Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving

Daniel Wilbers[1] Christian Merfels[1] Cyrill Stachniss[2]

***Abstract*— Localizing a vehicle in a map is essential for automated driving and various other robotic applications. This paper addresses the problem of vehicle localization in urban environments. Our approach performs a graph-based sliding window optimization over a set of recent landmark and odometry measurements for fast and accurate vehicle localization on third-party maps. Our work incorporates landmark priors from third-party maps into the estimation problem and shows how to exploit the sliding window formulation for revising data associations. We describe how to construct our factor graph and derive its necessary factors to model the information from the map as a prior over the landmark detections. We implemented our approach on an automated car and thoroughly tested it on real-world data. The experiments suggest that the approach provides highly accurate pose estimates, is fast enough for automated driving applications, and outperforms localization using particle filters.**

I. Introduction

Localization is a fundamental task for automated vehicles. Localization computes the vehicle position given a map, which may store additional information that the vehicle might not be able to infer from its sensors. Thus, precise localization with respect to a map is a prerequisite to exploit the map information. In this paper we focus on the problem of map-based self-localization in the context of automated vehicles.

In map-based localization, the vehicle pose is inferred by aligning the current sensor readings to the map of the environment. Most maps for vehicles are globally georeferenced. Therefore, localization often includes a global navigation satellite system (GNSS). However, standard GNSS systems alone are often not accurate enough or not reliably available in regions with degraded or missing satellite reception. Furthermore, high-precision systems combining RTK-GNSS and IMUs are often expensive and thus are not used in series production cars. In contrast, map-based approaches, including ours, are able to compute a global pose with the help of a map in such regions and therefore increase availability while at the same time increasing accuracy.

The localization information in our commercially available, third-party maps consists of landmarks. In contrast to raw sensor data maps, such as point clouds, landmark maps are easier to inspect by a human for correctness and to maintain. Landmark maps also require smaller storage and memory capacities. Most landmarks are at least to some degree sensor-independent as they can generally be detected by different sensors. This is important for sharing maps between multiple vehicles that have different sensor setups.

[1]Daniel Wilbers and Christian Merfels are with Volkswagen Group Research, Wolfsburg, and the University of Bonn, Germany. [2]Cyrill Stachniss is with the University of Bonn, Germany.

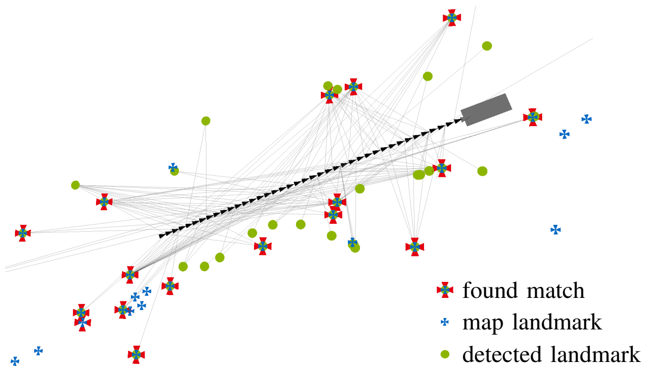

Fig. 1: Matches between landmark observations and map landmarks constrain the vehicle trajectory (black triangles). The connections (gray lines) illustrate from which poses the landmark were observed. The current vehicle pose is depicted as a gray rectangle.

The contribution of this paper is an approach to accurate localization on third-party maps for automated vehicles. It formulates localization as a sliding window factor graph problem, which is closely related to modern simultaneous localization and mapping (SLAM) systems. We consider detections of pole-like objects as landmarks and match them to an existing third-party map, not built by the vehicle. The strength of our approach comes from integrating a third-party map as prior knowledge about the states of the detected landmarks. This allows us to localize the vehicle within a few centimeters of its actual pose without building own maps. See Fig. 1 for an example.

For addressing the state estimation problem, we build up a factor graph with poses and landmarks and in this sense, the approach can be classified as a combination of localization and SLAM: We locally estimate landmark locations as in SLAM but keep a localization map that is not globally updated as in localization. The reason for this procedure is that detecting changes in the landmark locations could, instead of being beneficial, potentially degrade the accuracy of the verified prior map and thus should be executed with care. We therefore avoid updating our map on the fly but estimate updated landmark positions to store them for a later verification step.

In sum, we make four key claims: Our approach (i) provides accurate pose estimates for automated driving based on third-party maps, (ii) allows the vehicle to continuously localize globally accurate, (iii) can successfully revise map associations to increase the localization quality, and (iv) is fast enough for application in an automated vehicle.

II. Related Work

Over the last 30 years extensive research has been done in the field of mapping and localization [1]. We refer to the work of Cadena et al. [2], which provides a broad overview over the challenges and open questions in the field. A recent overview with a special focus on autonomous driving is given by Bresson et al. [3]. Localizing vehicles in an urban environment has been tackled with various combinations of algorithms, sensors, and types of landmarks. Predominantly, the methods of choice for pure localization are either particle or Kalman filters, while SLAM is usually solved by graph-based optimization, see Stachniss et al. [4] for a review of these three methodologies.

The approach by Levinson et al. [5] uses 2D dense lidar reflectivity maps which are generated using graph-based SLAM, while the localization is carried out with a particle filter. They extend their approach to use probability maps and a two-dimensional histogram filter for localization [6]. In contrast, Ziegler et al. [7] use visual features and a Kalman filter for localization on a point feature map, whereas Schuster et al. [8] apply graph-based SLAM using radar sensors for mapping and use random sample consensus map matching for localization. Instead of relying on dense map representations, Brenner [9] proposes the use of pole landmarks for global localization, which is from the application point of view similar to our work. The approach by Spangenberg et al. [10] uses pole landmarks from camera for localization with a coupled particle and Kalman filter. In this paper, we rely on pole landmarks extracted from lidar data and use sliding window graphs. We follow the paradigm that all of the above approaches separate mapping and localization. But instead of building the required maps ourselves, we rely on maps from a third-party distributor.

From a mathematical point of view, our localization approach is closely related to graph-based SLAM [11]. Nowadays, graph-based optimization is considered the de-facto standard for SLAM. However, graph-based approaches for map-based localization applied in urban scenarios still need investigation. Wu et al. [12] contribute to that by investigating the suitability of point and lane features for automated driving. Although their approach uses a similar lidar setup and additionally incorporates camera features, our evaluation suggests that our technical approach provides better results. A difference is that we directly detect poles in the lidar scans without using a grid map, which depends on odometry data and thus propagates odometry errors. Additionally our data association strategy (see Sec. IV-C) benefits from temporal filtering.

For automated driving, our localization algorithm must be computationally efficient as the car requires pose estimates in (near) real-time. One way of limiting the size of the state vector or the number of constraints is to sparsify landmarks and poses in the graph based on information-theoretic metrics [13], [14]. Another one is relying on sliding windows, which can either discard data or use marginalization [15]. While the first one might ignore important data and correlations, the latter suffers from linearization errors, which accumulate over time. Another advantage is that sliding windows ease the integration of out-of-sequence measurements compared to filter approaches [16]. While we presented in our previous work how to synchronize odometry and pose measurements [17] we highlight here different options for landmark measurements. The approach presented in this paper is also used in our other work [18], [19].

Recent research targets on leveraging information from existing maps. The approach by Vysotska and Stachniss [20], [21] derives pose-graph constraints from Open Street Map data to improve mapping and localization. Similarly, Kümmerle et al. [22] derive pose-graph constraints from aerial images. The integration of data based on shape-files is shown by Roh et al. [23]. Lee et al. [24] show how to incorporate road network maps as priors into particle filters. More close to our work is the integration of point features shown in [12]. As recently stated by Bresson et al. [3], current localization approaches based on prebuilt maps are not yet suitable for automated driving. With respect to automated driving, mapping companies are working on closing this gap by providing highly accurate maps. We therefore especially investigate how to use such a third-party map as prior knowledge for localization.

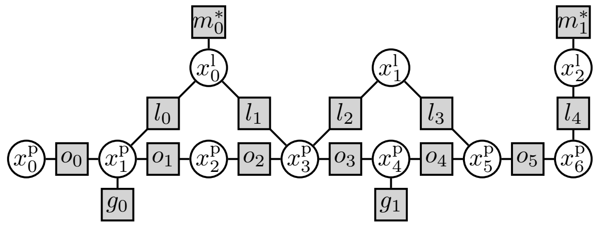

Fig. 2: Exemplary factor graph representation with consecutive pose states x^{p}, landmark states x^{l}, odometry factors o, landmark-observation factors l, GNSS pose factors g, and temporally smoothed map factors m^*. Each factor has a corresponding measurement z_i and information matrix $\boldsymbol{\Omega}_i$. Using map factors allows us to constrain landmarks such that their estimated position is globally bounded. As a result, we can implicitly determine a globally accurate vehicle pose.

III. Graph-based Sliding Window Localization

In the following, we explain our sliding window graph-based localization and show how to use third-party maps in form of priors in the graph representation. Our formulation to localization is similar to SLAM as landmarks are part of the state vector and we refine their locations based on observations within the sliding window graph. The updated landmark locations are, however, not immediately fed back into our prior map to avoid that a localization error compromises the map. We store the computed landmark updates separately in order to refine the prior map only if the update is confirmed over multiple, independent runs. We consider this design decision essential for fulfilling safety requirements in automated driving.

A. Graph-based Optimization

We formulate the problem of determining vehicle poses and landmarks based on a set of measurements as an optimization problem. Here, we use a similar notation to Grisetti et al. [11]. In general, we want to find the maximum a posteriori solution $\boldsymbol{x}^*$ which maximizes the probability $p(\boldsymbol{x} \mid \boldsymbol{z})$ with states $\boldsymbol{x}$ and measurements $\boldsymbol{z}$. In our case, the state vector $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}^{\mathrm{p}} & \boldsymbol{x}^{\mathrm{l}}\end{bmatrix}$ is divided into poses $\boldsymbol{x}^{\mathrm{p}} = \begin{bmatrix}\boldsymbol{x}^{\mathrm{p}}_1, \dots, \boldsymbol{x}^{\mathrm{p}}_T\end{bmatrix}$ and landmarks $\boldsymbol{x}^{\mathrm{l}} = \begin{bmatrix}\boldsymbol{x}^{\mathrm{l}}_1, \dots, \boldsymbol{x}^{\mathrm{l}}_K\end{bmatrix}$. We define the poses as $\boldsymbol{x}^{\mathrm{p}}_t \in SE(2)$ and landmark positions as $\boldsymbol{x}^{\mathrm{l}}_k \in \mathbb{R}^2$. The number of poses T defines the size of the sliding window, whereas K represents the number of landmarks. Using Bayes rule, we write the optimization problem as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\ p(\boldsymbol{x} \mid \boldsymbol{z}) = \underset{\boldsymbol{x}}{\operatorname{argmax}}\ p(\boldsymbol{z} \mid \boldsymbol{x}) p(\boldsymbol{x}), \tag{1}$$

with $\boldsymbol{z}$ capturing all measurements inside the current sliding window. We separate the prior term into priors over vehicle poses and landmark positions as $p(\boldsymbol{x}) = p(\boldsymbol{x}^{\mathrm{p}})p(\boldsymbol{x}^{\mathrm{l}})$. As the priors on the vehicle pose are unknown we assume a uniform distribution for $p(\boldsymbol{x}^{\mathrm{p}})$ and drop the term in the following. Assuming Gaussian distributions and independent and identically distributed measurements, we transform Eq. (1) to

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_i \mathbf{e}_i(\boldsymbol{x}, \boldsymbol{z}_i)^\top \boldsymbol{\Omega}_i \mathbf{e}_i(\boldsymbol{x}, \boldsymbol{z}_i) + \mathcal{F}^{\mathrm{map}}(\boldsymbol{x}^{\mathrm{l}}), \tag{2}$$

with error function $\mathbf{e}_i$, information $\boldsymbol{\Omega}_i$ of the measurement related to measurement $\boldsymbol{z}_i$, and prior knowledge about the landmarks $\mathcal{F}^{\mathrm{map}}$. Defining the error functions $\mathbf{e}_i$ and corresponding information matrices $\boldsymbol{\Omega}_i$ is one of the core aspects of any graph-based approach. We can view each error function together with the corresponding information matrix as a factor and each state variable as a node of a so-called factor graph. We illustrate such a factor graph in Fig. 2. Each factor constrains the states based on the corresponding error function $\mathbf{e}_i$ and information matrix $\boldsymbol{\Omega}_i$. We distinguish between the functions for odometry errors of subsequent states $\mathbf{e}^{\mathrm{odo}}(\boldsymbol{x}^{\mathrm{p}})$, absolute pose errors $\mathbf{e}^{\mathrm{abs}}(\boldsymbol{x}^{\mathrm{p}})$ (e.g., from GNSS), landmark observation errors $\mathbf{e}^{\mathrm{obs}}(\boldsymbol{x}^{\mathrm{p}}, \boldsymbol{x}^{\mathrm{l}})$, and map errors $\mathbf{e}^{\mathrm{map}}(\boldsymbol{x}^{\mathrm{l}})$.

A main aspect of our approach is to apply prior knowledge about the landmark states $\boldsymbol{x}^{\mathrm{l}}$ through the map factors m. We incorporate the prior information $\mathcal{F}^{\mathrm{map}}$ as

$$\mathcal{F}^{\mathrm{map}}(\boldsymbol{x}^{\mathrm{l}}) = \sum_k \mathbf{e}^{\mathrm{map}}_k(\boldsymbol{x}^{\mathrm{l}})^\top \boldsymbol{\Omega}_k \mathbf{e}^{\mathrm{map}}_k(\boldsymbol{x}^{\mathrm{l}}). \tag{3}$$

This form allows us to investigate the effect of the map factors on our optimization. As estimated landmarks and the priors from the map both live in the same coordinate system the error function is simply

$$\mathbf{e}^{\mathrm{map}}_k(\boldsymbol{x}^{\mathrm{l}}) = \boldsymbol{x}^{\mathrm{l}}_k - \boldsymbol{m}_k, \tag{4}$$

with the global position $\boldsymbol{m}_k$ of the landmark in the map.

B. Using Third-Party Maps

Nowadays third-party maps used in the automotive domain are often created in a semi-automatic way, which still requires a lot of manual work. The covariances for landmarks created in such a process are often not reliable or simply not available.We now show how to substitute the unknown map information matrix $\boldsymbol{\Omega}_k$ with an educated guess based on the overall map quality.

In general we assume that in a 2D global reference frame the quality of both coordinate directions is similar. Hence, we assume an isotropic covariance for all map factors and rewrite the map factor summands as

$$\mathcal{F}^{\mathrm{map}}(\boldsymbol{x}^{\mathrm{l}}) = \frac{1}{\sigma_m^2} \sum_k \mathbf{e}^{\mathrm{map}}_k(\boldsymbol{x}^{\mathrm{l}})^\top \mathbf{e}^{\mathrm{map}}_k(\boldsymbol{x}^{\mathrm{l}}). \tag{5}$$

Depending on the assumed map quality we compute the constant variance of the map factors as

$$\sigma_m^2 = \frac{1}{\gamma(c)} r^2, \tag{6}$$

with γ being the two-dimensional inverse-chi-squared cumulative distribution function, confidence c, and radius r. To illustrate this, consider the following example. If we are confident that 95% $(c = 0.95)$ of our landmarks in the map were measured within an error radius of $r = 0.02\,\mathrm{m}$, then Eq. (6) helps us to make an educated guess for the variance constant, which is $\sigma_m^2 \approx 6 \times 10^{-5}$. We use this value in our experiments.

C. Iterative Optimization

Solving the nonlinear optimization problem given in Eq. (2) can be done with an iterative algorithm like Gauss-Newton or Levenberg-Marquardt. As derived by Grisetti et al. [11], we split the problem into linearizing Eq. (2) and computing an iterative update $\Delta\boldsymbol{x}^*$ by solving the linear system

$$\boldsymbol{H} \Delta\boldsymbol{x}^* = -\boldsymbol{b}. \tag{7}$$

The structure of $\boldsymbol{H}$ is directly determined by the graph structure and Jacobians of the error functions. The Jacobian of our map error function $\mathbf{e}^{\mathrm{map}}$ is the identity matrix $\boldsymbol{J}_k = \boldsymbol{I}$ and consequently the corresponding Hessian block and system vector are

$$\boldsymbol{H}^{\mathrm{map}}_k = \frac{1}{\sigma_m^2} \boldsymbol{I}, \tag{8}$$

$$\boldsymbol{b}^{\mathrm{map}}_k = \frac{1}{\sigma_m^2} \boldsymbol{I} \mathbf{e}^{\mathrm{map}}_k. \tag{9}$$

This stems directly from the fact that the map error function is linear. We calculate the complete system as

$$\boldsymbol{H} = \sum_i \boldsymbol{J}_i^\top \boldsymbol{\Omega}_i \boldsymbol{J}_i + \sum_k \frac{1}{\sigma_m^2} \boldsymbol{I} \tag{10}$$

$$\boldsymbol{b} = \sum_i \boldsymbol{J}_i^\top \boldsymbol{\Omega}_i \mathbf{e}_i + \sum_k \frac{1}{\sigma_m^2} \boldsymbol{I} \mathbf{e}^{\mathrm{map}}_k \tag{11}$$

respecting the correct rows and columns of the summands.

D. Limiting the State Dimension

Using a sliding window approach instead of a full batch solution limits the dimension of the state vector such that it can be computed efficiently. One way of limiting the size of the graph is to marginalize out old information with the Schur complement [15]. The idea of marginalization

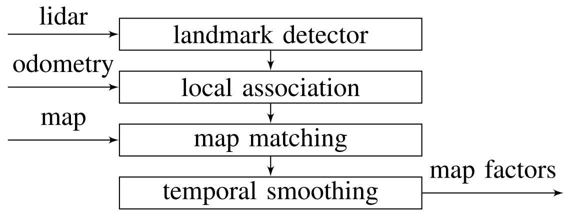

Fig. 3: Overview of the steps for deriving map factors from detected and mapped landmarks.

is to preserve information while simultaneously keeping the state dimension fixed. Marginalized states, however, introduce linearization errors that accumulate over time. It can happen that linearization errors yield incorrect marginal distributions, which might cause a lock-in [15]. Additionally, marginalization causes fill-in in the system matrix $\boldsymbol{H}$.

Another way of limiting the graph size is to simply truncate old states and ignore the data outside the current sliding window. Although this approach forgets information, it does not accumulate linearization errors and avoids fill-in without additional computational effort. Moreover, marginalization has the drawback that faulty data-associations would still have an effect even if the decision has been revised. By truncating the graph we avoid this behaviour. While marginalization serves to constrain the vehicle pose based on old data, using high information map landmarks as priors sufficiently constrains the vehicle pose. That is, provided that enough landmarks in the current sliding window could be associated with the map. As a result, truncation is in our case favorable over marginalization. In our approach, we limit the number of poses in the graph. The poses are generated at a fixed frequency such that we get a steady estimate of the vehicle pose. Contrary to the poses, the number of landmarks in the graph is not limited. It is already naturally bounded by the structure of the environment.

IV. Data Association for Using Prior Maps

One of the major aspects which affect the performance of graph-based approaches is how data association is handled during graph construction. In our case, we distinguish between finding out if multiple detections from different timestamps belong to the same landmark (*local association*) and matching the identified landmarks of the local map to the third-party map (*map matching*). Furthermore, we robustify the map matching step by taking decisions from previous sliding windows into account, an approach we refer to as *temporal association smoothing*.

Our association approach is feedback-free as it does not rely on previous graph optimizations. This contributes to the reliability of our system. In the following, we discuss each of the three steps separately. See Fig. 3 for an illustration of the overall process. Afterwards, we describe the required time synchronization between all measurements when constructing the graph.

A. Local Association

As a first step, in every sliding window, we associate new landmark measurements to previous ones. We call this step local association as we project landmark measurements into a local map based on odometry data. Similar to a nearest-neighbor approach, we compute the Euclidean distance between each new detection and its surrounding detections. Afterwards, we decide based on a threshold if they belong to the same object. The identified objects correspond to the landmarks which are optimized in the graph. We compute an initial position for each identified object in the local map by averaging over the locally projected measurements and reuse it during map matching. Compared to iterative closest point (ICP) approaches our odometry-based algorithm reliably works if the landmarks are not steadily detected in every time step and even handles situations with very few and noisy detections.

B. Map Matching

As a second step, we match the local map to the third-party map, which consists of point coordinates in a global reference frame. We start by projecting the local map into the global frame by using a previous pose estimate. This projection limits the search area and must only be approximate. By doing so we are left with the task of finding the translational and rotational offset between both maps. We use a variant of ICP to search for associations between landmark detections and mapped landmarks. Coping with false positive and false negative landmark detections is one of the main challenges during this step. These errors either depend on the detector performance or are due to physical reasons (e.g., line of sight is blocked). Therefore the overlap between detected and map landmarks might be small. The cost function of our ICP-variant penalizes non-matches and rates the quality of possible associations to find the best alignment and deal with the possibly small overlap. Based on the found alignment, we compute a set of map matches $M_t = \{m_1, \ldots, m_N\}$, with the individual matches m_i. The matches in the set M_t are only based on the measurements inside of the current sliding window. As we repeat this map matching process once for every sliding window, we get different proposed sets $M_{1:T}$ over time. These sets are not directly used in the graph, but contribute to the next step.

C. Temporal Association Smoothing

As a third step, we compute a temporally consistent set of associations $M^* = \{m_1^*, \ldots, m_N^*\}$. To do so we consider all map matches found for previous sliding windows $M_{1:T}$. An important factor during this step stems from the fact that each set of map matches M_t is estimated only based on the measurements inside the corresponding sliding window. By looking at previous matching results, we can overrule current results and filter out outliers. This is done on an individual level for each landmark in the graph. Furthermore, if the map matching step for the current sliding window missed some landmarks we still include the map match based on previous findings. The temporal smoothing even helps in ambiguous situations for which the map matching step produces varying results. Our strategy allows us to always add the most probable map associations to the graph. In cases where we found at a later point in time that a map match is likely to be wrong, the temporal smoothing automatically revises the association and chooses the more likely option. We represent the temporal smoothing as an optimization problem for each individual landmark $\boldsymbol{x}_k^{\mathrm{l}}$ in the graph as

$$m_i^* = \underset{m_i}{\operatorname{argmax}}\, p(m_i|M_{1:T}) = \underset{m_i}{\operatorname{argmax}} \frac{\sum_t \rho_k(M_t, m_i)}{\sum_{t,j} \rho_k(M_t, m_j)}$$
$$\rho_k(M_t, m_i) = \begin{cases} 1 & m_i \text{ matched to } \boldsymbol{x}_k^{\mathrm{l}} \text{ in } M_t \\ 0 & m_i \text{ not matched to } \boldsymbol{x}_k^{\mathrm{l}} \text{ in } M_t \end{cases}, \tag{12}$$

where m_i^* is the optimal association for landmark $\boldsymbol{x}_k^{\mathrm{l}}$, and $\rho_k(M_t, m_i)$ is an indicator function. An important point in our strategy is that the associations are not based in any way on the results of the graph optimization. By avoiding this feedback loop, we prevent the feedback propagation of errors and are not prone to errors like early convergence to local optima.

At this point we have clustered landmark measurements to landmark objects and found out if there is a corresponding match to the third-party map. Overall, our association strategy allows us to introduce the concept of temporal consistency for map matches, handles delayed associations decisions, and enables us to revise associations.

D. *Time Synchronization*

The input to our graph consists of odometry, GNSS data, landmark detections, map landmarks, and map associations. Generally, this data is not synchronized to our pose estimates. There are three different options to cope with this problem. The first option is to *interpolate* between two subsequent measurements. Second, we utilize highly frequent odometry data to *project* measurements to the required timestamp. This option is only advisable if the measurements are within the vehicle frame. Third, it is also possible to simply *ignore* the timestamp misalignment and connect the measurement to the next best node. Which of theses three methods is favorable depends on the type of sensor and the frequency at which nodes are generated. In our case, we use option one for landmark and odometry measurements, and option two for GNSS pose estimates.

V. Experimental Evaluation

Our experiments are designed to support our key claims and illustrate the suitability of our method for automated driving. These key claims are that (i) sliding window graph-based localization on landmark maps provides highly accurate pose estimates for automated driving based on third-party maps, (ii) we continuously localize globally accurate even though GNSS is only used for initialization, (iii) our approach successfully revises map associations to increase the localization quality, (iv) and that our approach is fast enough for application in an automated vehicle. Our experiments are based on an urban drive of 16 km with a prototype vehicle, which detects pole-like landmarks in lidar-scans from Velodyne VLP-16 sensors. See Fig. 4 for the trajectory. We rely on g2o [25] for graph optimization.

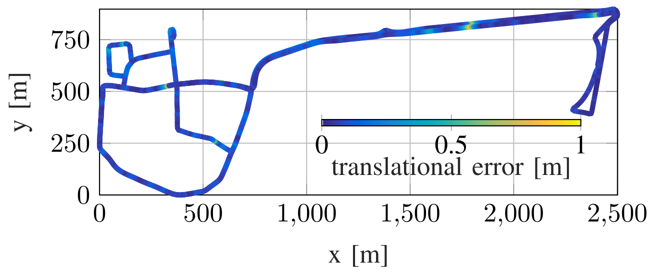

Fig. 4: Trajectory of our 16 km dataset. It contains a variety of typical urban scenarios like stop-and-go traffic, heavy and light traffic, junctions and streets of different sizes, and covers vehicle velocities between 0 km/h and 70 km/h. The color of the trajectory shows the euclidean position error achieved by our GBL approach.

A. *Graph-based Localization for Automated Driving*

Our first experiment is designed to show that our approach provides accurate pose estimates. We test our method in a real-world urban scenario and compare the localization results against a state-of-the-art particle filter (*PF*), which operates on the same map and detections. Using a post-processed RTK-GPS solution with an estimated accuracy of 2 cm as a reference allows us to compute global errors instead of map-relative ones. To show the potential of our approach, GNSS was only used for initialization and not used later during pose estimation. This corresponds to a complete GNSS outage after initial startup. The PF operates in the same setting. Additionally, this demonstrates the capability of our approach in GNSS-denied regions. Fig. 5 compares the lateral and longitudinal error distributions of our experiment. It shows that our approach, called *GBL*, has less outliers than the particle filter in both distributions. Comparing the variances of the error distributions highlights that our approach is more accurate than the particle filter and thus favorable.

B. *Influence of a low-cost GNSS*

Our second experiment is designed to show the global accuracy of our approach. We compare in Tab. I the performance in terms of absolute trajectory error (ATE), average lateral error, and average longitudinal error of a standard-consumer GNSS and a particle filter to two variations of our system. To compute the ATE, a ground truth trajectory is registered to the timestamps of the vehicles pose estimates and afterwards, the mean Euclidean distance for each estimate is computed. Including GNSS data in our graph-based localization (variant 1, named GBL + GNSS) slightly decreases the performance, whereas using GNSS only for initialization (variant 2, named GBL) provides the best results. We attribute this effect to the Kalman-filter based preprocessing inside the GNSS-receiver. Usually, this preprocessing handles multipath effects insufficiently. As a result the GNSS-based estimates are biased. Incorporating this biased data into our sliding window graph, likewise results in biased pose estimates and therefore in less accuracy. The problematic is discussed in more detail by Noack et al. [26].

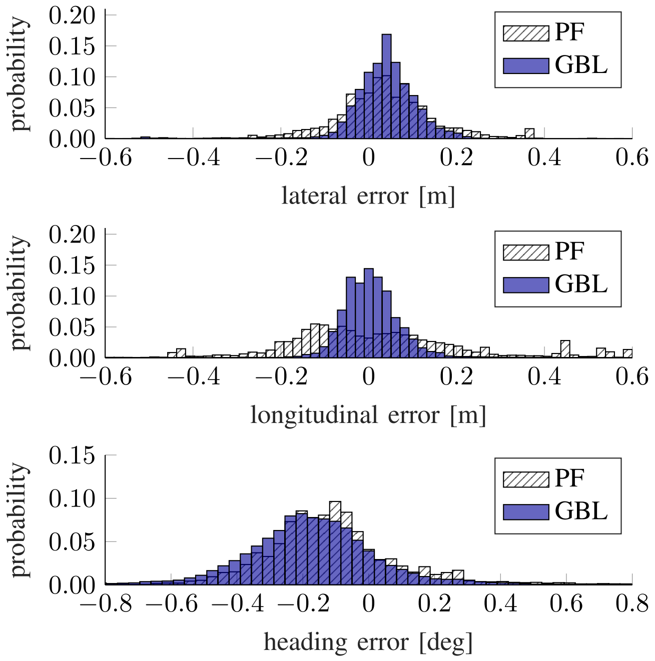

Fig. 5: Comparing the empirical error distributions of our graph-based localization (GBL) against a particle filter (PF) shows the gain in accuracy of our method under full GNSS outage.

error	GNSS	PF	GBL + GNSS	GBL
lateral	0.948 m	0.101 m	0.152 m	0.075 m
longitudinal	1.000 m	0.225 m	0.124 m	0.057 m
heading	5.539°	1.369°	0.418°	0.252°
ATE	1.530 m	0.267 m	0.218 m	0.103 m

TABLE I: Absolute mean errors during a 16 km urban drive. Our GBL approach shows the lowest localization errors if neglecting the low-cost GNSS.

Despite this effect, GNSS is still useful in scenarios without a map at all and provides an option for recovery.

C. Impact of our Data Association Strategy

Our third experiment demonstrates the ability of our system to revise associations between detected and map landmarks. An example of such a revision is shown in Fig. 6. Additionally, Fig. 7 depicts another example where additional landmark detections over time are helpful. During our 16 km test drive, the map associations were revised by our system 35 times. In an additional experiment the ability to revise associations was turned off. The ATE increased from 10.3 cm to 20.1 cm. This highlights the need for subsequent verification of previous map associations. Therefore our strategy for revising associations inside the graph construction based on subsequent additional information is beneficial for the performance of our system.

D. Runtime

Our fourth experiment serves to support our claim that our system is fast enough for usage in a prototype vehicle, which requires pose updates every 50 ms (20 Hz) for automated driving. In our experiments, we limited the number of poses inside the sliding window to 500. The poses inside the sliding

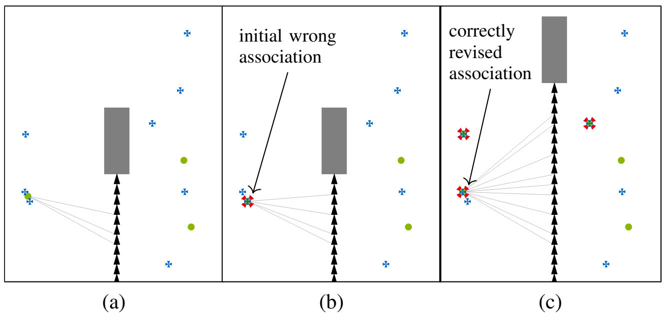

Fig. 6: Illustration of a situation in which a map association is revised after receiving additional landmark detections. For clarity we only show the connections (gray lines) to the revised landmark. (a) Initial unclear situation in which the landmark detection is not associated to the map. (b) An additional inaccurate landmark detection induces a false map association. (c) The situation is revised after additional landmark detections were made.

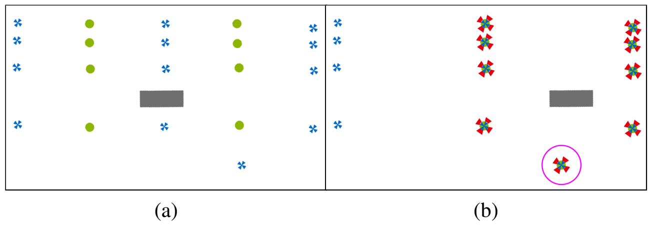

Fig. 7: An ambiguous situation is resolved over time. (a) The vehicle pose (gray) is undetermined as it could either be in the left or in the right side of the image. The correct map associations between map landmarks (blue) and observations (green) are unclear. (b) With an additional landmark detection over time (purple circle) the setting is correctly assessed such that all true map associations (red) are found.

window are 50 ms apart. In contrast, the number of landmarks in the sliding window is generally unlimited but is in practice bounded by the structure of the environment. During the test drive our system aims to compute a vehicle pose every 50 ms. It successfully provided the required frequency within a standard deviation of 1.2 ms. The small deviations are a side-effect caused by our software framework. The time needed for an optimization of one sliding window graph is approximately 14.1 ms ± 3.8 ms. This means that our approach is sufficiently fast for automated driving.

VI. Conclusion

In this paper we presented a sliding window factor graph approach for localization. It uses pole-like landmarks detected with a lidar and matches them against an existing map from a third-party distributor. We showed how to incorporate the map as prior information about landmarks and evaluated our approach with a prototype vehicle in a real-world urban scenario. The vehicle can accurately localize itself even under complete loss of GNSS. A comparison to a state-of-the-art particle filter illustrates the gain in accuracy of our approach. Especially, the ability to revise map associations makes our approach favorable. Our experiments suggest that our approach is fast and accurate enough for the localization of automated vehicles.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robot. & Automation Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[2] C. Cadena, L. Carlone, H. Carillo, 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," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Vehicles*, 2017.

[4] C. Stachniss, J. J. Leonard, and S. Thrun, *Springer Handbook of Robotics*. Springer, 2016, ch. Simultaneous Localization and Mapping, pp. 1153–1175.

[5] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Proc. Robotics: Sci. and Syst. (RSS)*, 2007.

[6] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 4372–4378.

[7] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive – an autonomous journey on a historic route," *IEEE Intell. Transp. Syst. Mag.*, 2014.

[8] F. Schuster, C. G. Keller, M. Rapp, M. Haueis, and C. Curio, "Landmark based radar SLAM using graph optimization," in *IEEE Trans. Intell. Transp. Syst.*, 2016.

[9] C. Brenner, "Global localization of vehicles using local pole patterns," in *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, 2009, vol. 5748.

[10] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016, pp. 2161–2166.

[11] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, pp. 31–43, 2010.

[12] C. Wu, T. A. Huang, M. Muffert, T. Schwarz, and J. Graeter, "Precise pose graph localization with sparse point and lane features," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2017.

[23] H. Roh, J. Jeong, Y. Cho, and A. Kim, "Accurate mobile urban mapping via digital map-based SLAM," *Sensors*, vol. 16, no. 8, p. 1315, 2016.

[13] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear graph sparsification for SLAM," in *Proc. Robotics: Sci. and Syst. (RSS)*, 2014, pp. 1–8.

[14] N. Carlevaris-Bianco and R. M. Eustice, "Conservative edge sparsification for graph SLAM node removal," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 854–860.

[15] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robot.*, vol. 27, no. 5, pp. 587–608, 2010.

[16] A. Ranganathan, M. Kaess, and F. Dellaert, "Fast 3d pose estimation with out-of-sequence measurements," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2007.

[17] C. Merfels and C. Stachniss, "Sensor fusion for self-localization of automated vehicles," *J. of Photogrammetry, Remote Sensing and Geoinformation Sci.*, vol. 85, no. 2, pp. 113–126, 2017.

[18] D. Wilbers, L. Rumberg, and C. Stachniss, "Approximating marginalization with sparse global priors for sliding window SLAM-graphs," in *Proc. IEEE Int. Conf. Robotic Computing*, 2019.

[19] D. Wilbers, C. Merfels, and C. Stachniss, "A comparision of particle filter and graph-based optimization for localization with landmarks in automated vehicles," in *Proc. IEEE Int. Conf. Robotic Computing*, 2019.

[20] O. Vysotska and C. Stachniss, "Improving SLAM by exploiting building information from publicly available maps and localization priors," *J. of Photogrammetry, Remote Sensing and Geoinformation Sci.*, vol. 85, no. 1, pp. 53–65, 2017.

[21] ——, "Exploiting building information from publicly available maps in graph-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016, pp. 4511–4516.

[22] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Auton. Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[24] K. W. Lee, S. Wijesoma, and J. I. Guzman, "A constrained SLAM approach to robust and accurate localisation of autonomous ground vehicles," *J. Robot. Auton. Syst.*, pp. 527–540, 2007.

[25] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[26] B. Noack, S. J. Julier, and U. D. Hanebeck, "Treatment of biased and dependent sensor data in graph-based SLAM," in *Proc. IEEE Int. Conf. Inform. Fusion (FUSION)*. IEEE, 2015, pp. 1862–1867.