




LiDAR SLAM Positioning Quality Evaluation in Urban Road Traffic

Franz Andert^(✉)  and Henning Mosebach

German Aerospace Center (DLR), Institute of Transportation Systems,
Berlin, Germany
franz.andert@dlr.de
<https://www.dlr.de/ts>

Abstract. This paper addresses the positioning quality of Simultaneous Localization And Mapping (SLAM) based on Light Detection and Ranging (LiDAR) sensors within urban road traffic. Based on the assumption of functional capability of existing SLAM implementations, the paper evaluates specific details of urban car drives that arise when SLAM is to be used for automatic car control. In the presented case, LiDAR-based positioning is done with the Google Cartographer software which generates real-time updates that are compared to GNSS reference. The evaluation is done by using own Light Detection And Ranging (LiDAR) sensor recordings from urban driving. Next to the overall GNSS-free path estimation, the paper zooms into some typical situations (e.g. waiting at busy intersection, driving curves) where SLAM might be inaccurate.

Keywords: GPS-free navigation · SLAM · LiDAR data processing

1 Introduction

Reliable positioning and navigation is essential for all kinds of automated vehicles. In outdoor applications, global navigation satellite systems (GNSS) are ubiquitous, however, positioning is subject to systematic errors of several meters, temporal loss due to signal interruption, reflections, or multi-path, and intentional disturbance. Low integrity is highly probable in the proximity of obstacles, e.g. when driving in urban areas. Despite the fact that there are high-quality GNSS receivers with included integrity monitoring [16], satellite-based or ground-based augmentation, multi-GNSS as combining GPS, Galileo, Glonass etc. [14], multi-antenna devices, and of course coupled navigation with inertial sensors [9], signal reception quality is still crucial. With regard to positioning performance requirements [6], automatic driving in urban canyons, car parks, or within dense traffic flow is a challenge with high demand on additional navigation technologies.

Parts of this work have been supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) within the project *Cooperative Mobility in the Digital Test Area Düsseldorf* (KoMo:D), Grant Agreement No. 16AVF1006H.

Next to GNSS/INS, wheel odometry, visual lane detection, position matching to known maps, or adaptive cruise control based on distance sensing are common technologies used for automatic driving [2] and state-of-the-art for driving assistance functions [5]. However, not all cases are covered in terms of safety and reliability, preventing navigation being certified for highly automated or driverless use.

Another navigation approach is based on the idea of self-location relative to the environment using exteroceptive sensors as cameras or distance sensors and by re-finding previously measured features. In contrast to the above-mentioned techniques, arbitrary environment features can be used, i.e. there is no special need to rely on road markers, signs, or communication infrastructure. Being generally based on remote sensing ideas, the use for navigation arose in robotics. This class of technology comprises visual odometry, simultaneous localization and mapping (SLAM), or optical-aided navigation, depending on the used algorithm and special application. One advantage is that neither a-priori knowledge about the environment nor satellite positioning are required, however it can be combined to that. Typical applications are focused on mapping/exploration or on positioning/navigation; and they are various: from indoor-capable vehicle or robot navigation [18], augmented reality [12], micro air vehicles [8], and last but not least automatic car driving [13]. As a background technology, SLAM can be considered as state-of-the-art including its known shortcomings (especially position error accumulation and dependency on remarkable stationary objects), however reliable use to control a car in dynamic traffic has to be proven.

2 Towards SLAM Navigation in Urban Traffic

From its origins in robotics [3], SLAM found its way into automatic driving research as one of the navigation technologies to complement GNSS. A recent survey can be found in [1]. Together with research results, gigabytes of (raw) data are nowadays publicly available, and they are widely used for further algorithm development, optimization, and benchmarking. Next to a variety of camera data listed e.g. in [17], the popular KITTI dataset [7] as well as the Udacity dataset [4] provide LiDAR data which can be used to test the presented type of SLAM. While standard benchmarks are highly effective for evaluation of perception algorithms, they typically do not allow evaluation of closed-loop systems. Therefore, it was chosen to generate own data in this paper. In contrast to the majority of other papers in this field, no new algorithm (or an optimization of an existing one) will be presented, and the paper is not aimed at the reduction of the total position error e.g. at the end of the test drive. Instead, typical problems that arise within the drive are picked and discussed with an eye on the use for vehicle control. The evaluation data comes from urban driving tests with four LiDARs on a car (Fig. 1) driven through city traffic.



Fig. 1. Testing vehicle: Volkswagen e-Golf operated by DLR.

3 Experimental Setup

3.1 Vehicle and Sensors

Testing vehicle is an electric powered Volkswagen Golf operated by DLR's Institute of Transportation Systems. As other vehicles from the institute's fleet [11], it includes some modifications for experimental and user-defined vehicle automation and control. The modular and exchangeable equipment comprises multiple environment sensors (LiDAR, cameras, RADAR) GPS/INS with RTK capability, on-board computers for data logging, real-time data processing and vehicle control as well as various communication interfaces (WiFi, cellular, ITS-G5). Details of the sensor processing hardware are listed in Table 1.

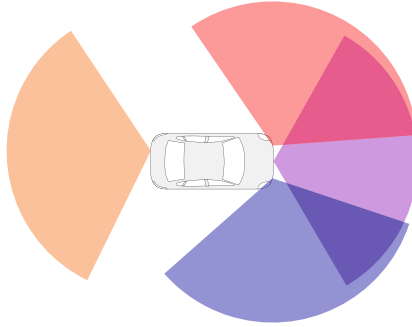
Table 1. Details of the sensor data processing computer.

Type	Vecow ECX-1200 embedded PC
CPU	Intel Core i7-8700T, 2.4 GHz
GPU	NVidia GTX 1050 Ti, 4 GB RAM
RAM	8 GB, DDR4-2133
Hard drive	512 GB, SSD
Interfaces	6x GigE LAN, 4x RS-232/422/485, GPIO, etc.
Operating system	Ubuntu 16.04 LTS, ROS kinetic

The current testbed design considers integration into future off-the-shelf vehicles such that e.g. roof sensor racks are not used. Relevant sensors are four LiDAR sensors (see Table 2), mounted as shown in Fig. 2 so that almost a full 360° horizontal field of view is achieved. Multi-sensor integration includes extrinsic calibration and time synchronization so that data is aligned.

Table 2. LiDAR sensor specification.

Type	ibeo LUX 4L
Field of view	$110^\circ \times 3.2^\circ$, 4 horizontal scan lines
Range	>50 m
Range accuracy	0.1 m (range independent)
Frame rate	25 Hz

**Fig. 2.** LiDAR sensor setup: three at front, one at back.

As ground truth reference for SLAM evaluation, satellite navigation data is recorded together with LiDAR data during the driving tests. The device used is a NovAtel SPAN solution with RTK capability and included GPS/INS navigation filter which returns 100 Hz pose updates.

3.2 Sensor Data Processing

The major processes run within the Robot Operating System [15] including sensor drivers and data logging. As one of the components, SLAM is performed by Google Cartographer software [10]. This application can combine multiple laser range finders and is capable of generating map and pose updates in real-time. In this context, pose output (position x, y and heading ψ) is of main interest for the navigation purpose discussed in this paper. The general processing architecture is shown in Fig. 3.

The LiDAR driver returns sensor raw data which consists of a point cloud for every sensor, and multiple LiDAR point clouds are merged to scan frames for every time step. They can be optionally aligned by use of inertial data. Local matching performs registration between the scan frames and data fusion into a map, i.e. an occupancy grid. As the SLAM software is capable of matching local sub-maps into larger maps which are globally consistent (i.e. loop closure), it can optionally create globally consistent trajectory output at cost of trajectory smoothness. This is naturally only possible when previously mapped areas are entered again.

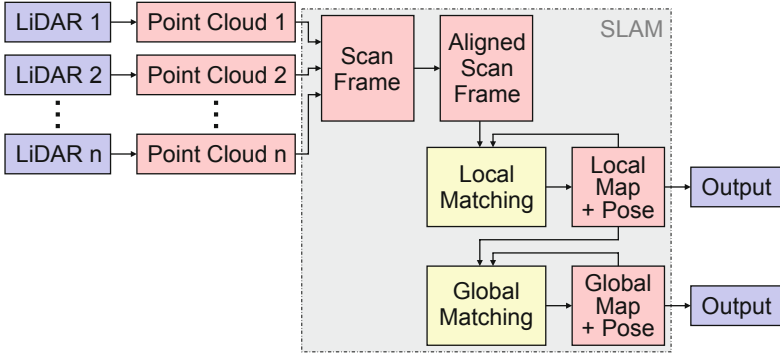


Fig. 3. Simplified SLAM architecture.

3.3 Testing Methodology

To evaluate different SLAM settings with comparable input, SLAM is executed in post-processing using sensor raw data recorded during test drives. Both data recording and SLAM are processed on on-board hardware (Table 1) and with real-time data playback so that the test comes close to live execution with regard to frame rates, timing issues, and possible frame drops. Some SLAM parameters are tuned to achieve good results, however the parameter choice is tested on different input sequences to be generally usable for the presented vehicle setup. The major parameters with non-default values are listed in Table 3.

The setup uses Cartographer’s 2D map and trajectory builder with only LiDAR point cloud inputs, and without inertial or odometry data such that raw SLAM behavior will be evaluated. In the beginning, the map is empty, meaning unknown occupancy. The vehicle pose is initialized with $(x, y, \psi) = (0, 0, 0)$. At this point, true heading is close to but not exactly north. The map is empty, meaning unknown occupancy. For ground truth comparison, GPS/INS poses are transformed into the local coordinate system.

4 Driving Test

Data recording is performed during a 10-minute drive in the city center of Düsseldorf, Germany. The ride goes over two laps around a rectangular city block with close start and ending points. Figure 4 shows the mapping progress as a part of SLAM to re-calculate the vehicle’s trajectory. The pictures show snapshots at different points of the drive, visualizing how the map is filled with obstacle boundaries (black) and free space (white) from the LiDAR point clouds.

5 Data Evaluation

With the used onboard computer, processing of 25 Hz LiDAR data is completely real-time capable. At default grid map resolution of 25 cm, typical delays of

Table 3. Trajectory builder parameters within Cartographer.

Software version	Release 1.0.0
min_range	1.0 m
max_range	40 m
min_z	-1.0 m
max_z	5.0 m
num_accumulated_range_data	4
ceres_scan_matcher.occupied_space_weight	2.0
ceres_scan_matcher.translation_weight	10.0
ceres_scan_matcher.rotation_weight	15.0
motion_filter.max_distance_meters	1.0 m
motion_filter.max_angle_radians	0.2 deg
submaps.resolution	0.25 m
range_data_inserter.hit_probability	0.58
range_data_inserter.miss_probability	0.48

about 5–10 ms (except jitter outliers) between LiDAR input and pose output are observed, which may allow even higher frame rates. Hence, there is much reserve for the use of higher grid resolution (see Sect. 5.3), more environment sensing, or filtering methods, e.g. integration of inertial and wheel sensors. In the next subsections, pose estimation results based on the driving test are presented.

5.1 Whole Trajectory

Figure 5 shows a top view of the SLAM-based position estimation. The used metric coordinates are related to UTM grid coordinates and thus slightly rotated to the illustration in Fig. 4. Together with Fig. 6 providing position and heading estimation over time, the plots show the two laps around the city block, one calculated with enabled SLAM loop closure, and one calculated without. Error accumulates as expected when using SLAM without any GNSS or external map hints. It can be seen that loop closure has no effect until the end of the first full circle, which is also expectable since no previously mapped areas are present until then. At the end of the sequence, SLAM accumulates a position error of about 4 m with loop closure, and 14 m without. With a total path length of about 1300 m, this equals $\approx 1\%$ error compared to the distance driven. The plots do also show the typical problems of GNSS in the proximity of obstacles. There are immediate jumps of up to 3 m, which is obviously not the true driving trajectory.

5.2 Selected Data Snapshots

Now, specific driving situations are of interest. Beginning with the sequence start, and thus, the sometimes critical SLAM initialization phase, Fig. 7 compares easting of GPS and SLAM over time. At this point, no special problems

are detected, i.e. error accumulation is present but as expected, and no time delays can be observed.

Next point of interest is the standing car after driving some meters. Figures 8 and 9 show easting and heading where the values should be truly constant (vibrations are supposedly very low because of electrical engine). Here, some differences are visible: The GPS/INS easting coordinate (and also northing not shown) is affected by GPS-typical jumps and secondly updates, being even less stationary than the SLAM coordinate which is only noisy. The visible 5 cm variation is about 1/5 of the grid resolution of 25 cm. In contrast to that, GPS/INS heading tends to be very stable, while SLAM is downgraded by a 0.5° noise.

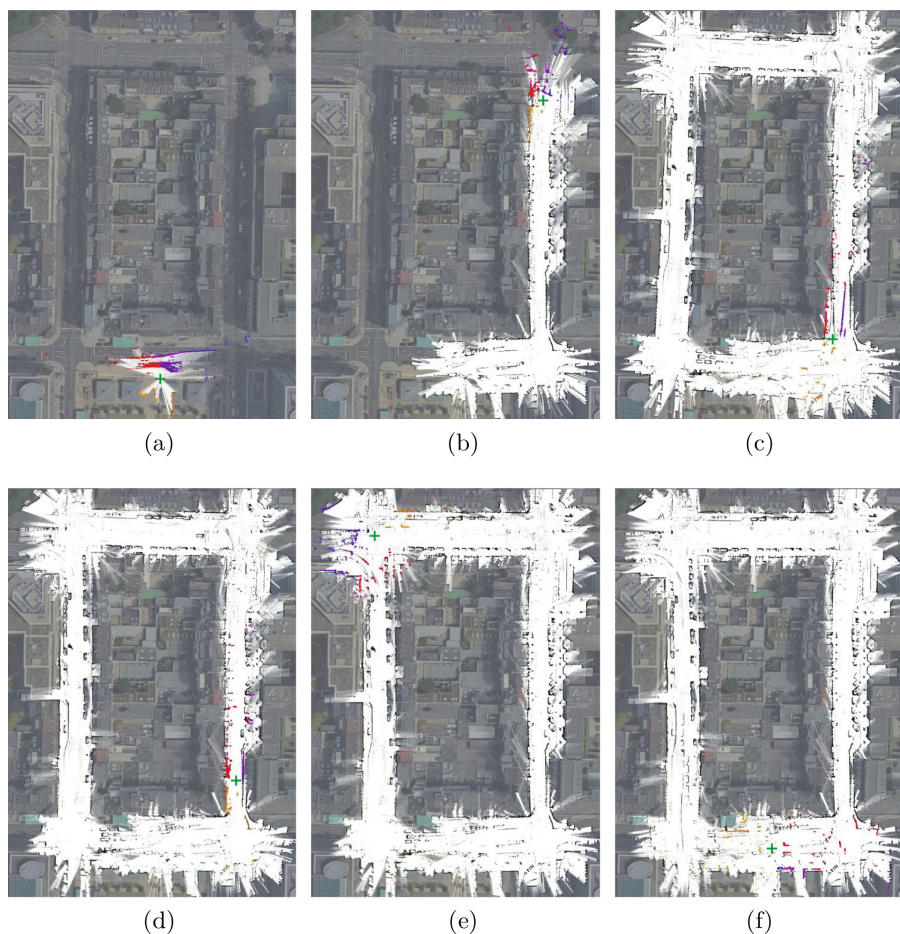


Fig. 4. SLAM results on drive within Düsseldorf city. Occupancy map, vehicle position (+), LiDAR scans (colors as in Fig. 2): (a) starting position, (b) second corner, (c) before and (d) after first loop closure, (e) re-finding corner example, (f) end of drive. Background image: GeoBasis-DE/BKG (2009), Google.

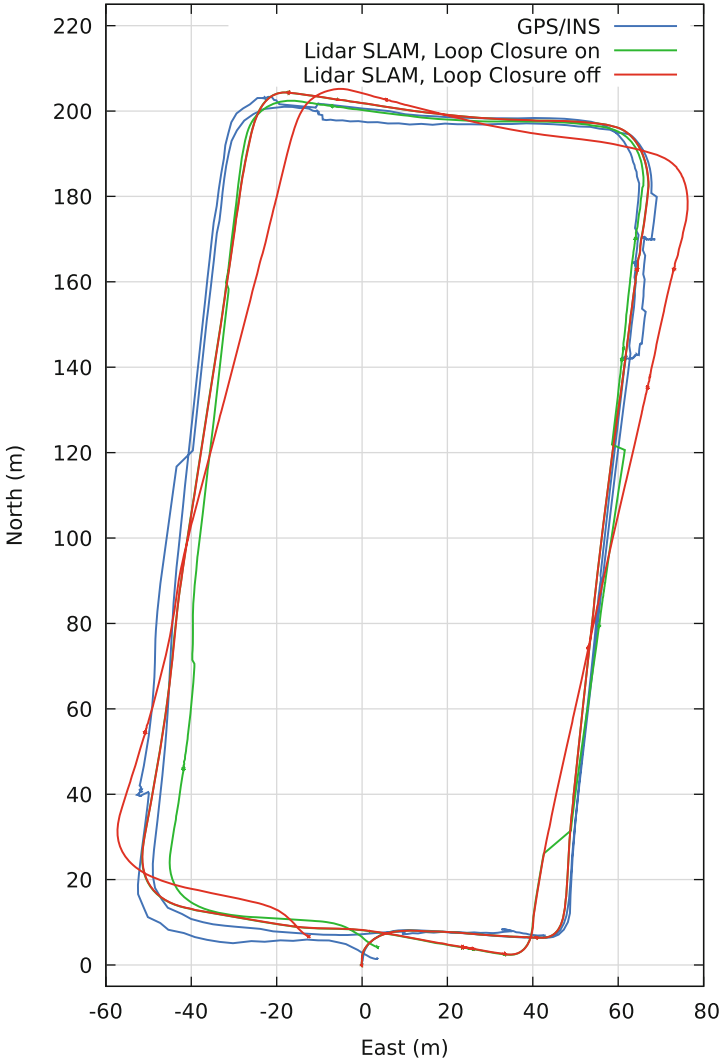


Fig. 5. Trajectories, local xy -view, Düsseldorf dataset.

Zooming out (Fig. 10) shows a wider view of this driving phase where the car waits at an intersection. Although several moving cars are in the sensors' field of view, being mapped and replaced by free space again, suitable positioning is achieved. The GPS position jump at 112s might be the larger issue here.

At a later turn, the car does not have to stop at the intersection. As visible in Figs. 11 and 12, the curvature is reconstructed within SLAM, however there are already some absolute errors accumulated.

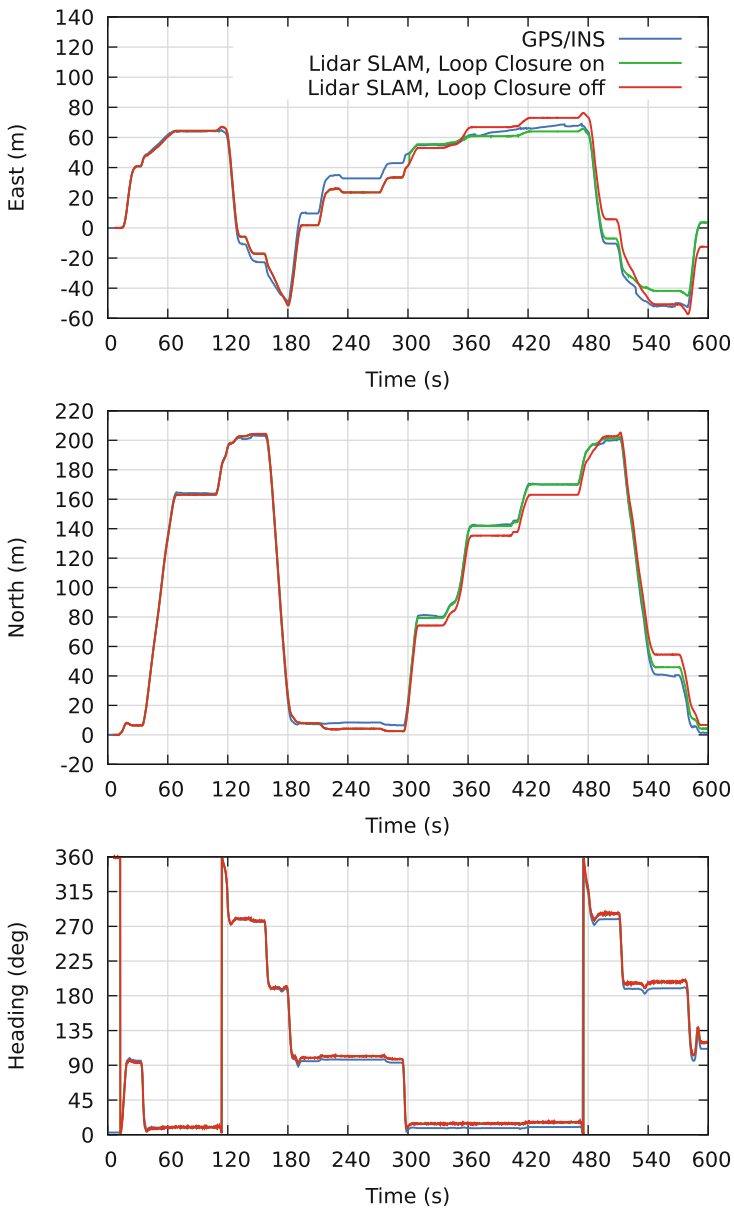


Fig. 6. Time plots, Düsseldorf dataset.

Finally, this evaluation takes a closer look at two specific points. The first is the point of first global correction (loop closure) at around 301s from the beginning. Figure 13 shows exemplarily the easting coordinate correction towards GPS/INS ground truth. Northing and heading are corrected in a similar way. A

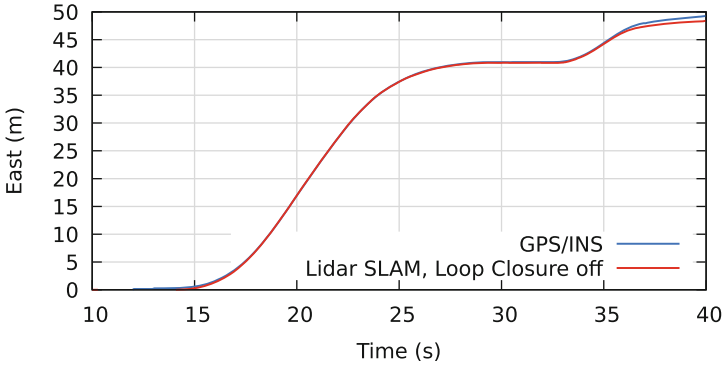


Fig. 7. Start of drive, east coordinate. (Before loop closure takes effect, SLAM with loop closure *on* is equivalent to SLAM with loop closure *off*, thus this graph is not shown in the particular figures.)

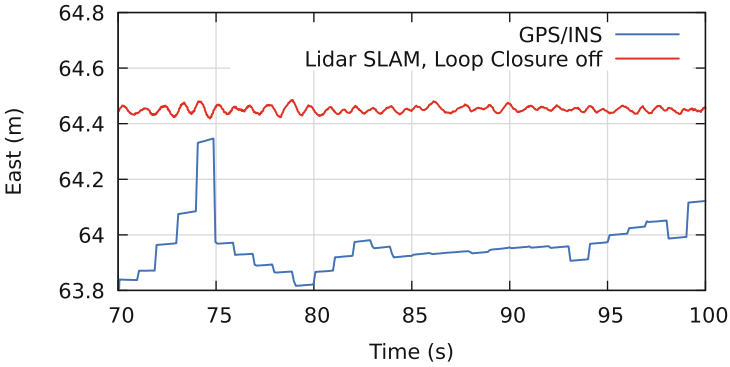


Fig. 8. Standstill, east coordinate.

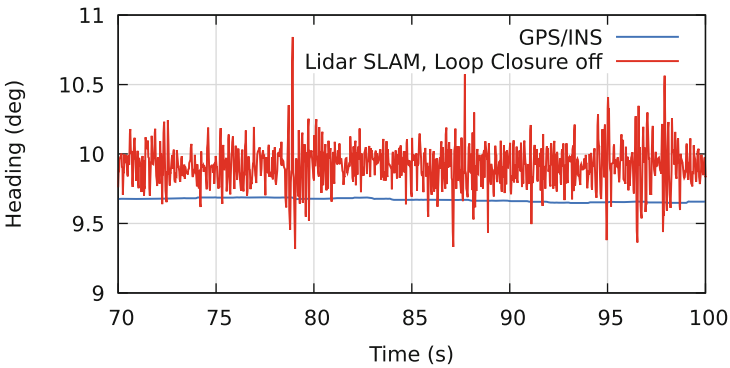


Fig. 9. Standstill, heading coordinate.

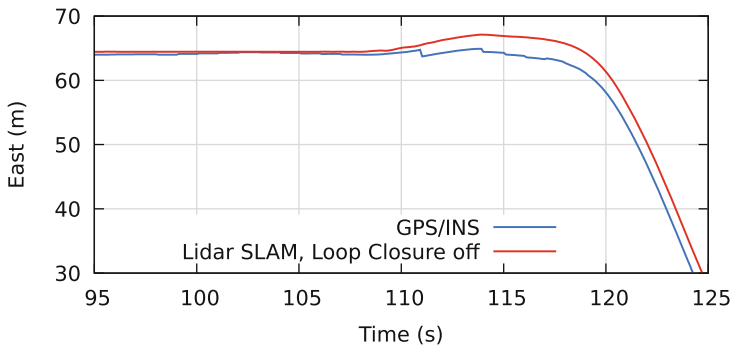


Fig. 10. Intersection waiting and curve, east coordinate.

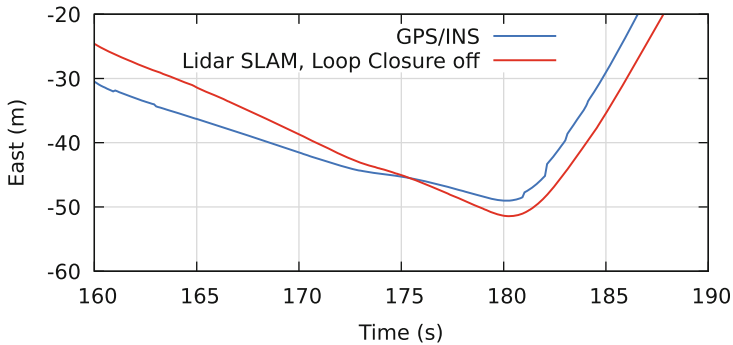


Fig. 11. Driving curve, east coordinate.

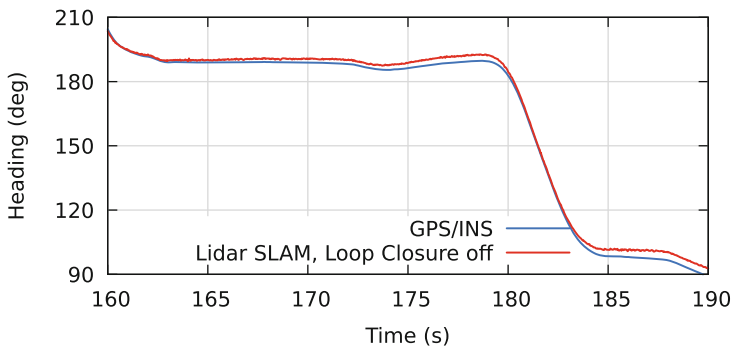


Fig. 12. Driving curve, heading coordinate.

jump of around 10 m is however critical when being used for control algorithms. Since the second lap begins now with driving within an already mapped area, later global corrections apply, too. Since they do not mark a loop closure in such a manner, the consequent position jumps are much smaller.

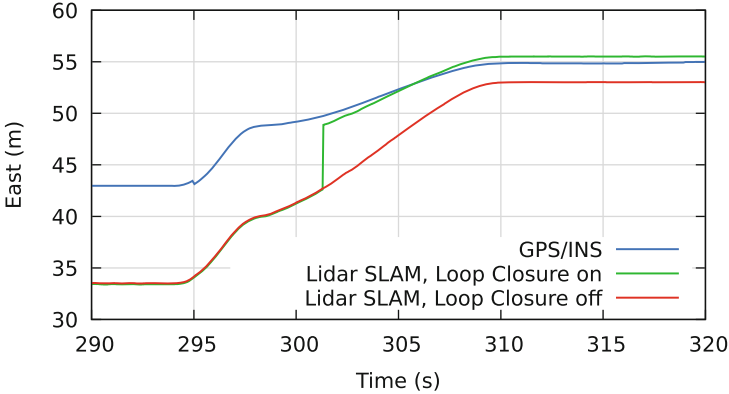


Fig. 13. Loop closure point, east coordinate.

The last close view is again more in the beginning of the data sequence where the car is standing while waiting at an intersection. The point of interest is timing at a very detailed zoom shown in Fig. 14. Ground truth data is available as specified: at 100 Hz without remarkable jitter. If the computer is fast enough, this specific SLAM implementation provides updates as fast as possible (which are also up to 100 Hz here). To provide meaningful information, only SLAM-based positions with new LiDAR data included are plotted. The 25 Hz updates are then visible in the plot, however some jitter is shown. Altogether, this seems to be very good and suitable for control inputs when the raw local SLAM output without loop closure is used.

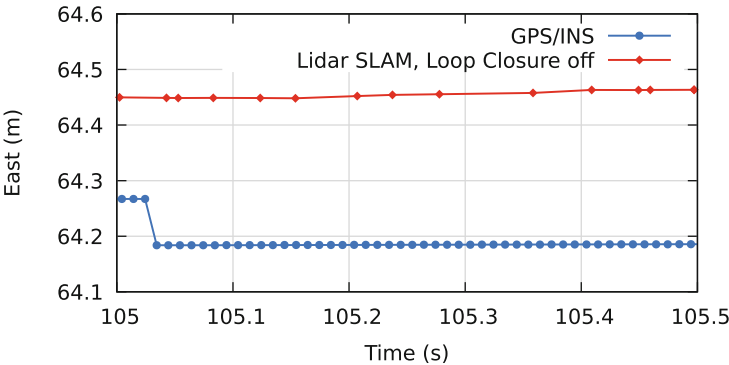


Fig. 14. Single data frames, east coordinate.

5.3 Different SLAM Settings

As listed in Table 3, all the above calculations are based on a grid resolution of 25 cm. This is one major parameter to increase or decrease depending on the available computation power. To test the effects, SLAM was run again with grid resolutions of 10 cm and 50 cm. The main result is generally comparable mapping and localization, as expected with increased or decreased accuracy. Processing time correlates quadratic with grid resolution (i.e. linear with number of grid cells). With the used hardware, real-time processing and loop closure are achieved in all cases. Major difference seems to be local positioning uncertainty, visible as noise at standstill. The plots in Figs. 15 and 16 extend the previous plots (Figs. 8/9). They show an effect on translational and rotational noise with somehow linear correlation to grid resolution. Further, total position and heading error accumulation is also lower at higher grid resolutions.

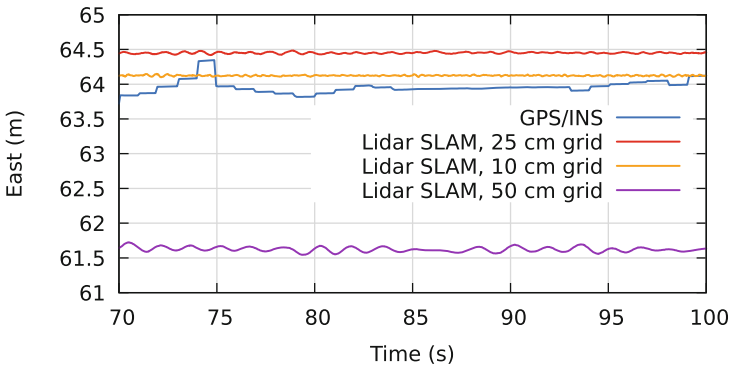


Fig. 15. Standstill, east coordinate. Extension of Fig. 8 with different grid resolutions.

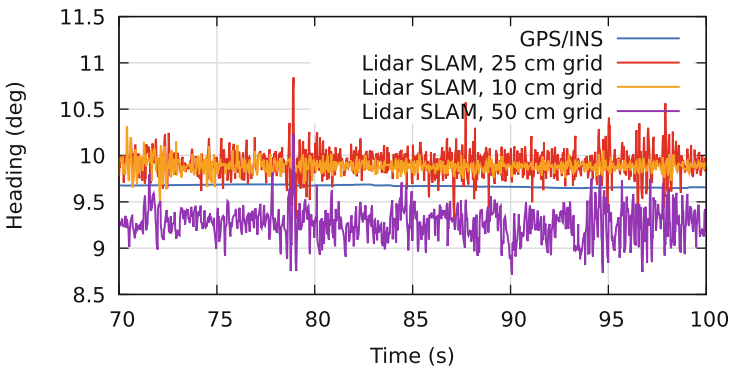


Fig. 16. Standstill, heading coordinate. Extension of Fig. 9 with different grid resolutions.

5.4 Other Datasets

To prove reproducibility, a different dataset is evaluated. Figure 17 shows the SLAM trajectory estimation result from a ride with the same car and sensor configuration around the Automotive Campus in Helmond, Netherlands. Parameters are equal as above, with a grid resolution of 25 cm.

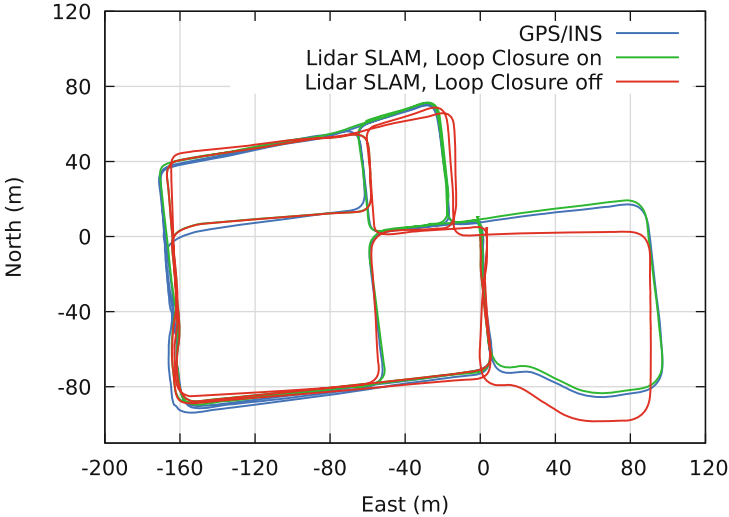


Fig. 17. Trajectories, local xy -view, Helmond dataset.

6 Conclusion and Future Work

In situations where satellite positioning and also techniques as lane detection and map matching are subject to fail, SLAM can be a reasonable complement towards safe vehicle navigation. This paper has a special focus on noise to and robustness in urban car driving, and it can be shown that up-to-date SLAM software is capable of finding its way in dynamic road traffic situations. Future work will investigate situations where SLAM is subject to fail without any optimizations or intelligent switching between different techniques. For example, stationary objects that start to move (e.g. congestion) might disturb the map and thus localization if their amount is too high. Other work will focus on specific applications, e.g. automated valet parking at very close distances when there is no need to open doors anymore.

References

1. Bresson, G., Alyased, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: a survey of current trends in autonomous driving. *IEEE Trans. Intell. Veh.* **2**(3), 194–220 (2017)

2. Urmson, C., et al.: Autonomous driving in urban environments: boss and the urban challenge. *J. Field Robot.* **25**(8), 425–466 (2008)
3. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. *IEEE Robot. Autom. Mag.* **2**(13), 99–110 (2006)
4. Gonzalez, E., Higgins, M., Cameron, O., et al.: The udacity open source self-driving car project (2016). <https://github.com/udacity/self-driving-car>
5. European global navigation satellite systems agency: report on road user needs and requirements - outcome of the European GNSS' user consultation platform. Tech. rep. GSA-MKD-RD-UREQ-233537 (2018)
6. European telecommunications standards institute: satellite earth stations and systems (SES); GNSS based location systems; part 3: performance requirements. Tech. rep. ETSI TS 103 246–3 (V1.1.1) (2015)
7. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. *Int. J. Robot. Res. (IJRR)* **32**(11), 1231–1237 (2013). <http://www.cvlibs.net/datasets/kitti>
8. Hening, S., Ippolito, C., Krishnakumar, K., Stepanyan, V., Teodorescu, M.: 3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments. In: *AIAA SciTech Forum* (2007)
9. Henkel, P., Sperl, A.: Precise RTK positioning with GPS/INS tight coupling and multipath estimation. In: *International Technical Meeting of the Institute of Navigation*, pp. 1015–1023 (2016)
10. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2D LiDAR SLAM. In: *IEEE International Conference on Robotics and Automation*, pp. 1271–1278 (2016). <https://github.com/googlecartographer>
11. Kaschwich, C., Wölfel, L.: Experimental vehicles FASCar-II and FASCar-E. *J. Large-Scale Res. Facil.* **3**, 111 (2017)
12. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: *International Symposium on Mixed and Augmented Reality* (2007)
13. Levinson, J., Thrun, S.: Robust vehicle localization in urban environments using probabilistic maps. In: *IEEE International Conference on Robotics and Automation*, pp. 4372–4378 (2010)
14. Li, X., Zhang, X., Ren, X., Fritsche, M., Wickert, J., Schuh, H.: Precise positioning with current multi-constellation global navigation satellite systems: GPS, GLONASS, Galileo and BeiDou. *Sci. Rep.* **5**, 8328 (2015)
15. Quigley, M., et al.: ROS: an open-source robot operating system. In: *ICRA Workshop on Open Source Software* (2009). www.ros.org
16. Wang, J., Ober, P.B.: On the availability of fault detection and exclusion in GNSS receiver autonomous integrity monitoring. *J. Navig.* **62**, 251–261 (2009)
17. Yin, H., Berger, C.: When to use what data set for your self-driving car algorithm: an overview of publicly available driving datasets. In: *IEEE International Conference on Intelligent Transportation Systems* (2017)
18. Zhang, J., Singh, S.: LOAM: lidar odometry and mapping in real-time. In: *Robotics: Science and Systems Conference* (2014)