# 3D SLAM With Scan Matching and Factor Graph Optimization

Thomas Emter and Janko Petereit
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB,
Fraunhoferstr. 1, 76131 Karlsruhe, Germany
{thomas.emter, janko.petereit}@iosb.fraunhofer.de

## Abstract

For autonomous navigation, a mobile robot needs the capability to estimate its pose while simultaneously mapping its environment. This contribution presents an approach for fusing data from multiple asynchronous sensors using factor graphs. Full 3D SLAM is performed on data from several localization sensors and point clouds from a 3D LiDAR. The scans from the LiDAR are integrated by scan matching for relative motion estimation and are also used for loop closure.

## 1 Introduction

It is an essential capability of a mobile robot to be able to map its environment with its sensors. Because all sensors of the robot are subject to noise and inaccuracies, this is a great challenge. In the beginning the map and the localization will not yet be precise and their errors are mutually interdependent. In order to build a correct and precise map while providing an accurate localization, their uncertainties and mutual dependencies have to be modeled and accounted for. Methods and algorithms to solve this problem are well known as Simultaneous Localization and Mapping (SLAM) [1].

In order to improve the precision of the localization and the map, multiple sensors can be combined by means of sensor fusion. The fusion has the advantage that errors stemming from the sensor measurements can be reduced and even mitigation of sensor outages is possible. In case of 6DoF (degrees of freedom) localization and mapping, more than one sensor is needed to ensure observability of all degrees of freedom.

In this contribution, a factor-graph-based approach to combine sensor fusion of a GPS, an inertial measurement unit (IMU), and wheel odometry with scan matching of point clouds from a 3D LiDAR scanner is presented. From the IMU measurements, a relative motion can be derived, whereas GPS provides absolute measurements. Scan matching of two consecutive LiDAR scans also yields a relative measurement. An existing framework for factor graphs is used to include all measurements and a vehicle model to perform full 6DoF localization and 3D mapping simultaneously. As the sensors mounted on a mobile robot have different data rates and are not synchronized, the factor graph structure is exploited to model estimation problems with multiple asynchronous sensors. The paper provides an evaluation of the overall performance and robustness against sensor failures with regard to the impact of different sensor configurations. In order to embed the factor graph in a system for on-line localization, a parallel filtering and smoothing scheme is presented and evaluated with regard to its timing properties.

The following section gives an overview over related work. Afterwards, the SLAM scheme utilizing factor graphs is explained with subsections covering scan matching, loop closure, and map building. A section presenting the results is followed by a conclusion closing the paper.

## 2 Related Work

Extended Kalman filters were successfully used to solve the SLAM problem for a long time [2]. They were extended to track multiple hypotheses in order to be more robust against wrong data associations [3]. For 2D SLAM, particle-filter-based algorithms have been shown to be very effective as they implicitly model multiple hypotheses, e.g. [4]. Each particle establishing a different estimate of the path and the map. 3D SLAM features six instead of three degrees of freedom (6DoF) and much more particles and thus increased processing power is needed. In addition, the maps need more memory in 3D, rendering particle filters more and more infeasible.

Graph optimization has been used for 2D SLAM for a long time [5]. For 3D SLAM, mostly graph-based methods are used because of the aforementioned limitations of the filtering approaches concerning memory and computational effort [6].

In [7] it is shown that if the variance of the heading remains small, EKF-based 2D SLAM is robust and able to build consistent maps. In graph-based methods, the inclusion of loop closures is straightforward by means of introducing additional constraints into the graph, but falsely detected loop closures may lead to inconsistent maps [8]. By integrating a sensor with absolute measurements like GPS, the loop closure can be made even more robust [9].

For estimation of the relative pose between scans from a 3D LiDAR sensor, several scan matching algorithms have been developed. The *Velodyne SLAM* algorithm extracts surfaces from the scans to efficiently estimate the poses. Thus, the algorithm is tailored for scenarios where planes exist, which makes it well suited for urban scenarios [10]. Methods like

*iterative closest point* (ICP) are based on matching points and make no assumptions on the scenario. Hence, they are better suited for off-road or heterogeneous environments [11]. The *Generalized-ICP* (GICP) algorithm is a variant especially developed for data originating from scanning sensors [12].

# 3 Factor Graph SLAM

Besides reducing the resulting errors, fusion of multiple sensors like GPS and scan matching increases the robustness by compensating for each other's weaknesses. For example, GPS may be very heavily disturbed in confined areas [13] whereas the precision of scan matching is very good in areas with many objects. In flat areas with open sky view GPS is very reliable but scan matching may not be very effective due to lacking structures.

Compared to filtering approaches which are based on the Markov assumption, in graph optimization, past states can be re-estimated. This is very important for loop closures, because in this case the robot returns to a place after a long time and corrections can be back-propagated, i.e., errors accumulated along the path can be corrected and consistent maps be built.

Factor graphs represent the estimation problem as a graphical model [14] and [15]. A factor graph being a bipartite graph has two types of nodes: variable nodes $x_i$ and factor nodes $f_j$. Since the graph is bipartite, the edges are only between factor nodes and variable nodes. Sensor measurements and other constraints are represented by factor nodes and the state estimates by variable nodes. Solving the SLAM problem equals to estimation of the full joint probability $p(X|Z)$ over all states $x_i$ given all sensor measurements or constraints $z_j$. The joint probability can be represented by a factor graph as each factor $f_j$ encodes a measurement likelihood $p(z_j|X_j)$ with $X_j$ being all state variables $x_i$ involved in factor $f_j$, i.e., being connected via edge $e_{ij}$.

The presented SLAM algorithm is implemented using the GTSAM framework as back-end. It provides a convenient factor graph structure as well as different optimization algorithms [16]. It also provides a factor for IMU preintegration. Since an IMU measures at a very high rate, multiple consecutive measurements can be preintegrated into a single factor to reduce node count and therefore computational complexity [17] and [18]. As the IMU has a significantly higher rate than the other sensors, this preintegration also allows for a synchronization to the other sensor measurements by combining the IMU measurements for the according periods of time. Although the IMU has a much higher data rate than the other sensors, an exact synchronization cannot be assured. Therefore, in our approach the IMU preintegration is extrapolated upon the inclusion of a new node based on the current measurement data. The interval of the next following preintegration of new IMU measurements is shortened accordingly.

The presented SLAM algorithm is primarily used for the estimation of the 6DoF pose, but also the linear velocity and IMU biases are estimated; the latter for internal reasons. As the IMU biases change slowly over time, they can be estimated at a lower rate.

**Figure 1** shows an example for a factor graph, where the variable nodes $x_n$ are shown as large cyan circles and the factor nodes as small circles. For clarity, the variable nodes of the linear velocity and IMU biases are not shown. GPS measurements $g_m$ are prior factors colored in green, possessing only one edge. IMU measurements $i_{n-1,n}$ are between-factors modeling a relative motion from state $x_{n-1}$ to state $x_n$ and are colored in magenta. The relative motion of the scan matching $s_{n-1,n}$ is also modeled as between-factor from state $x_{n-1}$ to state $x_n$ and colored in yellow.

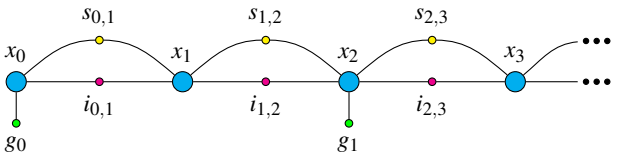



**Figure 1** Illustration of a factor graph with variable nodes $x_n$ as large cyan circles and the factor nodes as small circles.

The measurements from wheel odometry and the vehicle model are also omitted from the figures for the sake of clarity. The wheel odometry is modeled by another between-factor, which introduces a constraint in the 2D plane attached to the vehicle coordinate system. The vehicle model is also a between-factor incorporating the kinematic constraints of the vehicle. While the factor of the wheel odometry connects two nodes according to its measurement interval, the vehicle model factor connects each consecutive pair of nodes. The latter means that each time a new node is added to the graph, a vehicle model factor is also added connecting the new node with its predecessor.

## 3.1 Scan Matching

With scan matching, the relative motion between two consecutive scans of a 3D LiDAR scanner can be estimated. Utilizing scan matching for relative motion estimation can be denoted LiDAR odometry. In the presented SLAM approach, the Generalized-ICP (GICP) is used for scan matching [12]. It extends the well-known ICP algorithm by a probabilistic model in the minimization step taking into account the locally planar structure. It can be interpreted as local plane-to-plane matching, which is better suited for point clouds originating from moving 3D LiDAR scanners. Despite exploiting locally planar structures, the GICP is still well suited for unstructured scenarios and not reliant on strictly planar surfaces in the environment. The estimated relative motion has full six degrees of freedom and is included into the graph as between-factor $s_{n-1,n}$ connecting state $x_{n-1}$ with state $x_n$, cf. **Figure 1**.

## 3.2 Loop closure

Closing a loop, which happens when the mobile robot returns to a known place, is one of the biggest challenges in SLAM, because uncertainties accumulate over the traveled

path. When loop closures are not accounted for or loops are falsely detected and integrated, the resulting map will be inconsistent.

A potential loop closure has to be detected first. This is accomplished by comparing the current pose with past poses regarding their Euclidean distance. In order to avoid loop closures with poses in the immediate past, potential loop closures are omitted until a traveled distance $> d_{pre}$ is reached. The comparison is performed for poses predating this pose until the Euclidean distance is below a certain distance $d_{loop}$ with $d_{loop} < d_{pre}$. For a potential loop closure, the GICP is performed and its fitness score is evaluated to determine if a loop closure has occurred. This verification helps to avoid false positives, which may lead to inconsistencies in the resulting map [8]. A successful loop closure is integrated into the factor graph as between-factor $s_{0,n}$ connecting state $x_0$ with state $x_n$, see **Figure 2**. Thus, LiDAR scan matching is additionally used for loop closure besides relative motion estimation of consecutive scans, cf. Section 3.1.

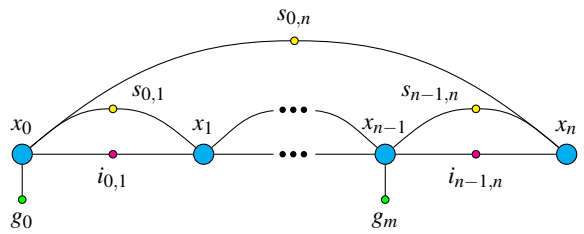



**Figure 2** Illustration of a loop closure between states $x_0$ and $x_n$.

## 3.3 Processing

The Velodyne has a scanning rate of 10 Hz, i.e., a rotation takes 100 ms. The vehicle moves during this time leading to a skewing of the scan; hence, the resulting scan has to be rectified. This is accomplished by inertial correction based on a high-rate 6DoF pose estimation.

After collecting all data and feeding it into the graph as values and factors, a full optimization by the Levenberg-Marquardt algorithm can be performed in the GTSAM framework. Of course, it is possible to also do intermittent optimizations, but the full optimization algorithm has to be performed all over again.

If intermittent updates are necessary, a more efficient alternative is to use the *iSAM2* algorithm for optimization [19]. It is able to perform incremental updates, thus reusing results from earlier runs. The efficiency is achieved, because new factors and values mostly affect only small parts of the graph. Thus, only a small part of the graph has to be optimized most of the time.

In [20] concurrent smoothing and filtering is proposed, combining the advantages of a full factor graph optimization with the on-line capabilities of a filter. In this paper, a similar scheme is introduced combining incremental iSAM2 smoothing or intermediate Levenberg-Marquardt optimization (LM) with a parallel on-line Extended Kalman filter (EKF). The EKF also estimates the full 6DoF pose and is explained in more detail in [21]. It can be used for the aforementioned inertial correction of the LiDAR scans.

For the parallel processing scheme, the implemented front-end algorithm incorporates three non-blocking and thread-safe data pipelines providing the sensor data to multiple parallel threads, as depicted in **Figure 3**. The first pipeline provides data for the thread of the graph optimization algorithm. As the graph optimization needs at least several measurements $z_t$, the first optimization starts at $t_1$ with the measurements from $t_0$ up to $t_1$. The second pipeline is an intermediate pipeline for a catch-up EKF and the third pipeline feeds the on-line EKF at full rate. For the EKFs, all sensor data except LiDAR data for scan matching is used. The reason for omitting the scan matching in the on-line part is the processing time needed for the GICP, which would compromise the on-line capability due to the introduced latency.

Since the factor graph optimization takes some time (from $t_1$ to $t_2$ or from $t_2$ to $t_4$), the availability of the result lags behind the on-line EKF. Therefore, the intermediate catch-up filter is initialized with the result of the graph optimization algorithm (at $t_2$ and $t_4$, cf. green arrows in **Figure 3**) and fed with data from the second pipeline until it is in sync with the on-line EKF. When synchronized to the on-line EKF, the state of the catch-up filter is copied to the on-line EKF (at $t_3$ and $t_5$, cf. blue arrows in **Figure 3**). Thus, the on-line EKF is still able to run at its full rate, while additional precision from the scan matching is also introduced to the on-line EKF by the synchronization. All mentioned algorithms are running in parallel threads to ensure best performance and minimal interaction between the updates only for synchronization purposes. Besides the three threads processing the pipelines, there are other threads, e.g., for filling the pipelines with new data or for logging data for evaluation.

## 3.4 Mapping

While integrating new data into the factor graph, mapping is only preformed implicitly, as the LiDAR data is only used for scan matching up to this point. An explicit map is built only after graph optimization. Mapping is accomplished by using the *Normal Distributions Transform* (NDT) representation [22].

To build an NDT map, the environment is divided into voxels. For each voxel a normal distribution over all points within the voxel is calculated, i.e., each voxel contains a 3D mean and a covariance matrix. By interpreting the normal distribution as its $\sigma$-ellipsoid, the shape of the environment can be modeled. A flat surface for example leads to a disk shaped ellipsoid.

To correct errors and cope with noise or dynamic objects, which should not be integrated into the map, a method for taking information about free space into account is included. An existence probability is additionally attached to each voxel. The existence probability is updated by each observation, i.e., increased by a point falling into a voxel and decreased while tracing the laser ray from its origin through free space to that voxel. In the voxel map, this is accomplished with a 3D Bresenham algorithm [23]. The

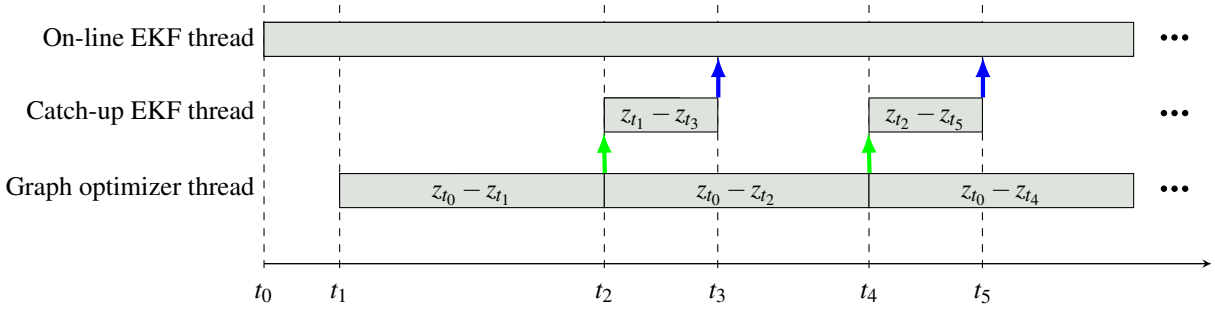



**Figure 3** Timing diagram of the parallel filtering and smoothing scheme.

normal distribution of a voxel with its existence probability falling below a certain threshold is eliminated.

# 4 Results

Due to the lack of ground truth data, the following results are qualitative evaluations and quantitative comparisons. An *Xsens MTi-G-700* delivers IMU and GPS measurements with rates of 100 Hz and 4 Hz respectively. The 3D LiDAR data is from a *Velodyne HDL-64E*, which has a scanning rate of 10 Hz, i.e., a rotation takes 100 ms. The wheel odometry sensor is custom-made and has a rate of 10 Hz.

## 4.1 Full SLAM

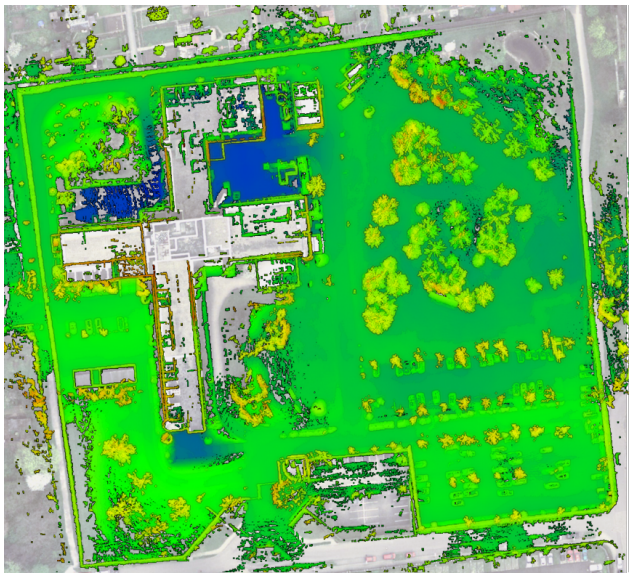


**Figure 4** NDT map over a satellite image of the area. The height is color-coded from blue (low) via green to red (high).

The resulting map after a full optimization is shown in **Figure 4** over a satellite image of the area. By inclusion of GPS measurements, the map is implicitly geo-referenced. The vehicle traveled for about 1.8 km with multiple loop closures. The cross shaped building in the left half was surrounded two times. After the second turn, the parking lot in the right bottom corner was driven trough and finally the area in the top right corner with trees and bushes was covered. Thus, the tour covered both urban scenarios and unstructured environments.

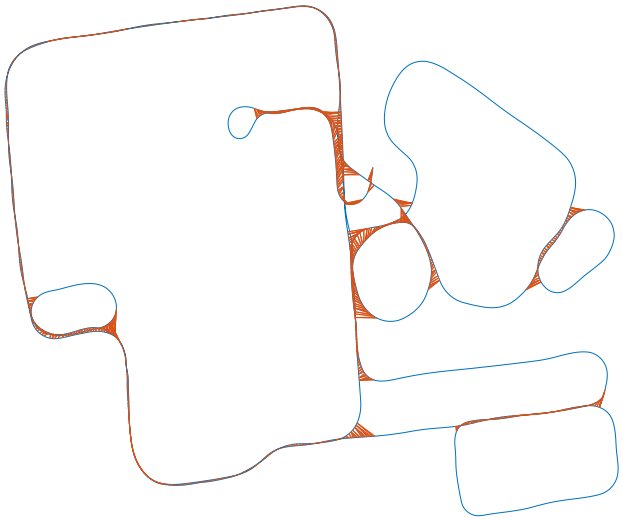


**Figure 5** Multiple loops; path in blue and loop closure constraints in red.

**Figure 5** shows the path in blue with multiple loops. The loop closure constraints are depicted in red. The large loop was traveled twice, which can be seen by the continuous loop closure constraints along the path.

### 4.1.1 On-Line Processing

The final path estimated incrementally with the iSAM2 algorithm is very close to the full optimization with the Levenberg-Marquardt algorithm. The maximum positional difference in 3D is 1.06 m; the maximum horizontal position difference is only 0.06 m. In **Figure 6** it can be seen that there exists a nearly constant offset of the final path estimated with iSAM2 (blue curve). As can be seen by the maximum horizontal position difference of a few centimeters, this offset is mainly in the altitude. The altitude measured by GPS is much more inaccurate compared to latitude and longitude resulting in a higher variance [24]. The difference of the on-line localization with the parallel EKF with parallel iSAM2 optimization is up to 12.08 m and 2.29 m on average. With regard to only the horizontal error, the maximum difference is 5.25 m and 0.78 m

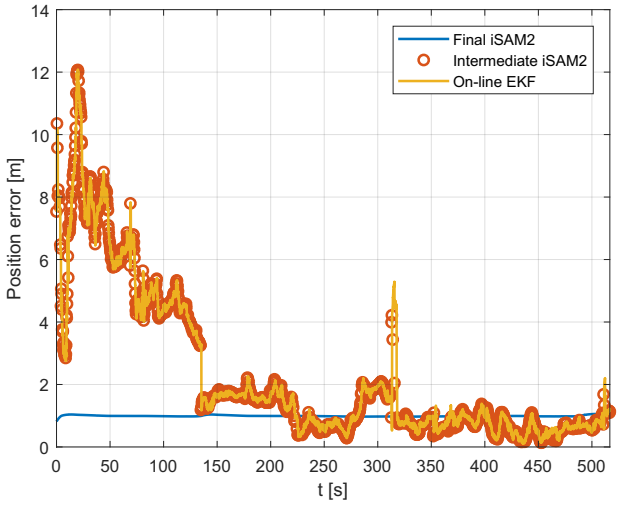
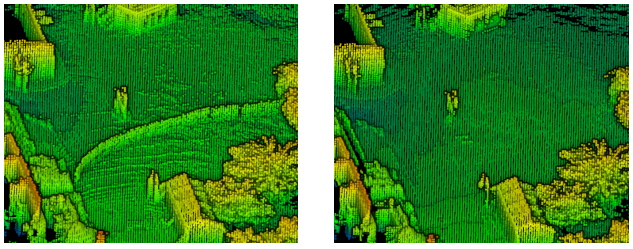



**Figure 6** Position errors of on-line localization compared to full optimization with Levenberg-Marquardt.

on average. The differences with a parallel intermediate Levenberg-Marquardt optimizer are very similar.
Although the difference of the final optimized path estimate is much lower, the intermediate and incrementally computed graph estimates also have high differences along the path. These incremental estimates are used to initialize the on-line EKF; see red circles in **Figure 6**. The differences are highest in the beginning before the first loop closure, which occurs at about 135 s and has a length of about 550 m, cf. **Figure 6**. After the first loop closure, the differences are much lower and closer to the final curve of the iSAM2 estimation.

## 4.2 Dynamic Environment

**Figure 7** shows a scenario with a walking person. When using plain mapping, the person walking from right to left is added permanently to the map, cf. **Figure 7(a)**.



(a) (b)

**Figure 7** Scenario with a walking person. Normal mapping (a); taking negative evidence into account (b).

By taking information about free space into account, the person will only be in the map for a short time and subsequently removed by negative evidence. If the area has been scanned before and enough negative evidence has been accumulated, the person might even not be added to the map at all. Thus, the person will not be in the final map, cf. **Figure 7(b)**. Comparing both figures it can be seen that without taking negative information into account some clutter is visible on the ground-level surface, whereas in **Figure 7(b)** the surface is flat and nearly artifact-free.

## 4.3 Simulated GPS outage

In order to evaluate the robustness against sensor failures an outage of the GPS for a period of about 35 seconds or 130.7 m was simulated.

**Table 1** Horizontal position errors in m

| | $e_{\text{min}}$ | $e_{\text{avg}}$ | $e_{\text{max}}$ |
|---|---|---|---|
| IMU | 0.01 | 1.38 | 4.48 |
| IMU+Odo | 0.07 | 0.93 | 2.51 |
| IMU+Odo+Model | 0.12 | 0.71 | 1.17 |
| IMU+SM | 0.18 | 0.76 | 1.58 |
| IMU+SM+Odo+Model | 0.06 | 0.28 | 0.62 |

In **Table 1** the horizontal position errors in the area of the GPS outage for different sensor configurations can be found. The errors are calculated compared to the result of using all sensors and no GPS outage. If only an IMU is present during the GPS outage, the errors amount to about 4.5 m. Additional inclusion of odometry can already reduce the average and maximum error significantly. Adding the vehicle model to the aforementioned sensors or alternatively using scan matching helps to further reduce the average and maximum errors comparably. Including all available sensor and model data, the errors are the lowest as expected.

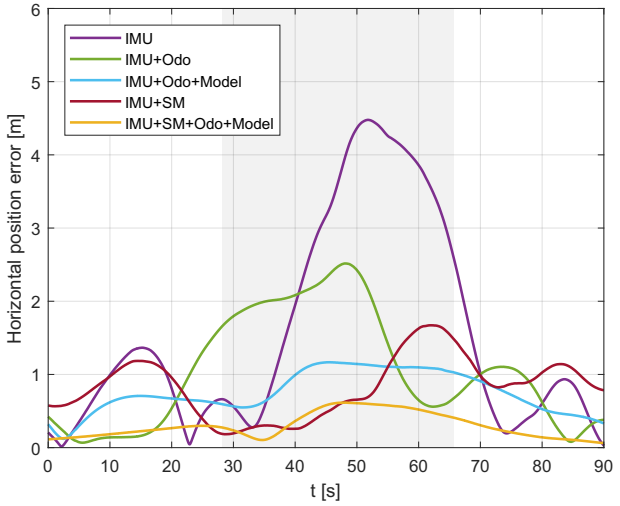



**Figure 8** Horizontal position errors. The period of GPS outage is shaded in gray.

**Figure 8** shows the horizontal position errors in the area of the GPS outage. The period of GPS outage is shaded in gray. It can be seen that the error using only IMU is the highest during the outage. The smoothing property of the graph optimization leads to smooth transitions and no hard jumps can be seen at the start or the end of the GPS outage. On the other hand, this leads to errors being present outside the GPS outage even when using all available sensor and model data.

Because the altitude is the most error prone as mentioned in Section 4.1.1, the 3D position error is evaluated separately for comparison.

**Table 2** 3D Position errors in m

| | $e_{\text{min}}$ | $e_{\text{avg}}$ | $e_{\text{max}}$ |
|---|---|---|---|
| IMU | 0.30 | 2.04 | 4.52 |
| IMU+Odo | 1.18 | 2.12 | 3.92 |
| IMU+Odo+Model | 0.56 | 1.60 | 2.81 |
| IMU+SM | 0.18 | 0.84 | 1.67 |
| IMU+SM+Odo+Model | 0.18 | 0.36 | 0.62 |

In **Table 2** the 3D position errors in the area of the GPS outage for different sensor configurations can be found. The errors are again calculated compared to the result of using all sensors and no GPS outage. If only an IMU is present during the GPS outage the errors are also the highest and amount to more than 4.5 m. In the 3D case, additional odometry only reduces the maximum error a little because it only introduces a relative constraint in a 2D plane. Adding the vehicle model to the aforementioned sensors or alternatively using scan matching helps to reduce the average and maximum errors significantly, because both are relative 3D constraints. Including all available sensor and model data, the errors are the lowest like in the 2D case as expected.
**Figure 9** shows a comparison of the resulting maps in the area, where the GPS outage occurred. In **Figure 9(a)**, where only an IMU was present during the outage, the map clearly shows strong distortions due to large errors during the outage. Both, the addition of wheel odometry with a vehicle model (**Figure 9(b)**) and scan matching (**Figure 9(c)**) greatly improve the robustness against GPS outages and the precision of the mapping result. **Figure 9(c)**) shows the reference map without GPS outage.

## 4.4 Timing

The timing assessment was conducted on a Xeon 2687W v2 CPU with eight cores running @ 3.4 GHz.
Along the path of about 1.8 km more than 65,000 measurements of all sensors were captured. Full optimization with Levenberg-Marquardt in GTSAM including all measurements and model data takes about 11.5 s.
In **Table 3** the timings of the parallel updates of the concurrent filtering and smoothing scheme are listed.

**Table 3** Timings of Updates

| | $t_{\text{min}}$ | $t_{\text{avg}}$ | $t_{\text{max}}$ | $\sigma_t$ |
|---|---|---|---|---|
| EKF update [µs] | 11.0 | 79.1 | 7598.0 | 74.9 |
| iSAM2 update [ms] | 12.0 | 382.8 | 9996.3 | 219.1 |
| LM update [ms] | 27.8 | 350.0 | 11525.3 | 273.4 |
| Catch-up filter [ms] | 0.2 | 15.5 | 105.8 | 22.2 |

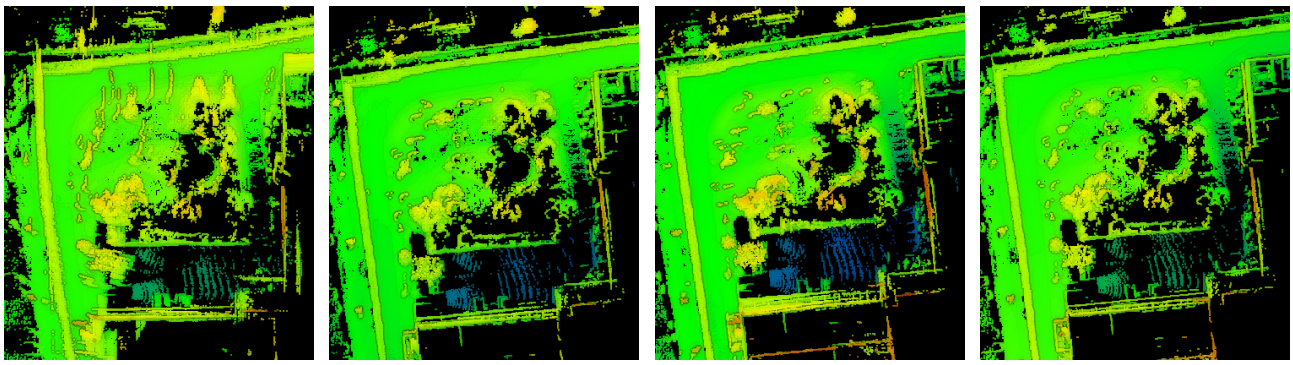
The on-line EKF updates only take 79.1 µs on average. This also includes the copy operation from the catch-up filter for synchronization. The maximum processing time of about 7.6 ms seems quite high and presumably stems from spurious delays of the operating system because only 35 of more than 65,000 updates took more than 1 ms. The iSAM2 updates take less than 400 ms on average but may take nearly 10 s, which is nearly as high as the full optimization with Levenberg-Marquardt. The timings of the continuous Levenberg-Marquardt updates are very similar. This can be explained by the front-end algorithm, which generates very many loop closure constraints. Loop closure constraints have a great impact on the efficiency of the iSAM2 regarding incremental updates, as these constraints have influences on large parts of the graph. Thus, the capability of iSAM2 of updating only small parts of the graph cannot be utilized most of the time. The subsequent catch-up filtering also takes several ms. Given the processing time of the on-line EKF well below 10 ms, the parallel structure is suitable for on-line operation.
The insertion of one LiDAR scan with approximately 120,000 3D points into the NDT map takes about 7 ms plus 32 ms for including negative information. Building the whole map (**Figure 4**) with about 5,000 scans and taking negative information into account takes roughly 200 seconds. When operating in environments where no dynamic objects are to be expected or only mapping over short time intervals is necessary, negative information may be omitted resulting in a more than fivefold speedup.

# 5 Conclusion

In this paper an evaluation of factor graphs for 6DoF SLAM utilizing the GTSAM framework was presented. The robustness of the algorithm against sensor failures has been evaluated and compared for different sensor configurations. It has been shown that the algorithm successfully mitigates simulated GPS outages and that aiding the IMU with additional sensors or a vehicle model greatly improves the localization and mapping.
For mapping, a scheme based on NDT maps accounting for information about free space was introduced. This helps to filter dynamic objects from the map and allows for error corrections.
A proposed parallel on-line concurrent filtering and smoothing scheme was compared to the results of a full optimization and its timing was evaluated demonstrating its on-line suitability.

# ACKNOWLEDGMENT

The described research forms part of the project entitled “AKIT – Autonomy kit for near-serial-production work vehicles for networking and assisted rescue from safety hazards”, which is being promoted in the course of the “Innovative rescue systems” announcement of the BMBF within the scope of the German government’s “Research for civil safety” program.



(a) (b) (c) (d)

**Figure 9** Mapping in the area of the GPS outage with IMU and GPS only (a); with IMU, GPS, wheel odometry, and vehicle model (b); with IMU and scan matching (c); reference map without outage using IMU, GPS, wheel odometry, and vehicle model (d). The height is color coded from blue (low) via green to red (high).

## References


[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachusetts: The MIT Press, 2005.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I", *IEEE Robotics & Automation Magazine*, vol. 13, 2 2006.

[3] ——, "Simultaneous localization and mapping: Part II", *IEEE Robotics & Automation Magazine*, vol. 13, 3 2006.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling", in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[5] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping", *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[6] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching", *Robotics and Autonomous Systems*, vol. 56, no. 2, 2008.

[7] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm", in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3562–3568.

[8] J. S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments", in *Computational Intelligence in Robotics and Automation, 1999. CIRA '99. Proceedings. 1999 IEEE International Symposium on*, 1999, pp. 318–325.

[9] T. Emter, "Integrated Multi-Sensor Fusion and SLAM for Mobile Robots", *Proceedings of the 2011 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*, J. Beyerer and A. Pak, Eds., 2012.

[10] F. Moosmann and C. Stiller, "Velodyne SLAM", in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, 2011, pp. 393–398.

[11] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm", in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[12] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP.", in *Robotics: Science and Systems*, vol. 2, 2009.

[13] M. S. Braasch, "Multipath", in *Springer Handbook of Global Navigation Satellite Systems*, P. J. Teunissen and O. Montenbruck, Eds., Cham: Springer International Publishing, 2017, pp. 443–468.

[14] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Factor graph based incremental smoothing in inertial navigation systems", in *2012 15th International Conference on Information Fusion*, 2012.

[15] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing", *Robotics and Autonomous Systems*, vol. 61, no. 8, 2013.

[16] *GTSAM*, `https://bitbucket.org/gtborg/gtsam`, Accessed: 2018-01-03.

[17] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors", in *Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation", in *Robotics: Science and Systems*, 2015.

[19] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree", *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[20] S. Williams, V. Indelman, M. Kaess, R. Roberts, J. J. Leonard, and F. Dellaert, “Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing”, *Int. J. Rob. Res.*, vol. 33, no. 12, Oct. 2014.

[21] T. Emter and J. Petereit, “Integrated Multi-Sensor Fusion for Mapping and Localization in Outdoor Environments for Mobile Robots”, in *Proceedings SPIE 9121, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2014*, 2014.

[22] M. Magnusson, *The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection*. Örebro Universitet, 2009.

[23] J. E. Bresenham, “Algorithm for computer control of a digital plotter”, *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[24] J. Kouba, F. Lahaye, and P. Tétreault, “Precise point positioning”, in *Springer Handbook of Global Navigation Satellite Systems*, P. J. Teunissen and O. Montenbruck, Eds., Cham: Springer International Publishing, 2017, pp. 723–751.