi-level parking structure. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3395–3400, May 2009. 1

[19] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*, volume 4, page 1. Citeseer, 2007. 1, 2, 3, 5

[20] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4372–4378, May 2010. 1, 2, 3, 5, 6, 7

[21] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016. 4

[22] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. 6

[23] W. Maddern, G. Pascoe, and P. Newman. Leveraging experience for large-scale LIDAR localisation in changing cities. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1684–1691, May 2015. 2

[24] R. Matthaei, G. Bagschik, and M. Maurer. Map-relative localization in lane-level maps for ADAS and autonomous driving. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 49–55, June 2014. 2

[25] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Image-based localization using hourglass networks. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 870–877, Oct 2017. 3

[26] G. Pandey, J. R. McBride, and R. M. Eustice. Ford campus vision and LiDAR data set. *International Journal of Robotics Research*, 30(13):1543–1552, 2011. 6

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017. 2, 3, 7

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 3

[29] A. Schlichting and C. Brenner. Localization using automotive laser scanners and local pattern matching. In *IEEE In-*

telligent Vehicles Symposium Proceedings, pages 414–419, June 2014. 2

[30] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Robotics: Science and Systems (RSS)*, volume 2, page 435, 2009. 2

[31] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using LSTMs for structured feature correlation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 627–637, 2017. 3

[32] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4670–4677, May 2018. 1, 2, 3, 5, 6, 7

[33] P. Wang, R. Yang, B. Cao, W. Xu, and Y. Lin. DeLS-3D: Deep localization and segmentation with a 3D semantic map. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[34] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286 – 304, 2015. 2, 3

[35] J. Wiest, H. Deusch, D. Nuss, S. Reuter, M. Fritzsche, and K. Dietmayer. Localization based on region descriptors in grid maps. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 793–799, June 2014. 2

[36] R. W. Wolcott and R. M. Eustice. Fast LIDAR localization using multiresolution gaussian mixture maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2814–2821, May 2015. 1, 2

[37] R. W. Wolcott and R. M. Eustice. Robust LIDAR localization using multiresolution gaussian mixture maps for autonomous driving. *The International Journal of Robotics Research*, 36(3):292–319, 2017. 1, 2

[38] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. MVSNet: Depth inference for unstructured multi-view stereo. *CoRR*, abs/1804.02505, 2018. 4

[39] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong. LocNet: Global localization in 3D point clouds for mobile vehicles. In *IEEE Intelligent Vehicles Symposium Proceedings*, June 2018. 3

[40] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita. LiDAR scan feature for localization with highly precise 3-D map. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 1345–1350, June 2014. 2

[41] A. Zeng, S. Song, M. Niener, M. Fisher, J. Xiao, and T. Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, July 2017. 3, 5