Road Mapping and Localization using Sparse Semantic Visual Features

Wentao Cheng, Sheng Yang, Maomin Zhou, Ziyuan Liu, Yiming Chen, Mingyang Li

***Abstract*—We present a novel method for visual mapping and localization for autonomous vehicles, by extracting, modeling, and optimizing semantic road elements. Specifically, our method integrates cascaded deep models to detect standardized road elements instead of traditional point features, to seek for improved pose accuracy and map representation compactness. To utilize the structural features, we model road lights and signs by their representative deep keypoints for skeleton and boundary, and parameterize lanes via piecewise cubic splines. Based on the road semantic features, we build a complete pipeline for mapping and localization, which includes a) image processing front-end, b) sensor fusion strategies, and c) optimization back-end. Experiments on public datasets and our testing platform have demonstrated the effectiveness and advantages of our method by outperforming traditional approaches.**

I. Introduction

To enable autonomous vehicles to fully navigate across urban environments, one of the key technical component is mapping and localization, which provides high-fidelity pose estimates [1]. In general, the design guidelines for mapping and localization can be summarized as: accurate, robust, and efficient. The first two properties are to ensure the safety of autonomous vehicles under varying environments, and the last one is to allow the low-cost usage of processor, memory, and storage to make autonomous vehicles more affordable.

In this work, we focus on mapping and localization primarily using visual sensors, which are typically of low prices, low power consumptions, and widely utilized in current commercial vehicles. To date, most visual maps consist of 3D distinctive points, computed by probabilistically aggregating and optimizing information from point features detected from input images [2, 3]. However, although point feature based algorithms are widely explored in recent years, they still suffer from the weakness in efficiency and robustness [4]. On one hand, those algorithms typically extract hundreds of point features per image, and compute millions or thousands of millions of points for town-size maps. This requires prohibitive amount of computational resources for both offline computing and online query process [5]. On the other hand, traditional point features might not always be suitable for long-term persistent localization, since they might be difficult to be consistently recognized across different light/weather conditions or might be of high probability of disappearance [6].

Manuscript received: October 15, 2020; Revised: January 15, 2021; Accepted: February 16, 2021.

This paper was recommended for publication by Editor Eric Marchand upon evaluation of the Associate Editor and Reviewers' comments.

All authors are with the Alibaba Group, Hangzhou, China. E-mail: {wentaocheng,shengyang,maomin.zmm,yuan.lzy,yimingchen,mingyangli}@alibaba-inc.com. W. Cheng and S. Yang contributed equally.

Digital Object Identifier (DOI): see top of this page.

To tackle the above problems, one solution is to constrain map elements to be both 'persistent' and 'compact' when building a map. Specifically, map elements should be persistent for *valid* registration, where dynamic elements are filtered, and stable features are retained. Additionally, map elements need to be sparse and frequently observable to achieve compactness, whereas stored targets are highly query-able for *efficient* registration. To achieve those properties, instead of primarily relying on geometrical features as traditional methods do [2, 3, 7], one can consider both geometrical and semantical attributes in feature representation.

In existing mapping and localization methods, high-level feature representations have been designed to enhance the estimation performance, by enriching applicable geometric components of metric maps. Representative methods include the ones that utilize lines [8], planes [9], and boxes [10]. Inspired by recent success in computer vision algorithms for semantic understanding, detecting semantic components in mapping and localization is also under active investigation. [11] proposed a method to encode semantic information via a neural network to conduct end-to-end 6-DoF positioning. However, this method lacks of pose uncertainty guarantees and thus cannot be used directly for applications that require high safety. Alternatively, [12] introduced an algorithm for semantic descriptor learning, which however does not meet the demands of low-cost computing.

In autonomous driving applications, we notice that common roads naturally consist of persistent and compact elements: standardized traffic signs (hung or painted on the ground), lanes, light poles, and so on. Therefore, to facilitate stable and low-cost autonomous driving systems, we propose to detect and parameterize road elements and design a novel semantic mapping and localization pipeline. We view this pipeline the key contribution of this work, which includes:

- A convolutional neural network (CNN) backed image processing front-end to extract semantic features.
- Methods on parameterizing road elements and designing loss functions.
- Semantic optimization modules that can be used for both offline mapping and online localization.

We notice that there exist methods that are conceptually similar to our method, by segmenting road images and selecting points in stable regions, e.g., [13]. However, point features in stable semantic regions might not necessarily be stable and compact, and high level information, e.g., curves, is not utilized. By contrast, our method utilizes multi-source semantic information and provides more compact representation, reaching better 'persistence' and 'compactness'.

II. Semantic Mapping and Localization

A. System Overview

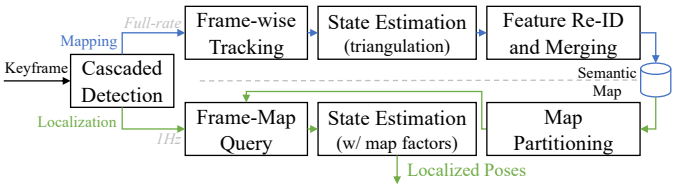


Fig. 1: Overview of the proposed semantic mapping and localization pipeline. The blue arrow connects blocks in mapping stage to build compact maps offline, and the green arrows indicate the data flow of online localization stage.

The backbone of our mapping and localization system is a tightly-coupled state optimization framework with both batch and sliding window strategies [5, 14, 15], as shown in Fig. 1. Specifically, our algorithm proposes to build semantic maps based on standardized road entities offline, and use such maps for online localization. The involved semantic entities include three major types: horizontal objects, ground objects, and lanes (Sec. II-B). Giving a keyframe, the perception module performs cascaded deep detection to extract instances and their representative points as visual features (Sec. II-C).

During offline mapping, the perception module is executed for every keyframe. Detected results between sequential keyframes then are tracked (Sec. II-D), to establish multi-view associations for jointly estimating camera trajectory and landmark poses (Sec. II-F). Subsequently, re-observed entities in previously visited road sections are re-identified and merged through loop detection (Sec. II-G). Finally, these optimized states are serialized as map assets for localization (sec. II-H). During online map-aided localization, the perception module runs at lower frequencies to allow low-cost consumptions on computational units. Therefore, semantic features are obtained via a mixed detection and tracking strategy (Sec. II-I). Those features are matched against the saved maps, and used by a sliding-window optimization based odometry system to reduce global drifts.

B. Selection of Road Features

Considering map sparsification and query effectiveness, the following standardized targets on urban roads are suitable to be detected as semantic landmarks: 1) Lamps and traffic signs on top of the poles beside roads are stable and high enough to be captured by front-facing cameras. 2) Although sometimes occluded by vehicles, the ground area occupies nearly half of each image, making those high contrast signs painted on the ground unneglectable. 3) Similar to ground signs, painted solid and dashed lanes are also frequently observed. Solid lanes provide one directional motion constraints and corners from dashed lanes can be considered as indexed point landmarks. In this work, we choose the above mentioned semantic types as target objects, as visualized in Fig. 2, to build our semantic maps.

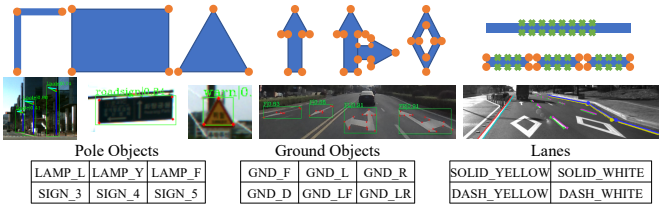


Fig. 2: Semantic entities and their structures defined and used in our maps, with examples of detected instances. Orange: Indexed deep points. Green: Contour sample points.

C. Detection of Road Features

Our two-stage cascaded detection module first performs instance level detection to get instances as *indexed representative pixels* on box (i.e., both pole and ground) objects and *sample pixels* on lane contours. Then, along these detected dash lanes, we evaluate 64×64 image patches to cascadingly detect *indexed dashed lane corners*. To reduce duplicated computations on shareable processes such as feature extraction, we refer to an anchor free detection method CenterNet [16], which separates the low-level feature extraction process from the top-level heads, to make these heads adaptable to different tasks.

Network design. We use the DLA-34 [17] and the DCN module as the backbone for feature extraction. After deconvolution, we get a downsampled feature map for multiple heads H with their loss functions listed in Table I.

1) **Object boxes and keypoints.** Following the human pose estimation task of the CenterNet [16], our model supports a mixed detection of multiple object boxes and keypoints with six proposed heads.
2) **Lanes.** The boundary regression head H_7 outputs the left-right and top-bottom boundary points relative to each foreground point. The output type head H_8 is a three-category classification branch, which determines the left-right or top-bottom (or neither) boundaries of a lane point. We also integrate embedding head H_9 proposed by LaneNet [18] for reducing the dimension of feature representation of each location. These feature representations are later clustered via DBSCAN, and thereby different lane types can be distinguished.
3) **Dashed lane corners.** Given 64×64 image patches, we detect corners in pairs for generating a line segment. To compare with annotations, we add two heads H_{10} and H_{11} to compute angular and length differences, respectively.

D. Feature Tracking for Semantic Entities

Given two consecutively detected frames, the tracking module first accumulates their relative transformation $\mathbf{T}'$ via integrating IMU measurements [19]. For ground objects including signs and lanes, we use the following reprojection equation to map a pixel p onto the other frame as p' as:

$$p' = \pi\left(\mathbf{T}' \cdot \pi'(p, z_p)\right), \text{ with } \mathbf{G} \cdot \pi'(p, z_p) = 0 \qquad (1)$$

where z_p is the depth of pixel p considered valid if positive. $\pi(\cdot)$ and $\pi'(\cdot, z)$ are the camera projection and back-projection operators respectively. We use $\mathbf{G} = [\theta, \phi, d]$

TABLE I: Loss definition and weight w of different heads H proposed for tasks 1) - 3) listed above.

Note	Loss Definition	w	1)	2)	3)
H_1	Center: L1 Focal loss	1.0	√	√	√
H_2	Center: L1 offset by the output stride	1.0	√		√
H_3	Box: L1 loss of box width and height	0.1	√		
H_4	Indexed point: L1 Focal loss	1.0	√		√
H_5	Indexed point: L1 offset by the output stride	1.0	√		√
H_6	Indexed point: L1 offset to the center point	1.0	√		
H_7	Lane: L1 distance to four sub-boundaries	0.2		√	
H_8	Lane: Cross entropy loss for sub-boundary types	1.0		√	
H_9	Lane: Instance embedding loss [18]	1.0		√	
H_{10}	Line Segment: L1 angular difference	1.0			√
H_{11}	Line Segment: L1 length difference	1.0			√

to denote a plane in the camera frame, and $\mathbf{G}\cdot\mathbf{x} \triangleq [\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta]^\top\mathbf{x} + d = 0$. Such coefficients are solved jointly the state estimation as shown in Sec. II-F.

We use the Hungarian matching strategy to associate ground features in both *instance-wise* and *pixel-wise* manners in the pixel space: 1) During *instance-wise* association, we compute the Intersection-over-Union (IoU), on both polygons for regular objects and 5.0 pixel width polylines for lanes. 2) During *pixel-wise* association, we compute the reprojected pixel distance for their indexed keypoints. Matches with IoU percentage $< 50\%$ and pixel distance > 5.0 are ignored.

For keypoints detected in vertical objects such as poles, we use the optical flow method [20] for tracking between frames. During feature tracking, we keep classical keypoints extracted, described, and tracked by the GFTT extractor [21] and the FREAK descriptor [22], since they are not only a part of the visual-inertial odometry, but also stably tracked point features worthy of being included in structured objects. Unlike segmentation that outputs masks, detected 2D boxes might contain GFTT feature keypoints from the background areas, especially in the pole instances. Hence, we perform outlier rejection for these background feature keypoints during the state initialization discussed in Sec. II-F.

E. Representation and Initialization of Road Lanes

We use the piecewise cubic Catmull-Rom spline curves [23] to represent the left and right contour of each 3D lane through a series of control points $\mathbf{C}_k \in \mathbb{R}_3$. Every four continuous control points will determine the shape $\mathbf{C}(t') \in \mathbf{R}_3$, $t' \in [0, 1]$ between two middle points $\mathbf{C}_k$ and $\mathbf{C}_{k+1}$, as:

$$\mathbf{C}(t') = \begin{bmatrix} t'^3 \\ t'^2 \\ t' \\ 1 \end{bmatrix}^\top \begin{bmatrix} -\tau & 2-\tau & \tau-2 & \tau \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 0 & \tau & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}_{k-1} \\ \mathbf{C}_k \\ \mathbf{C}_{k+1} \\ \mathbf{C}_{k+2} \end{bmatrix}^\top, \tag{2}$$

where the tension $\tau = 0.5$ describes how sharply such a curve bends at its control points. For notation simplicity of a piecewise spline curve, we denote $t \triangleq k + t' \in [2, M+1]$ for a whole piecewise curve $\mathbf{C}(t)$ consisting of M segments controlled by $M+2$ points $\mathbf{C}_k, k \in [1, M+2]$. The first and last control points $\mathbf{C}_1$ and $\mathbf{C}_{M+2}$ are off-the-curve for adjusting the direction at its endpoints $\mathbf{C}_2$ and $\mathbf{C}_{M+1}$.

For initializing these control points, we first perform ray-ground intersection as Eq. 1 of its detected sample and corner pixels, to get the point set $\bigcup_{c,j}\{\mathbf{P}_{cj}\}$ from multiple camera observations c. Then, we randomly pick N samples among such a set as $\mathbf{C}_k, k \in [2, N+1]$ to examine the losses:

$$\begin{aligned} &\underset{\mathbf{C}_k}{argmin} \sum_{c,j} \|\mathbf{P}_{cj} - \mathbf{C}(t_{cj})\|^2_{{}^x\Omega} + \sum_{k=3}^{N+1} \| |\mathbf{C}_k - \mathbf{C}_{k-1}| - \lambda_1 \|^2_{{}^y\Omega}, \\ &with \quad \langle \mathbf{C}_1, \beta_1 \rangle \leftarrow \begin{cases} |\mathbf{C}_2 - \mathbf{C}_1| = |\mathbf{C}_3 - \mathbf{C}_2| \\ \beta_1 \cdot \mathbf{N}_2 = \tau(\mathbf{C}_3 - \mathbf{C}_1) \end{cases}, \end{aligned} \tag{3}$$

where the first Mahalanobis term is the 3D point-to-curve residual considering all samples. To calculate the distance, we explicitly solve the parameter t_{cj} through the equation $\mathbf{C}'(t_{cj})(\mathbf{C}(t_{cj}) - \mathbf{P}_{cj}) = 0$ piece by piece. The real roots of the quintic equation stands for possible stationary points, which are then compared with all integers between $[2, N+1]$ to pick up the global minimum point-to-curve residual. The second regularization term ensures these control points to be sampled evenly. We use the parameter $\lambda_1 = 40m$ for controlling density, and ${}^x\Omega = (0.1m)^2\mathbf{I}_3$, ${}^y\Omega = (0.25\cdot\lambda_1)^2\mathbf{I}_1$ for balancing their weights.

The off-the-curve point $\mathbf{C}_1$ (and analogously for $\mathbf{C}_{N+2}$) is initialized by solving the equation of $\langle \mathbf{C}_1, \beta_1 \rangle$ above. $\mathbf{N}_2$ is the 3D normal of $\mathbf{C}_2$ computed by searching its pixel neighborhoods intersected to the ground. Such equations mean to solve possible intersections of a sphere and a line. If we found more than one solutions, we accept the candidate $\mathbf{C}_1$ located at the opposite site of $\mathbf{C}_3$.

For repeatedly picking N sequential samples as middle control points during such a sample-consensus initialization, we first randomly pick one sample, and recursively expand on its two sides by randomly picking another sample whose distance is between $[0.5\lambda_1, 1.5\lambda_1]$. We generate at most 500 random control point sequences for each spline, and evaluate the loss in Eq. 3 to pick up the best sequence. Compared to existing works using B-splines [24] or polybéziers [25, 26], Catmull-Rom splines cross through all middle control points for us to use such a random picking strategy for initialization.

F. State Estimator Design

For clarity, we summarize the indexed notes used in our method in Table. II. To add such semantic entities into a tightly-coupled visual-inertial odometry [19], we introduce five new types of optimizable variables in our system summarized as following:

TABLE II: Multiple indexing notes used in state estimation.

a: ground and pole objects	b: splines c: frames $j \in \{m, n\}$
i: pixels of a as $(\cdot)^c_{a^i}$	k: control points of b as $(\cdot)_{b^k}$
m: samples of b as $(\cdot)^c_{b^m}$	n: corners of b as $(\cdot)^c_{b^n}$

1) The 3D location $\mathbf{P}_{a^i} \in \mathbb{R}_3$ of deep or classical keypoint i within a ground or pole object a. We use the inverse depth parameterization [27] to optimize these points.
2) The ground coefficients $\mathbf{G}$ in the camera coordinate as previously used in Eq. 1. We approximate observable regions of the ground within each frame as a plane. We

use spherical local parameterization[1] to apply updates on its two angles. $\theta \in [0, \pi], \phi \in [0, 2\pi], d \in \mathbb{R}_3$.

3) The vertical plane for a pole object a in the global coordinate denoted as $\mathbf{V}_a(\varphi, e) \cdot \mathbf{x} \triangleq [\cos\varphi, \sin\varphi, 0]^\top \mathbf{x} + e = 0$, whose Z-axis is horizontal with the gravity direction after the visual-inertial odometry is initialized. $\varphi \in [0, 2\pi], e \in \mathbb{R}_3$.
4) The control points $\mathbf{C}_{b^k} \in \mathbb{R}_3$ of a spline b, depicting the left or right contour of a road lane.
5) The dynamic association parameters $t^c_{b^j} \in \mathbb{R}$ for associating pixel $p^c_{b^j}$ detected on a frame c for a spline b. For clarity of notation, we use $j \in \{m, n\}$ for indexing sample and corner pixels, respectively.

Based on the new variables above and the original frame poses $\mathbf{T}_c$ for each image frame c, we add three types of constraints (Fig. 3) based on the detected and tracked semantic features, including:

1) **Points observation factors**. We tend to triangulate and parameterize regular keypoints as previous methods [14], through the following constraint:

$$ {}^1\mathbf{f}^c_{a^i} = \left\| p^c_{a^i} - \pi(\mathbf{T}_c^{-1} \cdot \mathbf{P}_{a^i}) \right\|^2_{{}^1\Omega^c_{a^i}}, \tag{4} $$

where $p^c_{a^i} \in \mathbb{R}_2$ is the location of the detected pixel. ${}^1\Omega^c_{a^i} = \sigma^2 \mathbf{I}_2$ is the noise covariance assigned w.r.t. the precision of detection σ. We use the reported median pixel error in Table III for assigning σ for different types of semantic keypoints, and 1.0 for GFTT keypoint considering its accurate detection.

2) **Spline observation factors**. We use the constraint below to dynamically associate sample and corner pixels $p^c_{b^j}$ as the measurements of control points $\mathbf{C}_{b^k}$ from a spline b:

$$ {}^2\mathbf{f}^c_{b^j} = \left\| p^c_{b^j} - \pi(\mathbf{T}_c^{-1} \cdot \mathbf{C}_b(t^c_{b^j})) \right\|^2_{{}^2\Omega^c_{b^j}}, \tag{5} $$

where $t^c_{b^j}$ is the introduced jointly optimized dynamic association parameter, and ${}^2\Omega^c_{b^j}$ is also set according to the precision of detected samples and corners, respectively.

3) **Coplanar prior factors**. Both vertical and horizontal coplanar priors are added to the optimization through the following form of residuals:

$$ \begin{aligned} {}^3\mathbf{f}_{a^i} &= \| \mathbf{V}_a \cdot \mathbf{P}_{a^i} \|^2_{{}^3\Omega_{a^i}}, \\ or \quad {}^3\mathbf{f}_{\mathbf{x}} &= \left\| \mathbf{G} \cdot (\mathbf{T}_c^{-1} \cdot \mathbf{P}^c_{\mathbf{x}}) \right\|^2_{{}^3\Omega^c_{\mathbf{x}}}, \mathbf{x} \in \{a^i, b^j\}, \end{aligned} \tag{6} $$

where ${}^3\Omega = (0.4m)^2 \mathbf{I}_1$ is assigned according to the thickness or noise of the corresponding planar objects. We assume that these observed lanes are locally planar within each camera view with consistent coefficients.

Initialization of ground and pole objects. We initialize the above optimizable variables sequentially from 1) to 5): For 1) when a GNSS-VIO trajectory is given, we triangulate these feature points from the estimated poses. For 2), we use their contained deep points $\bigcup_i \{\mathbf{P}_{a^i}\}$ to perform a 2D line fitting in the XOY plane. After 2) has been initialized, we use a thickness criterion $|\mathbf{V}_a \cdot \mathbf{P}_{a^i}| < \sigma_1 = 0.3m$ to reject background pixels contained in the detected 2D box. Similarly for 3), we transform each triangulated deep point of the ground to its observed image frame, and then perform 3D plane fitting for $\mathbf{G}$. If there are no ground signs detected for the ground initialization, we instead use the classical feature points within the convex-hull of detected lanes in each frame, and apply a RANSAC 3D plane fitting strategy [29] to remove keypoints on the moving vehicles.

[1]We refer readers to the Plane3D implementation from g2o [28].

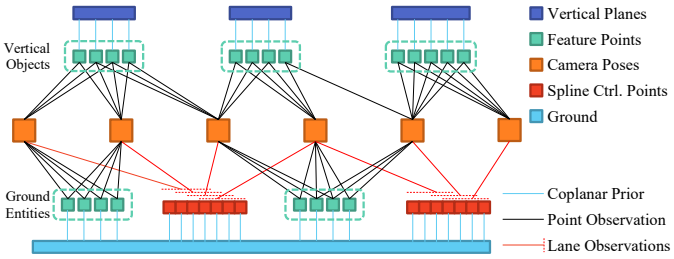


Fig. 3: Illustration of the proposed factor graph structure. Basic graph elements from visual-inertial odometry [19] systems are hidden.

Initialization of splines. Based on the solved ground coefficients $\mathbf{G}$, we use the method described in Sec. II-E to initialize 4). Finally for 5), we first assign $t^c_{b^j}$ with the solved 3D association parameter during spline initialization, and then perform a standalone non-linear optimization to refine. We filter those samples having multiple association hypotheses, i.e., exist sub-optimal association whose residual is less than 1.5 times of the best association. Compared to pixel-on-line whose projective association is deterministic as the perpendicular feet in the image space, explicitly formulating t_{aci} from such a piecewise curve as the *argmin* of t_{aci} in Eq. 5 is hard due to the piecewise representation and the projection operator $\pi(\cdot)$. So we add such a variable type during our state estimation to ensure the correctness of curve projective association. Specifically for those detected dashed lane corners because they are tracked pixel-wise, we associate multiple observations of a corner $p^c_{b^n}$ to one exact location $\mathbf{C}_b(t_{b^n})$ along the curve.

Offline mapping case. After these variables are initialized, we derive a factor graph optimization based on the common visual-inertial odometry constraints [7], with Cauchy loss functions added on the above factors for numerical stability, and the pose of the first frame fixed as identity. All keyframes and detected entities are involved in the final bundle adjustment for solving poses and positions jointly.

Online localization case. During online localization, we deserialize fixed semantic landmarks, i.e., control points of splines and regular 3D points, from a semantic map, and fix them in Eq. 4 and 5, to add constraints on relative poses between the camera and the map coordinate. Coplanar constraints as Eq. 6 are no longer required in this phase. We will further introduce how these factors are constructed through the proposed map query strategy in Sec. II-I.

G. Re-Identification and Feature Merging

We perform 3D-3D association for re-identifying semantic objects rather than a frame-wise bag-of-words [30] query.

The reason is that the density of duplicated objects (tens of meters) are relatively sparser than the localization uncertainty of GNSS-VIO odometry during mapping, and the visual appearances among these standardized road elements are too similar to be distinguished. During the instance-wise object and lane association, we regard triangulated objects with the distance between their centroids smaller than 5.0 meters (or 0.5 meters for lanes) as identical, and then cascadingly merge the observations of their contained deep and classical points in a Hungarian strategy: The semantic type of deep points is used for rejecting mismatches. While for each accompanied GFTT point, we use their FREAK descriptors among multiple frames for voting. We use the union-find algorithm to merge their observations and conduct another round of global state optimization.

H. Data Structure of Semantic Maps

After an open-loop (Sec. II-F) and a close-loop (Sec. II-G) bundle adjustment, we obtain the final frame poses and landmark positions. To save these assets for localization, we define the data structure of our semantic map $\mathcal{M}$ as the combination of observers $\mathcal{O}$ (i.e., frames used during mapping), landmarks $\mathcal{L}$, and a co-visibility graph $\mathcal{G}$ linking $\mathcal{O}$ and $\mathcal{L}$. For each observer c, we store its estimated pose in the global coordinate $\mathbf{T}_c$ and the closest GNSS measurement for online GPS queries. For each hierarchical semantic landmark, we store the semantic label and the 3D positions of contained deep and GFTT points. Neither FREAK descriptors nor frame-wise descriptors are stored in our semantic map.

I. Localization Based on Semantic Maps

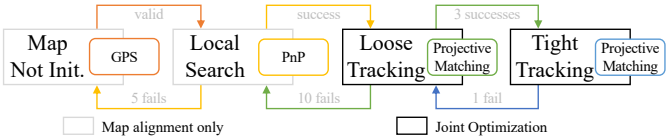


Fig. 4: State machine during online localization.

We use a state machine illustrated in Fig. 4 to assess the pose quality of online localization and perform different strategies accordingly. Starting from the *map uninitialized* state, in which the global transformation from the map coordinate to the current global coordinate $\mathbf{T}_{\mathcal{M}}$ is unknown, we use a coarse GPS measurement to retrieve the corresponding map partition, to get the corresponding observers in $\mathcal{O}_{\mathcal{M}} \subset \mathcal{O}$ whose relative translation is less than 30 meters, and switch to the *local search* mode.

In the *local search* mode, we can obtain the landmark subset $\mathcal{L}_{\mathcal{M}} = \mathcal{G}(\mathcal{O}_{\mathcal{M}})$, for establishing 2D-3D PnP-Ransac association [31] through indexed deep object keypoints. At least 12 inliers from 5 different entities are required for accepting such an estimated PnP pose as valid, to switch into the two *tracking* states and initialize $\mathbf{T}_{\mathcal{M}}$.

We use two stages of the *tracking* state as *loose* and *tight*, to apply different thresholds according to their pose quality. In both *tracking* states, reconstructed associations will be added to the sliding window optimization through Eq. 4 and 5, where the original $\mathbf{T}_c$ for the frame to global transformation is replaced by $\mathbf{T}_{c\mathcal{M}}$ for the frame to map transformation, and $\mathbf{T}_{\mathcal{M}}$ is regarded as an additional optimizable variable during the optimization.

In such two *tracking* modes, we project all map entities to the online detected frame c, and use the following criterion to accept a projective association through a Hungarian matching, as:

$$ {}^{\mathbf{y}}\mathbf{f}_{\mathbf{x}}^{c} < \|\sigma_2\|^2_{{}^{\mathbf{y}}\Omega_{\mathbf{x}}^{c}}, \ \mathbf{x} \in \{a^i, b^j\}, \ \mathbf{y} \in \{1, 2\}, \tag{7} $$

where threshold σ_2 in the *tight* mode is set as 20.0 and 3.0 pixels for deep and classical keypoints, respectively, due to their density on the projected image. In the *loose* mode, we loose σ_2 to 30.0 pixels for labeled deep keypoints, and disable classical keypoints since they are not distinguishable when the geometry is not accurate. To remove the influence of association ambiguity, we reject the association to a map point if there exists another sub-optimal candidate whose projected distance $\hat{\mathbf{f}}/\mathbf{f} < 4.0$. For state transformation, we regard a projective matching as stable if more than 4 associations are constructed.

III. Experimental Evaluation

A. Datasets

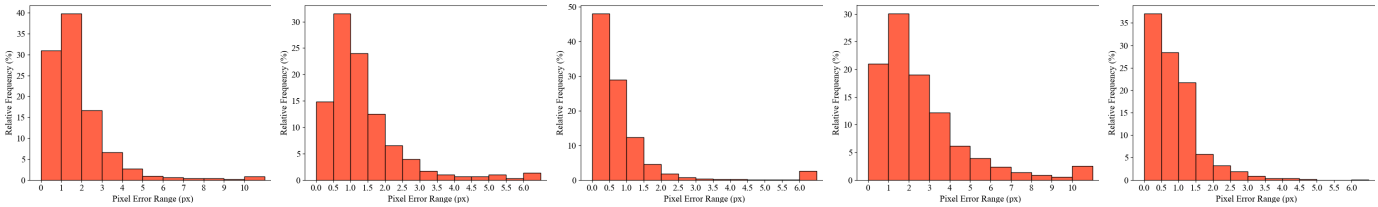
We use two real-world datasets to evaluate our performance, including a publicly available dataset KAIST [32], and a self-recorded dataset. For KAIST sequences, we use the left camera with IMU and GNSS measurements, on urban road sequences 26, 28, 38, 39. Note that some of their trajectories are overlapping with others. We also recorded two sequences in Hangzhou, from hardware synchronized sensors including a 10Hz ON AR0144 stereo camera, a 200Hz Bosch BMI 088 IMU and a 10Hz u-blox F9 RTK-GPS. We recorded a long sequence called Hangzhou-1 traveling 4.7 km for mapping, and another sequence Hangzhou-2 as a subset having 3.2 km for cross localization.

B. Training and Performance of Perception Model

Training. We manually annotated 3207 images extracted from these four KAIST sequences (4.4% of all). These annotations include 2D object boxes, lane contours, and instance keypoints illustrated in Fig. 2. We randomly divide labeled images into a training set and a test set with the ratio of 85% and 15%, and augment them through scaling and color enhancement to generate 512×512 model inputs. We use the Adam optimizer [33], and set the initial learning rate to 1.25×10^{-4}, which drops by 0.1 at 100 and 200 epochs, to train the model separately in different tasks. For Hangzhou sequences, we similarly annotated 2467 images for detection.

Performance. Table III shows the performance of our trained deep model on the KAIST dataset, with box objects grouped by three representative sub-types (poles, traffic signs, and ground signs) due to their different appearances on images. We evaluate these tasks separately on the test set, to conclude their classification precision, detection recall, and additionally the pixel error of extracted pixels in Table III. We also report the distribution histogram of pixel error in Fig. 5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2021.3068948, IEEE Robotics and Automation Letters



(a) Poles (b) Traffic signs (c) Ground signs (d) Lane samples (e) Lane corners

Fig. 5: Pixel error histograms of detected keypoints on different categories, with exceeded errors trimmed into the last bin.

TABLE III: Performance of our proposed multi-task detection model. PRE.: Precision. REC.: Recall. APE.: Average pixel error. MPE.: Median pixel error. We use the MPE. for measurement uncertainties σ in Eq. 4 and 5.

	Keypoint in Box			Lane	
	Poles	Signs	GND.	Samples	Corners
PRE.(%)	85.9	89.0	93.3	93.3	98.1
REC.(%)	95.4	94.2	81.6	89.0	94.2
APE.	3.809	2.237	1.689	2.973	0.857
MPE.	1.952	1.405	1.077	0.524	0.717

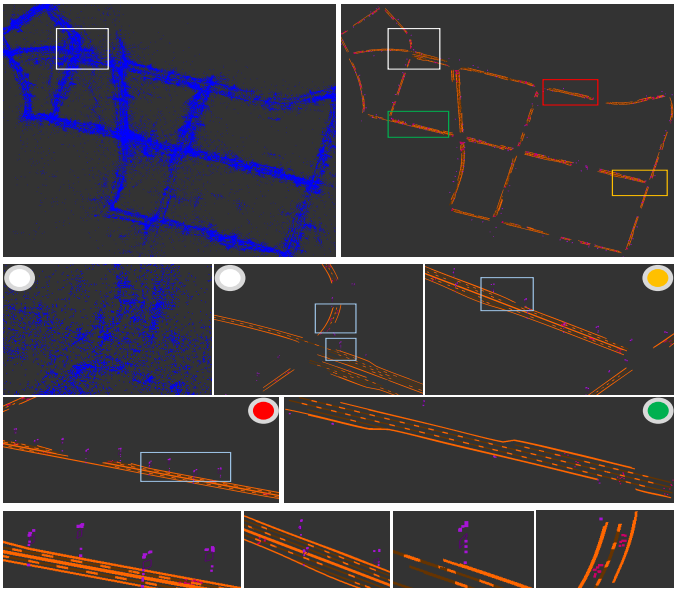


Fig. 6: The semantic map of KAIST-38 with zoom-in views. Blue: Conventional map for reference. Orange: Road lanes. Purple: Points of road poles. Red: Points of ground signs.

C. Evaluations on Localization and Mapping

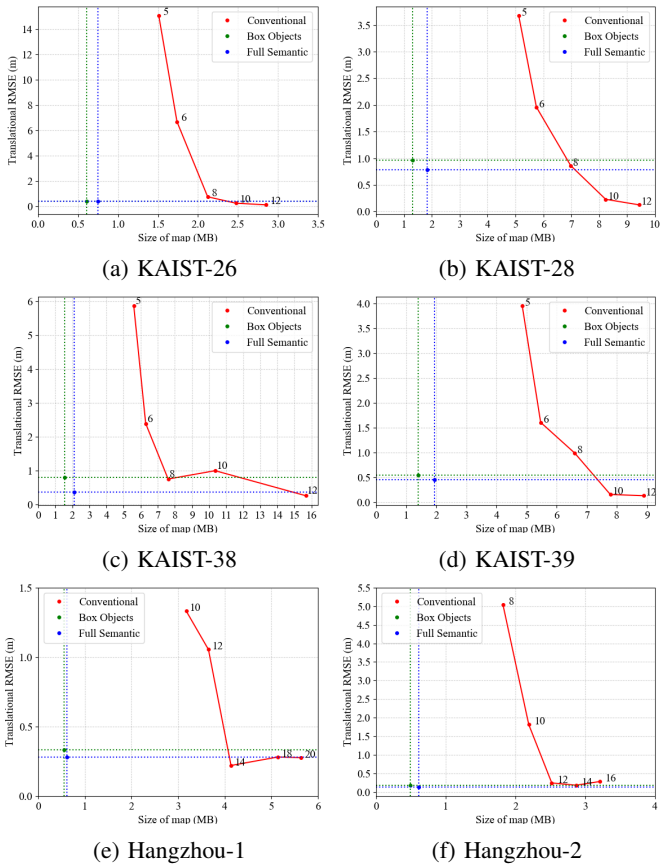
Experimental Settings. We choose to compare the proposed method with downsampled maps consisting of classical feature points described by FREAK [22]. About 500 GFTT [21] keypoints (same with our system) are extracted on each keyframe to perform sliding window optimization, frame-wise loop-closure, and final bundle adjustment. We use high-quality GNSS measurements during offline mapping and low-quality GPS during online relocalization. In detail, there are two differences between our proposed method and this baseline: 1) After the final bundle adjustment of offline mapping, we perform a spatial re-identification for semantic entities and thus require another round of final bundle adjustment. 2) During frame-to-map association in the online localization, our method replaces all classical frame-map local feature associations by these serialized semantic features stored in a semantic map.

To make the file size of these compared maps close, we use a set cover algorithm [34] to downsample the classical feature map. It ensures that each member in the involved map observers $\mathcal{O}$ is able to observe at least a certain number of map points. Also, each FREAK descriptor is compressed into a 10-dimensional vector (40 bytes in total). By varying the parameter K used in the set cover algorithm, we obtain several sparsified conventional maps with different memory footprints. We also provide another variant of our method as a box-object-only version which does not contain any lane procedures for ablation study. During the quantitative evaluation, we mainly test the absolute trajectory error by the translational root-mean-squared-error (T.RMSE) [35]. For self and cross relocalization, we register online localization trajectory to the offline mapping trajectory for demonstrating the map-aided localization ability. Fig. 6 visualizes a typical example of our semantic map.

Self-Relocalization. In this test, we run each sequence twice for mapping and map-aided localization, respectively. Fig. 7 lists the results on tested sequences. The semantic map generated by our proposed method requires much less file size while providing a competitive localization performance. For example, the semantic map of KAIST-38 (Fig. 7c) spends 1.54 MB disk space to reach 0.46 meters localization precision, while the conventional map requires 7.60 MB for 0.75 meters by setting $K = 8$. Adjusting the sparsification parameter K cannot reach the same size-efficiency of our method. Tests on Hangzhou sequences also reflect the same trend. Overall, the semantic maps in our proposed method are nearly 5 times smaller than those set cover point feature maps to reach similar localization performance.

Cross-Relocalization. We conduct the cross-relocalization experiments with both conventional maps and a state-of-the-art system Maplab [5]. Same with our two-stage manner, Maplab is also an open source mapping and localization framework. Since its original version does not utilize GNSS measurements, we transform those maps constructed by our baseline method [7, 36] to enhance the quality of localization summary maps. Furthermore, we test its included map sparsification method [37] for compression by different percentages D of point landmarks preservation.

Table IV shows such reciprocal cross-relocalization results. As can be seen, our proposed method with semantic maps



(a) KAIST-26 (b) KAIST-28

(c) KAIST-38 (d) KAIST-39

(e) Hangzhou-1 (f) Hangzhou-2

Fig. 7: Self-relocalization results on four KAIST and two Hangzhou sequences. Red: The results with sparsified conventional maps containing point descriptors under different set cover parameters K. Blue: Proposed method without lanes. Green: Proposed method.

containing box objects and road lanes accomplishes a successful localization on all cross-validation tests. For conventional maps, we test with multiple settings including a full-size map. Unfortunately, the localization consistently fails on running the KAIST-38 sequence on the map of 28 sequence, due to the severe change of visual features captured in different seasons and time periods. Hence, the set cover sparsification needs to apply a larger K value for cross-relocalization to maintain sufficient reliable features. For Maplab, The resultant map size is larger than expected, since the preserved informative 3D points are usually associated with more local visual descriptors. In contrast, our proposed method relies on deep features extracted from standardized and persistent road elements. In conclusion, such semantic replacement outperforms both conventional maps and Maplab by a larger margin on cross-localization, and experiments on Hangzhou sequences reflect the same trend, which shows the superiority in both compactness and localization accuracy.

Modular and Efficiency Analysis. Table. V presents the statistics of two *tracking* states involved in our localization. In most cases, our method runs in the *tight tracking* mode, which reflects a better localization accuracy. Table VI shows the average number of points used from different semantic

TABLE IV: Cross-validation results. We report the T.RMSE (T) in meters, map size in MB (Size), preserving percentage (D), set cover parameter (K), and the map size ratio (Sz.Ratio) between two compared methods and ours.

Maplab [5]			Conventional			Box objects only		Full Semantic	
T	Size	D	T	Size	K	T Size	Sz.Ratio	T Size	Sz.Ratio
KAIST-38 localization on KAIST-28 Map									
Failed			Failed			1.49	-	**1.40**	-
						1.31		1.83	
KAIST-28 localization on KAIST-38 Map									
12.85	35.77	20%	7.88	43.60	60	**0.96**	$\times 23, 28$	0.99	$\times 17, 21$
4.28	54.12	35%	10.97	57.29	80		$\times 35, 37$		$\times 26, 27$
2.86	65.92	50%	0.93	69.50	100	1.54	$\times 43, 45$	2.12	$\times 31, 33$
2.49	94.88	full	0.86	94.88	full		$\times 62, 62$		$\times 45, 45$
Hangzhou-2 localization on Hangzhou-1 Map									
8.89	11.27	20%	8.55	8.04	30	0.29	$\times 20, 14$	**0.18**	$\times 18, 13$
3.54	16.86	35%	0.89	10.37	40		$\times 30, 18$		$\times 27, 17$
1.59	19.66	50%	0.38	12.40	50	0.56	$\times 35, 22$	0.62	$\times 32, 20$
0.74	29.20	full	2.54	14.00	60		$\times 52, 25$		$\times 47, 23$

entities. As dash cams can observe and detect valid pole and sign objects relatively far than ground objects, these objects contribute the most to the localization.

We also analyze the time consumption for semantic object association in different localization modes. During online localization, both the perception and the map query module runs in a standalone thread. In the perception module, it takes 17.6 ms / 693 MB for lanes, 7.5 ms / 422 MB for corners, and 17.3ms / 1177 MB for other objects and keypoints if individually detected respectively. In the map query module, the average time consumption of the *local search* mode is around 300 ms, while for the dominant *loose tracking* or *tight tracking* modes it reduces to less than 1 ms. The overall time consumption is lower than our designed localization query frequency (1Hz).

TABLE V: The number of attempts, percentage, and T.RMSE in meters of different tracking states.

	Loose Tracking			Tight Tracking		
	Num.	%	T.RMSE	Num.	%	T.RMSE
KAIST Dataset						
38 on 28	157	9.3	1.01	1533	90.7	**0.81**
28 on 38	264	17.2	1.04	1272	82.8	**0.75**
26	20	3.9	**0.15**	488	96.1	0.17
28	161	10.8	1.13	1330	89.2	**0.47**
38	154	7.9	0.89	1796	92.1	**0.37**
39	111	7.4	0.50	1386	92.6	**0.13**
Hangzhou Dataset						
2 on 1	42	10.6	0.30	354	89.4	**0.16**
1	80	14.1	0.39	487	85.9	**0.27**
2	35	8.5	0.27	377	91.5	**0.17**

Stability of Semantic Mapping As experiments above mainly demonstrate the effectiveness during online localization, we additionally perform a quantitative evaluation on the mapping quality, by comparing trajectories during both conventional mapping and semantic mapping to their corresponding dataset baseline, i.e., the GNSS trajectory of Hangzhou sequences and provided ground truth trajectory of KAIST, for clarifying the stability of involving such semantic features. Table VII shows the mapping errors in both classical and semantic mapping methods. This suggests that introducing

TABLE VI: The summarization about the average number of points used from semantic entities under different experimental settings. We count the descriptor-less classical feature points (CP), deep points from poles and signs (P+S), ground (GND) boxed objects, and lane entities.

		Box Objects			Lane	
		CP	P+S	GND	Corners	Samples
KAIST	self	3.10	4.67	1.99	1.65	178.61
	cross	0.85	3.87	1.65	3.11	171.67
Hangzhou	self	0.82	6.83	3.52	3.45	64.35
	cross	0.70	8.12	3.51	2.96	68.14

semantic objects does not obviously influence the mapping quality.

TABLE VII: Mapping T.RMSE in meters compared to the given ground truth trajectory.

	KAIST				Hangzhou	
	26	28	38	39	1	2
Classical	1.75	2.69	2.61	2.41	0.81	0.85
Semantic	1.77	2.67	2.63	2.43	0.82	0.84

IV. Conclusion

In this paper, we presented a semantic mapping and localization pipeline. Entities including poles, signs, and road lanes, are detected and parameterized to form a compact semantic map for efficient and accurate localization.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *TRO*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *RAL*, vol. 2, no. 2, pp. 796–803, 2017.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

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

[5] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *RAL*, vol. 3, no. 3, pp. 1418–1425, 2018.

[6] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, *et al.*, "Benchmarking 6dof outdoor visual localization in changing conditions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8601–8610.

[7] M. Zhang, X. Zuo, Y. Chen, and M. Li, "Localization for ground robots: On manifold representation, integration, re-parameterization, and optimization," *arXiv preprint arXiv:1909.03423*, 2019.

[8] R. Gomez-Ojeda, F. Moreno, D. Zuiga-Nol, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *TRO*, vol. 35, no. 3, pp. 734–746, 2019.

[9] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up slam: Semantic monocular plane slam for low-texture environments," in *IROS*, 2016.

[10] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *TRO*, vol. 35, no. 4, pp. 925–938, 2019.

[11] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015.

[12] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," in *CVPR*, 2018.

[13] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation," in *CVPR Workshops*, 2018.

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

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *TRO*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[17] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *CVPR*, 2018.

[18] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards End-to-End lane detection: An instance segmentation approach," in *IEEE Intelligent Vehicles Symposium*, 2018.

[19] M. Zhang, Y. Chen, and M. Li, "Vision-aided localization for ground robots," in *IROS*, 2019.

[20] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, 1981.

[21] J. Shi *et al.*, "Good features to track," in *CVPR*, 1994.

[22] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *CVPR*, 2012.

[23] E. Catmull and R. Rom, "A class of local interpolating splines," in *Computer Aided Geometric Design*. Elsevier, 1974, pp. 317–326.

[24] L. Pedraza, D. Rodriguez-Losada, F. Matía, G. Dissanayake, and J. V. Miró, "Extending the limits of feature-based slam with b-splines," *TRO*, vol. 25, no. 2, pp. 353–366, 2009.

[25] D. Rao, S. Chung, and S. Hutchinson, "CurveSLAM: An approach for vision-based navigation without point features," in *IROS*, 2012.

[26] Z. Wang and L. Kneip, "Fully automatic structure from motion with a spline-based environment representation," *arXiv preprint arXiv:1810.12532*, 2018.

[27] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *TRO*, vol. 24, no. 5, pp. 932–945, 2008.

[28] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *ICRA*, 2011.

[29] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *ICRA*, 2011.

[30] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *TRO*, vol. 28, no. 5, pp. 1188–1197, 2012.

[31] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *IJCV*, vol. 81, no. 2, pp. 155–166, 2009.

[32] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *IJRR*, vol. 38, no. 6, pp. 642–657, 2019.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," in *RSS*, 2015.

[35] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *IROS*, 2018.

[36] Y. Chen, M. Zhang, D. Hong, C. Deng, and M. Li, "Perception system design for low-cost commercial ground robots: Sensor configurations, calibration, localization and mapping," in *IROS*, 2019.

[37] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *IROS*, 2015.