Accurate and Efficient Self-Localization on Roads using Basic Geometric Primitives

Julius Kümmerle [1], Marc Sons [1], Fabian Poggenhans [1], Tilman Kühner [1], Martin Lauer [2] and Christoph Stiller [2]

***Abstract*— Highly accurate localization with very limited amount of memory and computational power is one of the big challenges for next generation series cars. We propose localization based on geometric primitives which are compact in representation and further valuable for other tasks like planning and behavior generation. The primitives lack distinctive signature which makes association between detections and map elements highly ambiguous. We resolve ambiguities early in the pipeline by online building up a local map which is key to runtime efficiency. Further, we introduce a new framework to fuse association and odometry measurements based on robust pose graph optimization.**

We evaluate our localization framework on over 30 min of data recorded in urban scenarios. Our map is memory efficient with less than 8 kB/km and we achieve high localization accuracy with a mean position error of less than 10 cm and a mean yaw angle error of less than 0.25 ° at a localization update rate of 50 Hz.

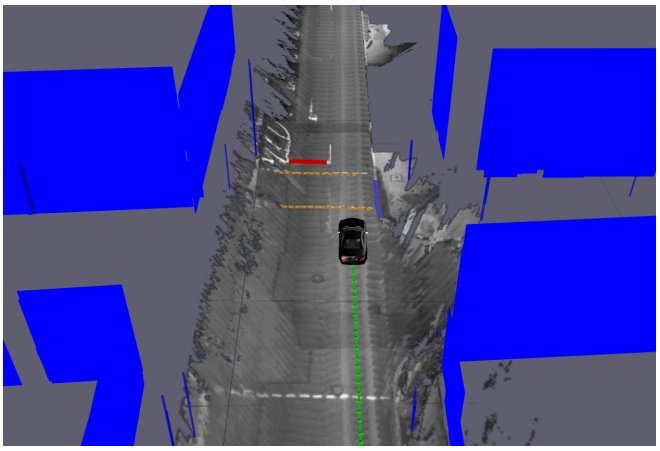

Fig. 1: Snapshot of evaluation drive with localization poses (green arrows) and detections of primitives.

I. Introduction

Up to the present day, series cars are not capable of localizing themselves accurately. Localization with high accuracy is becoming more important since next generation series cars will make use of high definition (HD) maps [1], [2]. HD maps provide detailed information about e.g. lanes, road infrastructure and more with centimeter accuracy. This information will be key to bridge today's shortcomings in perception, scene understanding, planning and behavior generation and thereby will raise autonomy in series cars to the next level. For this to work, next generation series cars have to be capable of localizing in HD maps with centimeter accuracy. Major challenges of introducing highly accurate localization to series cars is reliability and efficiency in the use of computational power and memory.

GNSS based localization is very efficient but it is not reliable enough due to shadowing, multipath and changing weather conditions [3]. Improvement on these problems have been achieved by fusing GNSS with odometry e.g. from IMU or camera [4]–[6]. Still, these approaches can fail e.g. when conditions are bad in the initialization stage. Therefore, localization based on GNSS is considered to be unreliable.

Other approaches use dedicated 3D maps which are only used for localization. In [7], [8], a LiDAR is used to build up a 3D map containing reflectivity of the environment. Classic camera-only approaches also build up 3D maps but with visual landmarks [9], [10]. In [11] a monocular camera is used to localize only based on geometry in a 3D map build up from LiDAR. All these approaches are reported to be highly accurate but need large amount of memory for dedicated localization maps.

Map size can be reduced if only specific structures or objects are use: Localization based on road markings can achieve position accuracy in the centimeter range in many cases [12], [13]. Another approach is to use pole-like structures such as street lamps, traffic signs and trees as landmarks [14], [15]. Especially in urban scenarios, pole-like structures frequently occur and can be used for accurate localization. Also house corners show potential as localization features [16] but more popular are walls for indoor localization [17]–[19]. Since not every type of feature is omnipresent all the aforementioned approaches lack robustness and reliability.

We propose a solution to the problem of highly accurate self-localization with limited resources for next generation series cars: Our main contributions are:

- a set of localization features which are memory efficient and reusable for other tasks such as planning and behavior generation (see chapter II)
- an online built-up local map to resolve ambiguity when associating detections with map elements
- a newly developed pose graph adjuster for robust localization based on asynchronous and delayed measurements

Evaluation on our map (see chapter IV) shows that our approach achieves highly accurate and reliable localization

[1] Intelligent Systems and Production Engineering, FZI Research Center for Information Technology, Karlsruhe, Germany. kuemmerle@fzi.de

[2] Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany.

in challenging inner city scenarios (see chapter V, snapshot in Fig. 1).

II. Localization Features

The choice of features has significant impact on the overall localization performance. We choose features based on two classes of criteria: The first class is focused on accuracy. This includes the most commonly used criteria such as features shall be easy to detect, frequently occurring, viewpoint and weather invariant, to name but a few. The second class of criteria is concerned with efficiency. Our features shall have compact representation to be memory efficient. Further, the features shall be usable beyond localization for other tasks like planning and behavior generation. This would make a dedicated localization layer in the map redundant and further no additional detectors have to run on the system. Based on the aforementioned criteria, we found poles, facades and road markings to be suitable. In the following, more details on the different types of features are given.

A. Poles

In road scenes, there are often lots of signs, traffic lights, street lamps and trees. They are all long-lasting static objects and therefore suitable for localization. Further, because poles are tall structures they are visible even if a car occludes the lower part. Signs and traffic lights are functional elements of road infrastructure and need to be detected by autonomous cars for planning and behavior generation. For detection of poles we use 3D LiDAR measurements and a similar approach as in [15]. We store a pole in the map by it's intersection point with the local ground surface, radius and angle of inclination. The last two attributes are used as a weak signature to distinguish between different poles. This is helpful for creating associations between poles in the mapping and the localization pipeline.

B. Facades

Localization based on facades is motivated by the fact that tall houses deteriorate localization quality when using GNSS. Furthermore, because of their size, houses can be detected from long distances. Geometric changes at house facades are rare so that a map is valid for a long time. Beyond localization, facades can be used as a precise target address to navigate to as well as for inferring observable space. We detect facades with a 3D LiDAR. We use planes to approximate the geometry of facades which is suitable in most cases. In the map, a detected facade is reduced to it's intersection with the ground surface and stored as a 2D line segment.

C. Road Markings

Road markings are designed to be easily recognizable. Their color is chosen to stand out from the rest of the street. As a consequence, it is comparably easy to segment road markings from the street. Besides easy detection, road markings occur frequently which makes them suitable for localization. The downside is that road features occur periodically so that there are many ambiguities in associating detections to map elements. This problem can be resolved with additional features and by building up a local map (see chapter III-B). Road marking detection is already used in series cars for e.g. lane assistance. Further, the type of road marking (e.g. dashed, straight or stop line) defines rules which are important for behavior generation and planning. For detection we use a stereo camera and the approach of [20]. Detections are represented as two 2D points which describe the center line in direction of the long axis of the marking. Additionally, the class of the marking is stored.

III. Localization Framework

A. Overview

The localization framework can be split up into four layers, namely sensor, detector, association and optimization layer (see. Fig. 2). The first layer comprises the sensors used

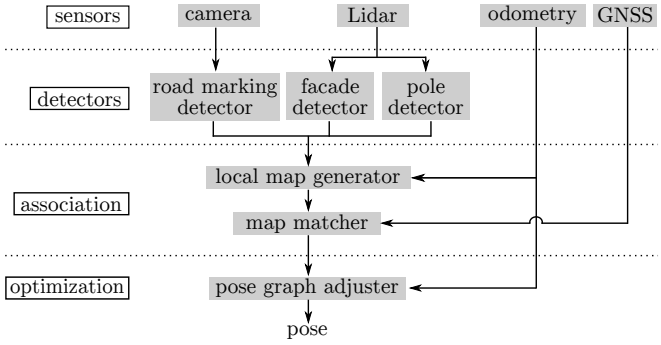

Fig. 2: Overview of localization framework

for localization. Cameras and LiDARs are used to detect localization features. Further, a GNSS receiver is used to have a rough estimate of the current pose and an odometry unit provides a local motion estimate. The detector layer consists of detectors for road markings, facades and poles. The association layer takes the detections as input and associates them to features in the global map. This is a two stage process. First, detections of the near past and present are accumulated in a local map by using odometry. This increases the number of detections in an association step and thereby resolves ambiguities. Second, the map matcher uses a rough pose estimate from GNSS as well as the last pose estimate of the localization as a prior to find an optimal association between features in the local and the global map. In the final optimization layer, pose graph bundle adjustment is used to update the pose based on feature association pairs and odometry measurements.

B. Local Map Generator

A drawback of our geometric primitive features is that they don't have a strong signature such as e.g. visual point features. This makes association of a detection to a map element highly ambiguous. To solve the association problem, we consider multiple detections. The relative positions of these detections serve as a pattern. If the pattern is unique

in the map the association can be solved easily. Ambiguity can occur because of too few detections and periodic patterns. Practical reasons for that are missed features because of limited sensor range and imperfection of the detectors. Further, periodicity is common in road scenes, especially for road markings. The problem of ambiguities can be solved by accumulating detections over time. The higher the number of different detections the less ambiguous the pattern gets. To generate a single pattern from detections of different points in time, the detections are transformed into a static world frame. Therefore the movement of the car has to be known. We use odometry based on wheel encoders as an estimate for the movement of the car. The local map consists of detections in the time span $\mathcal{S} = [t_b, t_e]$. The time t_e is always defined by the most recent detection. The traveled distance $d(.,.)$ between two time points is used to set t_b:

$$t_b = \begin{cases} t_e - \Theta_{t,max} & \text{if } d(t_e - \Theta_{t,max}, t_e) < \Theta_{d,max} \\ t_d & \text{else} \end{cases}, \quad (1)$$

where $\Theta_{t,max}$ is the maximum allowed duration of $\mathcal{S}$, $\Theta_{d,max}$ is the maximum traveled distance and t_d is defined by $d(t_d, t_e) = \Theta_{d,max}$. When accumulating detections longer than $\Theta_{t,max}$, the errors introduced by odometry distort the local map too much. Therefore, the first case in Eq. 1 sets a maximum duration of $\Theta_{t,max}$. The second case is used if the traveled distance reaches $\Theta_{d,max}$ in less than the maximum duration $\Theta_{t,max}$. As a rule of thumb, we assume that there are no more ambiguities if the car traveled $\Theta_{d,max}$.
The resulting local map has more features spread over a wider area and less missed features compared to a single-shot pattern. This makes associations reliable so that dealing with multiple association hypothesis in the optimization layer is redundant. This is crucial for efficiency in the optimization.

C. Map Matcher

Map matching is the task of finding associations between elements of the local map $\boldsymbol{\mathcal{M}}_l$ and global map $\boldsymbol{\mathcal{M}}_g$. We shift the estimated vehicle pose $\mathbf{p}$ to achieve best alignment of $\boldsymbol{\mathcal{M}}_l$ and $\boldsymbol{\mathcal{M}}_g$. A cost function $\mathcal{C}(\boldsymbol{\mathcal{M}}_l, \boldsymbol{\mathcal{M}}_g, \mathbf{p})$ is used to assess the quality of the alignment:

$$\mathcal{C}(\boldsymbol{\mathcal{M}}_l, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) = \sum_{\mathbf{f}_d \in \boldsymbol{\mathcal{M}}_l} \mathcal{C}_f(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}), \quad (2)$$

where $\mathbf{f}_d$ denotes a detected feature. The single feature association cost $\mathcal{C}_f(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p})$ is defined as following:

$$\mathcal{C}_f(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) = \begin{cases} \mathcal{C}_p(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) & \text{if pole} \\ w_{f,1}\mathcal{C}_s(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) & \text{if facade} \\ w_{f,2}\mathcal{C}_s(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) & \text{if road marking} \end{cases}, \quad (3)$$

where $w_{f,1}$ and $w_{f,2}$ are weighting factors for the different feature types. Pole association costs $\mathcal{C}_p(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p})$ are defined as

$$\mathcal{C}_p(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) = \begin{cases} \frac{d_p(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})}{d_{p,\max}} + w_p \frac{\Delta\alpha_p(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})}{\Delta\alpha_{p,max}} & \text{if } \exists \mathbf{f}_m \\ 1 + w_p & \text{else} \end{cases}. \quad (4)$$

A match $\mathbf{f}_m \in \boldsymbol{\mathcal{M}}_g$ exists if it is similar in radius, inclination angle and position to the detection $\mathbf{f}_d$. In case of multiple matches, the match with the lowest cost $\mathcal{C}_p$ is chosen. The pole matching cost is build up of two terms (Eq. 4 first case). The first term penalizes a high distance between $\mathbf{f}_d$ and $\mathbf{f}_m$ and the second term penalizes a large angle difference. The 2D point-to-point distance d_p is normalized with the maximum distance $d_{p,\max}$ of a matched pole pair $(\mathbf{f}_d, \mathbf{f}_m)$. The angle $\Delta\alpha_p$ between the poles is normalized with the maximum angle $\Delta\alpha_{p,max}$ of a match. If no matching pole could be found, the maximum cost is set for the pole $\mathbf{f}_d$ (Eq. 4 second case). The association costs for facades and road markings are computed the same way by using segment matching costs $\mathcal{C}_s(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p})$:

$$\mathcal{C}_s(\mathbf{f}_d, \boldsymbol{\mathcal{M}}_g, \mathbf{p}) = \quad (5)$$
$$\begin{cases} \frac{d_s(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})}{d_{s,\max}} + w_{s,1}\frac{\Delta\alpha_s(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})}{\Delta\alpha_{s,max}} + & \\ \quad w_{s,2}\frac{o_{\text{out}}(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})}{o_{\text{in}(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})} + o_{\text{out}}(\mathbf{f}_d, \mathbf{f}_m, \mathbf{p})} & \text{if } \exists \mathbf{f}_m \\ 1 + w_{s,1} + w_{s,2} & \text{else} \end{cases}.$$

Two segments are a match if they are of the same type (facade, dashed line, stop line, ...), are close in distance and orientation and have a sufficient amount of overlap. The segment matching cost (Eq. 5 first case) is build up of three terms. The first term penalizes the distance between the segments. More precisely, d_s is the distance between the center point of f_d to the line segment f_m. This distance is independent of the difference in orientation and therefore is more suitable than real segment-to-segment distance. The second term penalizes differences in 2D orientation $\Delta\alpha_s$. The third term considers the overlap of the two segments. We define a measure for overlap which is independent of the difference in orientation. One segment is rotated around the center point so that the segments are parallel. Then o_{in} is the overlapping length and o_{out} is the non-overlapping length when projecting one segment onto the other. The parameters $d_{s,\max}$ and $\Delta\alpha_{s,max}$ are used for normalization and $w_{s,1}$ and $w_{s,2}$ are weighting factors. Fig. 3 visualizes the three terms of segment matching costs. If no match was found, the maximum cost for the detection f_d is set (Eq. 5 second case).

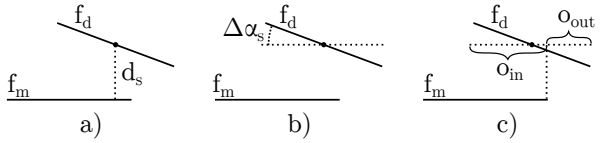

Fig. 3: a) distance , b) orientation difference and c) overlap cost for a segment match.

With Eq. 2-5 we have a tool to compare different proposals for aligning the local to the global map. We need to find good proposals to finally find the best alignment. From GNSS and the last localization iteration we already have pose priors. Therefore, generating proposals can be done by sampling around these priors. For each sampled pose the alignment

cost $\mathcal{C}$ is computed. The proposal with the lowest cost sets the associations for the new detected features. The association pairs are passed to the optimization layer to find an accurate pose estimate of the vehicle.

D. Pose Graph Adjuster

To estimate the final localization pose, we developed a sliding window non-linear least squares (NLS) estimator which adjusts the recent N poses based on odometry measurements and feature-associations as described in section III-C. In comparison to commonly used variants of the Kalman filter or Particle filters, estimating a window of recent poses enables to deal with asynchronous and delayed measurements due to different sensors and preprocessing runtimes, with less effort.

For the optimization, the incoming timestamped odometry measurements have a twofolded purpose. First, each new odometry measurement induces a new timestamped pose for the optimization. Hence, the rate f_o of the odometry defines implicitly the rate of the poses and the time span $l = f_o * N$ of the optimization window W_o. Second, each odometry measurement creates a residual for the optimization. Thereby, each odometry measurement provides a transformation $\mathbf{\Delta}_{i\to i+1} \in SE(3)$ which approximates the motion of the vehicle from pose $\mathbf{p}_i \in SE(3)$ to pose $\mathbf{p}_{i+1}$ at time i, $i+1$, respectively. Therefore, on the one hand, the adjuster minimizes a minimal parameterized expression of

$$r(\mathbf{p}_i, \mathbf{p}_{i+1}) = \left(\mathbf{p}_{i+1}^{-1}\mathbf{p}_i\right)\mathbf{\Delta}_{i\to i+1}, \tag{6}$$

for each odometry measurement within W_o.

On the other hand, the optimizer receives timestamped associations between mapped and detected features as depicted in Fig. 2. In general, the timestamps of these associations do not match exactly with the ones of the poses induced by the odometry. Hence, the adjuster provides two different methods to associate the feature-associations to the poses:

1) Nearest-Neighbor: A feature-association a_k at time k is associated to a pose $\mathbf{p}_i$ which has the closest timestamp i within W_o. However, this can become inaccurate if the odometry has a low rate f_o.
2) Interpolation: A feature-association a_k induces a new interpolated pose $\mathbf{p}_k$. For that, the measured pose difference $\mathbf{\Delta}_{i\to i+1}$, $i < k < i+1$ is split into two pose differences $\mathbf{\Delta}_{i\to k}$ and $\mathbf{\Delta}_{k\to i+1}$. To generate $\mathbf{\Delta}_{i\to k}$ and $\mathbf{\Delta}_{k\to i+1}$, the pose difference $\mathbf{\Delta}_{i\to i+1}$ is interpolated linearly in translation and rotation using the slerp-operation [21].

Each association a_k between a mapped feature $\mathbf{f}_m$ and a detected feature $\mathbf{f}_{d,k}$ at time k which is associated to pose $\mathbf{p}_i$ at time i states a cost term $r(a_k, \mathbf{p}_i)$ for the optimization:

$$r(a_k, \mathbf{p}_i) = \begin{cases} w_1 d_p(a_k, \mathbf{p}_i) & \text{for poles} \\ w_2 d_s(a_k, \mathbf{p}_i) + w_3 \Delta\alpha_s(a_k, \mathbf{p}_i) & \text{else} \end{cases} \tag{7}$$

, where w_1, w_2 and w_3 are weighting factors, d_p and d_s are distance measures (see chapter III-C) and $\Delta\alpha_s$ is an orientation difference measure (see chapter III-C). By adding cost terms for the associations, the pose estimation becomes over-constrained and needs to be adjusted. We use the Levenberg-Marquardt algorithm [22] to adjust all poses in W_o. Finally, the most recent pose in W_o is published as our localization result. To meet hard real-time constraints, the optimization ends after a defined duration independently of the current estimate and whether the optimization converged. The most crucial parts of NLS estimation are the initialization of the parameters (six for each pose in W_o) to estimate and to remove or reduce the influence of outlier constraints. For that, we assume that the odometry source provides measurements which are free of outliers. However, this assumption is not realistic for the provided associations due to high ambiguity of the used features. Hence, we apply Cauchy-loss functions [23] to all association constraints to level off the gradients of outliers after an initialization phase. In the initialization phase, loss functions must be deactivated so that large association residuals can force the poses to converge to the map. We compare the number of small residuals to the number of all residuals to detect when the initialization phase is completed. For that we define the ratio

$$\varepsilon = \frac{|\{r(a_k) | r(a_k) \in W_o \wedge r(a_k) < T_a\}|}{|\{r(a_k) | r(a_k) \in W_o\}|}, \tag{8}$$

where T_a is a hyperparameter of the optimizer. If ε is lower than a threshold T_ε, we assume that the poses in the optimization window are not well adjusted to the map and the robustification through loss functions is deactivated.

IV. Mapping Framework

Our map generation approach can be split into four parts. First, vehicle trajectory estimation based on visual point features fused with GNSS poses [10], [24]. Second, transformation of the detections from local to map frame. Third, clustering of detections which potentially describe the same elements. And finally, merging detections in a cluster to a single map feature.

V. Results and Evaluation

A. Dataset

For the purpose of evaluation, we collected data with our experimental vehicle *BerthaONE* [25]. For this dataset we used four Velodyne VLP16 LiDARs mounted flat on the roof, two BlackFly PGE-50S5M cameras behind the windshield, a Ublox C94-M8P GNSS receiver and an odometry unit which fuses steering angle and wheel speed. The sensor setup is precisely calibrated by using a spherical target for camera-LiDAR and LiDAR-LiDAR extrinsics [26]. Checkerboards are used for calibrating camera intrinsics and camera-camera extrinsics [27].

The dataset was recorded in the center of Sindelfingen, Germany (see Fig. 4). It consists of three drives of the same track at different days, daytimes and weather conditions. A single drive takes about 10 min for the 3.8 km track.

We have chosen this track because of two reasons: First, it represents an average inner city scenario with houses

Fig. 4: Track in Sindelfingen, Germany.

and road infrastructure. So, all our localization features are present even though in varying density. The second reason is that the track is considered to be challenging for autonomous driving. The road is mostly narrow which makes accurate localization crucial for safety. Further, conditions for GNSS based localization are bad since tall houses often shadow most GNSS satellites. There are multiple sharp curves and a roundabout which make fast localization updates important to stay within the lane. An additional challenge is dense traffic which often occludes parts of the sensors' field of view.

B. Groundtruth

Our localization framework is designed to achieve position accuracy in the single-digit centimeter range. This defines the accuracy for the groundtruth data needed for evaluation. We generate our groundtruth as following:
As mentioned before, we recorded three drives on the same track. One of them we call the mapping drive which is used to generate our map. The others we call evaluation drives which are used to evaluate localization accuracy relative to the map. We use a mapping approach based on pose graph bundle adjustment with visual point features [10], [24]. We solve the bundle adjustment problem for the mapping and evaluation drives at the same time as if they were a single drive. Therefore, the map and evaluation drives are consistently registered. Registration accuracy can be estimated by analyzing the localization noise which is below 1 cm for position and 0.1 ° for yaw angle.

C. Mapping Results

On the mapping drive we detect 33k road markings, 20k poles and 90k facades. Our mapping framework merges these detections to about 600 road markings, 500 poles and 760 facades. On average 55 road markings, 40 poles and 118 facades are merged to a single map element. Facade map elements are supported by more detections because facades are larger objects which can be detected from far distances. The overall map size is around 30 kilobyte which is less than 8 kilobyte per kilometer. This is very compact when compared to the 2.1 gigabyte needed for the feature map generated by our groundtruth method.

D. Localization Accuracy

We evaluate our localization framework on the evaluation drives and use the map generated on the map drive. We assess localization accuracy based on position error Δpos and yaw angle error Δyaw. Position error Δpos is often split up into lateral position error Δlat and longitudinal position error Δlon because they have different limits. Lateral position error Δlat has to be lower than $50\,\mathrm{cm}$ to stay within the lane but practically we aim for less than $15\,\mathrm{cm}$ for safety. Bounds on longitudinal position error Δlon are less strict since the lane is not left that quickly in longitudinal direction. A longitudinal position error of less than $25\,\mathrm{cm}$ is sufficient for safety.
In the following we evaluated the accuracy with different feature sets (see Table I).

TABLE I: Mean errors for different sets of features and different limits on maximum optimization runtime

Features used	**Δlat [cm]**	**Δlon [cm]**	**Δpos [cm]**	**Δyaw [°]**
facades	4	11	12	0.18
poles	7	8	12	0.33
road markings	10	29	34	0.47
all (100ms opt)	3	6	8	0.14
all (50ms opt)	3	7	9	0.18
all (20ms opt)	4	9	10	0.22
GNSS	156	135	227	1.50

When using only facades we already reach low errors in position and orientation. Especially Δlat and Δyaw are low. Low Δlat can be achieved because most of the house facades face orthogonal to the road direction and therefore are strong constraints for lateral direction. Further, because facades extend far they enable exact yaw angle estimation which also keeps Δlat low. Constraints on the longitudinal direction are only given if facades facing in driving direction are detected which is less frequent. This explains why Δlon is more than twice as large as Δlat. With a mean position error of 12 cm, localizing only based on facades already yields accurate results on this inner city track.
Localization only based on poles shows higher Δlat and Δyaw compared to facades only. Errors in longitudinal and lateral direction are almost the same because poles (represented as 2D points) have no single constraint direction such as facades (represented as 2D lines). In general, the yaw angle can be accurately estimated by using far detections. Since poles are small objects compared to facades, they can only be detected in close neighborhood. Therefore, Δyaw is larger in case of using only poles for localization compared to using only facades. The mean position error of 12 cm when only using poles is similar to the results for only using facades.
Localization only based on road markings is more challenging on this track because road markings are not always present. Odometry has to bridge some regions without markings. So, it is not surprising that accuracy is significantly worse compared to localization based on the other features. Furthermore, at some places periodicity causes localization to shift by one period length in longitudinal direction. This

is why especially Δlon is higher.
Using all feature types for localization leads to significantly better accuracy than only using one feature type. The major reason is higher availability of features. Further, the complementary strength of the features is beneficial for accuracy. E.g. facades provide accurate lateral but weak longitudinal positioning. In this case, road markings or poles can constraint strongly in longitudinal direction. Fig. 5 shows error plots of one evaluation drive. We notice that variance in Δlat

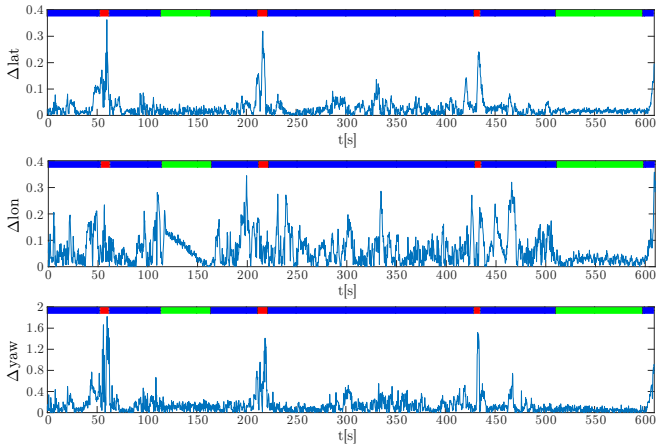

Fig. 5: Results of one evaluation drive with all feature types. Color code: low dynamic(green), normal dynamic(blue), high dynamic(red)

and Δyaw is low except for peaks at 60 s, 220 s and 440 s. When moving, an error in yaw angle must lead to increasing error of lateral position. That is why peaks in Δlat and Δyaw occur at the same time. We notice that these high peaks are all in sharp curves with high yaw rates of up to $30\,^\circ/\mathrm{s}$. Fig. 6 shows the scenario of time point 60 s for which Δlat and Δyaw are the largest. There are several sources of error

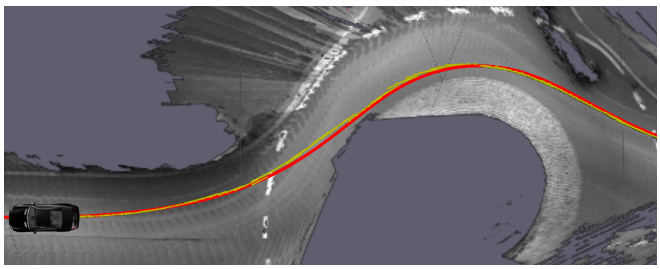

Fig. 6: Roundabout at time point 60 s. Poses from the localization framework(yellow). Groundtruth trajectory(red).

leading to this result: First, to generate pairs of groundtruth and localization poses we have to perform interpolation because of different timestamps. The time difference can be up to 50 ms. We perform linear spherical interpolation which is not perfect for non linear rotations (changing yaw rates). E.g. the worst case error is $0.38\,^\circ$ for linear rise of yaw rate up to a maximum of $30\,^\circ/\mathrm{s}$. Another error source is due to delays. Sensor delay (up to 50 ms for our cameras) in addition with detection and association time can sum up to over 100 ms. Therefore, roughly the last 100 ms of the optimization window W_o only consists of odometry data. Our odometry unit is a build-in system on the car and we can only stamp the data with the time at which it arrives at our system. So, an error in yaw angle of $1\,^\circ$ could be caused by a delay of 33 ms when we assume a step in yaw rate of $30\,^\circ/\mathrm{s}$. Further, we noticed errors in the map for features detected by LiDAR. Our LiDARs are spinning sensors with a spin rate of 10 Hz. We generate $360\,^\circ$ pseudo-single-shot scans by compensating the vehicle motion which is estimated by the odometry unit. Again, a delay in the odometry unit quickly leads to noticeable errors for dynamic drives. Adding up these error sources can explain the peaks in Δyaw and Δlat for dynamic drives in sharp curves.
Position error in longitudinal direction is more noisy than in lateral direction. This is expected since less features constraint the position in longitudinal compared to lateral direction.
Overall, position error is 98 % of the time smaller than 25 cm and reaches a maximum of 38 cm. These results are achieved with a runtime limit of 100 ms on the pose graph optimizer. Table I shows that results deteriorate with runtime limit of 50 ms and 20 ms but are still sufficient for autonomous driving. Setting the runtime limit lower than 20 ms increasingly leads to outliers. Further, we evaluated localization only based on GNSS. We used the solution provided by ublox. The errors are more than a magnitude larger compared to our feature-based solution.

VI. Conclusions and Future Work

In this work we proposed a localization framework which uses basic geometric primitives as localization landmarks. Our evaluation on real data has shown that requirements on accuracy, reliability and efficiency for localization of next generation series cars are met in inner city scenarios. Mean position error of less than 10 cm proofs that road markings, poles and facades are suitable landmarks for accurate localization on urban roads. Reliability is achieved through local map generation and robustification of the pose graph optimization. This is proven by a position error of less than 25 cm 98 % of the time. Memory efficiency of 8 kB/km was reached by a compact representation of the geometric primitives. Efficiency in computational power was shown by limiting the pose graph optimization runtime to a minimum of 20 ms.
In real case scenarios, even the best localization approach will fail some time. Therefore, we want to extend our framework with a measure to assess the current localization quality. Besides criteria based on availability of features and association costs we also plan to use other traffic participants such as cars and pedestrians to detect lost localization.

Acknowledgements

This work was funded by Daimler AG as part of the Tech Center a-drive initiative. The authors would like to thank Daimler AG for their support.

References

[1] *Hd map with roaddna*, http://download.tomtom.com/open/banners/HD_Map_with_RoadDNA_Product_Info_Sheet.pdf, visited on 2018-08-30.

[2] *Here hd live map - the most intelligent sensor for autonomous driving*, https://www.here.com/file/22296/download?token=5W8KZPvT, visited on 2018-08-30.

[3] B. Eissfeller, G. Ameres, V. Kropp, and D. Sanroma, "Performance of gps, glonass and galileo", in *Photogrammetric Week*, vol. 7, 2007, pp. 185–199.

[4] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: Introduction of contextual aspects", *Information fusion*, vol. 7, no. 2, pp. 221–230, 2006.

[5] M. Schreiber, H. Königshof, A.-M. Hellmund, and C. Stiller, "Vehicle localization with tightly coupled gnss and visual odometry", in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2016, pp. 858–863.

[6] K. Jo, K. Chu, and M. Sunwoo, "Interacting multiple model filter-based sensor fusion of gps with in-vehicle sensors for real-time vehicle positioning", *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 329–343, 2012.

[7] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments", in *Robotics: Science and Systems*, vol. 4, 2007, p. 1.

[8] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 4372–4378.

[9] H. Lategahn, M. Schreiber, J. Ziegler, and C. Stiller, "Urban localization with camera and inertial measurement unit", in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2013, pp. 719–724.

[10] M. Sons, M. Lauer, C. G. Keller, and C. Stiller, "Mapping and localization using surround view", in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2017, pp. 1158–1163.

[11] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps", in *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, 2016, pp. 1926–1931.

[12] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps", in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2013, pp. 449–454.

[13] F. Poggenhans, N. O. Salscheider, and C. Stiller, "Precise localization in high-definition road maps for urban regions", *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, 2018, forthcoming.

[14] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2161–2166.

[15] M. Sefati, M Daum, B Sondermann, K. D. Kreisköther, and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks", in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2017, pp. 13–19.

[16] J.-H. Im, S.-H. Im, and G.-I. Jee, "Vertical corner feature based precise vehicle localization using 3d lidar in urban area", *Sensors*, vol. 16, no. 8, p. 1268, 2016.

[17] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5182–5189.

[18] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up slam: Semantic monocular plane slam for low-texture environments", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1222–1229.

[19] J. Biswas and M. Veloso, "Depth camera based localization and navigation for indoor mobile robots", in *RGB-D Workshop at RSS*, vol. 2011, 2011, p. 21.

[20] F. Poggenhans, M. Schreiber, and C. Stiller, "A universal approach to detect and classify road surface markings", in *Proc. IEEE Intell. Trans. Syst. Conf*, 2015, pp. 1915–1921.

[21] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*. Citeseer, 1998, vol. 2.

[22] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large", in *European conference on computer vision*, Springer, 2010, pp. 29–42.

[23] S. Agarwal, K. Mierle, and Others, *Ceres solver*, http://ceres-solver.org.

[24] M. Sons and C. Stiller, "Efficient multi-drive map optimization towards life-long localization using surround view", *IEEE Trans. Intell. Transp. Syst.*, 2018, forthcoming.

[25] Ö. Ş. Taş, N. O. Salscheider, F. Poggenhans, S. Wirges, C. Bandera, M. R. Zofka, *et al.*, "Making bertha cooperate–team annieway's entry to the 2016 grand cooperative driving challenge", *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1262–1276, 2018.

[26] J. Kümmerle, T. Kühner, and M. Lauer, "Automatic calibration of multiple cameras and depth sensors with a spherical target", *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, 2018, forthcoming.

[27] T. Strauß, J. Ziegler, and J. Beck, "Calibrating multiple cameras with non-overlapping views using coded checkerboard targets", in *Proc. IEEE Intell. Trans. Syst. Conf.*, 2014, pp. 2623–2628.