2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Exploiting Building Information from Publicly Available Maps in Graph-Based SLAM

Olga Vysotska Cyrill Stachniss

***Abstract*— Maps are an important component of most robotic navigation systems and building maps under uncertainty is often referred to as simultaneous localization and mapping or SLAM. Most SLAM approaches start from scratch and build a map only based on their own observations and odometry information. In this paper, we address the problem of how additional information can be exploited, for example from OpenStreetMap. We extend the standard graph-based SLAM formulation by relating the nodes of the pose-graph with an existing map. As this paper suggests, we can relate the newly built maps with information from publicly available maps with the laser range finder data from the robot and in this way improve the map quality. We implemented and evaluated our approach using real world data taken in urban environments. We illustrate that our extension to graph-based SLAM provides better aligned maps and adds only a marginal computational overhead.**

I. Introduction

Maps are needed for wide range of robotic applications and most navigation systems rely on grid maps or feature maps. Computing such maps under uncertainty is often referred to as the simultaneous localization and mapping or SLAM problem. Over the last 25 years, a large variety of different solutions to this problem have been prosed in the robotics community.

An intuitive way to model the SLAM problem is via a graph. Solving a graph-based SLAM problem requires to construct a graph with nodes and edges. The nodes represent robot poses or landmark locations. An edge between two nodes represents a sensor measurement that constrains the two connected pose. In most cases, we have a larger number of edges than nodes and as a result of that, the constraints will be contradictory as observations are always affected by noise. The goal of the optimization step in graph-based SLAM consists of finding a configuration of the nodes such that the overall graph is maximally consistent with the observations. This involves solving an error minimization problem with a large number of unknowns. Given this methodology robot can construct the map by exploiting their own sensor data.

In addition to maps built for and from mobile robots, there are also other maps available. This includes Google maps, Open Street Maps (OSM), or maps from the cadastre. Such maps are typically made for human users, but they provide relevant information for most outdoor environments on earth. Therefore, several researchers investigate means for exploiting this information, for example for localizing

Olga Vysotska and Cyrill Stachniss are with Institute for Geodesy and Geoinformation, University of Bonn, Germany. This work has partly been supported by the European Commission under the grant number FP7-610603-EUROPA2.

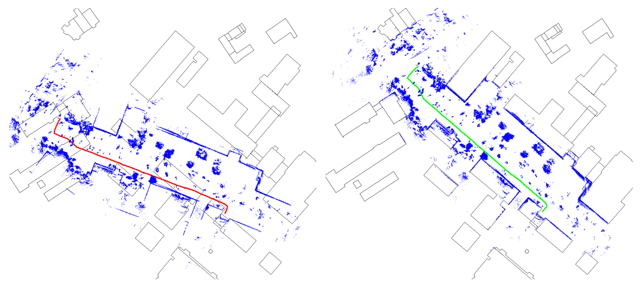

Fig. 1: Including the OpenStreetMap information to align robot's trajectory. Left: trajectory before alignment. Right: trajectory after alignment.

a robot in Open Street Maps [8] or using Google Street View [1], [17]. Although this information is frequently used for localization, it is seldom used for augmenting the mobile robot SLAM problem[1].

The main contribution of this work is to incorporate building information from existing maps into the pose-graph-based SLAM systems, see Fig. 1 for a small example. Our method considers building information from OpenStreetMap, cadastre information, or similar sources in the optimization process in order to produce better maps of the environments. We fuse the information about the layout of building with the sensor data recorded by the robot. We derive additional constraints for the pose-graph that align the robot's poses with the given map. This yields a better alignment so that the robot's perceptions could also be used to extend the maps. In our evaluation, we show that we can (i) improve the map alignment with our approach, (ii) can handle the situations in which the map data is partially outdated, for example if buildings have been demolished or new buildings have been built, and (iii) all operations yield only a small computational overhead compared to a standard graph-based SLAM system.

II. Related Work

The first work in robotics that addressed SLAM through least squares was the paper by Lu and Milios [16]. Subsequently, Gutmann and Konolige [11] focused on means for constructing such graph and for detecting loop closures. Over the last 15 years, a large number of different approaches to graph-based SLAM have been proposed, e.g.,[3], [9], [19] Most SLAM approaches assume Gaussian errors in the constraints. This renders them sensitive to data association outliers. A number of approaches has been proposed

[1]Although [8] is called OpenStreetSLAM, the approach performs localization on the road network provided by Open Street Maps.

to overcome this problem. For example, the approach of Sünderhauf and Protzel [20] scales the effect of potential outlier constraints while Agarwal *et al.* [2] proposed dynamic covariance scaling as an alternative scaling approach that does not increase the number of variables that need to be optimized.

Recently, several localization approaches were proposed that use the information from OpenStreeMaps [12] to improve robot localization. Most of them incorporate this information into the observation model of the Monte Carlo localization. For example, Hentschel *et al.* [13] represent buildings as 2D line features. This line map is then used to calculate the expected range measurement at a certain robot's locations and combines MCL with a form of Kalman filtering. In our work, we also use the information about buildings, but integrate the correspondences through an ICP-based matching procedure into a graph-based SLAM framework. Another approach, which is proposed by Floros *et al.* [8] uses champfer matching to align robot's trajectory with the road network extracted from publicly available maps. Each particle in MCL is weighted according to the reported champfer matching cost. Ruchti *et al.* [18] also use the information about the road network. Instead of relying on visual odometry as Floros *et al.*, they classify the 3D laser scans into road/non-road surfaces and the classification result is incorporated into the weight of the particles in the Monte Carlo localization. In contrast to that, our work incorporates buildings information obtained from OpenStreetMap into the graph-based SLAM as additional edge constraints.

Global consistency can also be achieved by aligning the robot's perceptions to aerial images as proposed by Kümmerle et. al [15]. The authors extract the features from the 3D point clouds and match them against planar structures extracted by Canny edge detection from the aerial images. Thus, this is related to our approach, which exploits building information from OSM. Kümmerle et. al furthermore apply Monte-Carlo Localization to localize the robot in the aerial images before generating a pose graph for solving the SLAM problem.

III. Graph-Based SLAM Exploiting Existing Maps as Background Knowledge

The optimization step in graph-based SLAM systems aims at finding the configuration of the nodes that minimizes the error induced by observations. We consider pose-graphs, i.e., SLAM graphs in which the nodes correspond to robot poses. This yields a state vector $X = (x_1, \ldots, x_n)^\top$ where x_i is the pose of node i. The error function $e_{ij}(X)$ for a single constraint between the nodes i and j is often the difference between an expected measurement $f(x_i, x_j)$ and the obtained measurement z_{ij}:

$$e_{ij}(X) \;=\; e_{ij}(x_i, x_j) \;=\; f(x_i, x_j) - z_{ij}. \quad (1)$$

Note that alternative representations can be used to avoid problems resulting from singularities in the angular components, see [10] for details.

As the error functions are typically non-linear, we linearize $e_{ij}(X)$ around the current best estimate

$$e_{ij}(X + \Delta X) \;\simeq\; e_{ij}(X) + J_{ij}\Delta X. \quad (2)$$

Here, J_{ij} is the Jacobian of the non-linear error function computed in the current state. The resulting minimization problem can be written as

$$X^* \;=\; \operatorname*{argmin}_X \sum_{ij} e_{ij}(X)^\top \Lambda_{ij} e_{ij}(X), \quad (3)$$

where Λ_{ij} is the information matrix associated to a constraint. Up to this point, this is the standard formulation of pose-graph SLAM.

A. Error Function Exploiting Existing maps

Eq. (3) is the error function to minimize in standard pose-graph SLAM. The main goal of our approach is to incorporate additional knowledge into the optimization process and relate the pose-graph to existing data. To achieve this, we extend the error function

$$X^* = \operatorname*{argmin}_X \sum_{ij} e_{ij}(X)^\top \Lambda_{ij} e_{ij}(X) + F^{map}(X), \quad (4)$$

where $F^{map}(X)$ is the error introduced by the mismatch between the robot's observation and the map information. Analogous to pose-pose constraints, we split up the component $F^{map}(X)$ into individual constraints between robot poses and the openStreetMap information:

$$F^{map}(X) \;=\; \sum_i e_i^{map}(X)^\top \Lambda_i e_i^{map}(X). \quad (5)$$

The key elements in Eq. (5) are the error function $e_i^{map}(X)$ and corresponding information or weight matrix Λ_i.

In the remainder of this section, we describe how to construct an error function $e_i^{map}(X)$ and Λ_i. Intuitively, the error function adds an additional constraint to the graph that anchors a pose of the robot to the specific location in the map. The key challenge here is to make the correct data association between the map and the robot's own sensor readings, obtained from the pose stored in the node of the pose-graph. Once this data association is solved and the correct coordinate transformations between the robot's poses and the map are computed, least square error minimization will provide us with the global alignment.

To make the data association between the map and the robot's poses, we use the building information in the map and the data from a 2D or 3D laser range finder installed on the robot. We do not consider the road network information in this work, as we do not want to restrict the robot to navigating on streets but allow for operating on sidewalks, foot paths, or in pedestrian zones. When aligning laser range data with the building information from OSM, a central challenge is that the laser scanner observes a large number of objects in the scene that are not stored in the map. Examples for such objects, which are not present in the publicly available map, are trees, cars, pedestrians. Therefore, we focus on large planar object seen by the laser scanner and

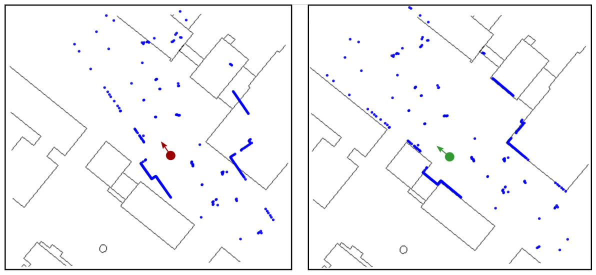

Fig. 2: An example of the robot's pose correction based on the aligning of the scan (blue) to the buildings in the map (black). Left: initial position of the robot. Right: position reported by ICP.

neglect most small or non-planer objects in the scene for the alignment.

B. Error Function Exploiting Building Information for Robots Equipped with Laser Range Scanners

In our work, we use the information about the buildings' geo-locations to enable the robot to take paths, independently from the road network. We obtain the building information directly from OpenStreetMap, which can be downloaded in form of an XML-file. Inside this file, the individual buildings are stored as separate nodes. Each node is a closed polygon describing the geo-referenced walls of the building, which directly yields a map of lines that shows the walls of the buildings in the environment. See the black polygons in Fig. 2 for an example.

As mentioned before, the laser scanner typically provides the scan of the environment covering a large number of objects that are not buildings and does that at a comparably high level of detail. This may hinder the matching procedure to make the correct data association between map and laser scan. Therefore, we filter the range scans so that most of the non-building objects are removed.

We investigated several techniques and in the end opted for an unsupervised approach that performs filtering based on line extraction and this does not require manually labeled training data and can be executed efficiently. We employ the Douglas-Peuker algorithm for converting the raw 2D range scan into a polyline. We convert the polyline into a set of potentially disconnected lines based on two parameters: the length of a line and the number of laser end points assigned to each line. In our current implementation, we maintain only lines with a length of at least 5 m containing at least 100 end points (for a scanner with a 0.25 deg resolution). This clearly eliminates also end-points belonging to walls, but overall, it keeps the number of false-positives small — which is more important for us in order to obtain a robust alignment between laser scan and map.

Since our aim is to incorporate the knowledge about the environment from the map into the graph optimization procedure to refine the robot's trajectory, the error function for this constraint should reflect the misalignment between the current robot's pose and the map. Intuitively, the bigger is the misalignment between the scan and the buildings in the map, the bigger the error should be. To estimate the (mis-)alignment, we use the Iterative Closest Point (ICP) [5] algorithm to match between the current laser scan and the map of building. For finding the correspondences in ICP, standard nearest neighbor data association is applied.

More precisely, the error term $e_i^{map}(x_i)$ is defined based on the pose difference between the current robot's pose x_i and the pose $\hat{x}_i$, computed by aligning the scan in the building map. The state of the robot x_i consists of translational t_i and rotational θ_i components, i.e., forms an element in $SE(2)$. The same holds for the state $\hat{x}_i$. Thus, we can express the error function as:

$$e_i^{map}(x_i) = \begin{pmatrix} \hat{R}_i^\top (t_i - \hat{t}_i) \\ \theta_i - \hat{\theta}_i \end{pmatrix}, \tag{6}$$

with $\hat{R}_i$ being the standard 2D rotation matrix corresponding to the angle $\hat{\theta}_i$.

For the ICP algorithm to operate reliably, one needs to have a good initial guess. In our setup, the initial guess is achieved by either manually specifying the first pose of the robot on the map or by using an initial guess from a consumer GPS. The initial guess of all successive poses is then automatically obtained from the odometry constraints of the graph or by incremental scan to scan alignment typically used in graph-based SLAM with laser range finders.

The Jacobian J_i of the error function given in Eq. 6 is

$$J_i = \frac{\partial e^{map}(x_i)}{\partial x_i} = \begin{pmatrix} \hat{R}_i^\top & 0 \\ 0 & 1 \end{pmatrix}. \tag{7}$$

Finally, we have to compute the information matrix Λ_i of a map constraint, which is the inverse of the covariance matrix of the ICP alignment, i.e., $\Lambda_i = (\Sigma_i^{ICP})^{-1}$. We compute the covariance matrix Σ_i^{ICP} from the ICP result by using Hessian method proposed by Bengtsson *et al.* [4]. An alternative method for propagating the noise from the scans to the covariance estimation of the robot's pose is described in [7] but we opted for the simpler approach by Bengtsson *et al.*. This method assumes the error function e^{ICP} used in the ICP algorithm to be quadratic near the optimal solution, i.e.,

$$e^{ICP} = \sum_k \|Tp_k - q_k\|^2, \tag{8}$$

where p_k are the point that belong to the detected buildings and q_k are the corresponding closest points in the buildings taken from the publicly available maps. The optimal transformation T that the ICP algorithm reports is found by minimizing the function e^{ICP} with the covariance matrix of T as

$$\Sigma_i^{ICP} = cov(T) = 2\sigma^2 \left(\frac{\partial^2}{\partial T^2} e^{ICP} \right)^{-1} = 2\sigma^2 H^{-1}, \tag{9}$$

where H is the Hessian matrix of e^{ICP} and σ^2 is the variance factor and depends on the sensor used.

So far, we described how to obtain the error function for 2D range data. We can also apply the same method using 3D range data. Here, we consider two setups. One is a

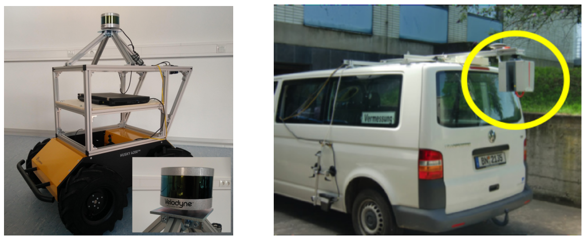

Fig. 3: Robots used in our experiments. Left: robot equipped with Velodyne VLP-16 laser scanner mounted parallel to the ground. Right: Profile scanner Z+F mounted in the back of a car.

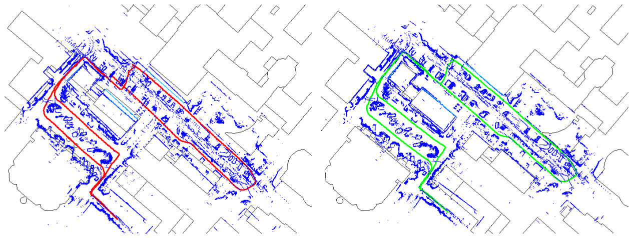

Fig. 4: Example of aligning the robot's trajectory with the buildings on the map and as a result of it improved loop closure, which also leads to more consistent robot map.

horizontally mounted 3D range scanner, here the 16-beam Velodyne Puck, and a 2D laser scanner using in profile mode as it is used in most surveying applications (see Fig. 3). For the Velodyne scan, we can basically perform the 2D procedure for the individual lasers (each ring) and merge the consistently found lines that may correspond to walls. For 3D point clouds obtained with a profile scanner, we build local point clouds and generate new virtual 2D laser scans given the planar surfaces in the cloud. We basically follow the approach proposed by Wulf *et al.* [21], which is also used in [6], [13].

C. Error Minimization

Given the error function e_i^{map} with corresponding information matrix Λ_i^{map} and Jacobian J_i, we can solve the optimization problem given in Eq. (3), for example using Levenberg-Marquardt. This yields to iteratively solving a linear system of the form

$$(H + \lambda I)\Delta X^* \quad = \quad -b, \tag{10}$$

with

$$H \quad = \quad \sum_{ij} J_{ij}^\top \Lambda_{ij} J_{ij} + \sum_i J_i^\top \Lambda_i J_i \tag{11}$$

and

$$b \quad = \quad \sum_{ij} J_{ij}^\top \Lambda_{ij} e_{ij} + \sum_i J_i^\top \Lambda_i e_i^{map}. \tag{12}$$

H and b are the key elements and are computed from the linearized error terms and λ is the damping factor used in Levenberg-Marquardt. The term ΔX^* refers to the increments that are added to the graph configuration in order to minimize our error function in the current iteration. In our implementation, we use the g2o framework [14] to conduct the minimization with dynamic covariance scaling [2] as a robust kernel. This yields an update of the graph configuration in every iteration of the form:

$$X \quad \leftarrow \quad X + \Delta X^*. \tag{13}$$

In practice, we do not execute this procedure in a batch fashion but incrementally or in small trajectory chunks (e.g., after 25 m of driving). This has two advantages. First, the data is available already online during mapping. Second, the correction of the trajectory up to a point in time t_1 will simplify the data association for the ICP step for subsequent matching with $t > t_1$ and, thus, has the potential to provide a better alignment. As a result of that, we obtain an optimized pose-graph that is aligned with the provided map.

IV. Experiments

The evaluation is designed to illustrate the performance of our approach and to support the three main claims made in this paper. These key claims are that (i) we can improve the map alignment with our approach, (ii) can handle the situations in which the map data is partially outdated, for example if buildings have been demolished or new buildings have been built, and (iii) all operations yield only a small computational overhead compared to a standard graph-based SLAM system. We furthermore illustrate that our approach is independent from the sensor setup and can successfully be used with 2D as well as 3D laser scanners.

A. SLAM Given Open Street Map Data

Our first set of experiments is designed to show that our approach exploits publicly available maps to locate the robot within these maps. Furthermore, by considering the individual robot's scans in alignment procedure, we may even find loop closures that are missed by the pose-graph SLAM otherwise. Fig. 4 shows a trajectory of the robot overlayed on the map when using traditional 2D graph-based optimization (red), i.e. without considering map information, and incorporating the map structure into the optimization process (green). As it can be seen, not only the robot's own map is better aligned with the structure of the environment, but also the loop closure was correctly detected due to the aligning laser scans to the buildings. Fig. 5 represents another example of the robot's trajectory, here using a 3D Velodyne data, which spans over a significantly larger area than the previous example. In Fig. 5, only the laser end points that do not belong to the ground plane are plotted. As it can be seen, the map produced by the robot is filled with substantial clutter in the environment, which makes the aligning procedure more challenging. Nevertheless, our approach is able to fix the misalignments that come from the inaccuracy of the initial position, see Fig. 5 middle column upper and bottom image, as well as loop closing error introduced by the pure pose-graph approach, see Fig. 5 middle center.

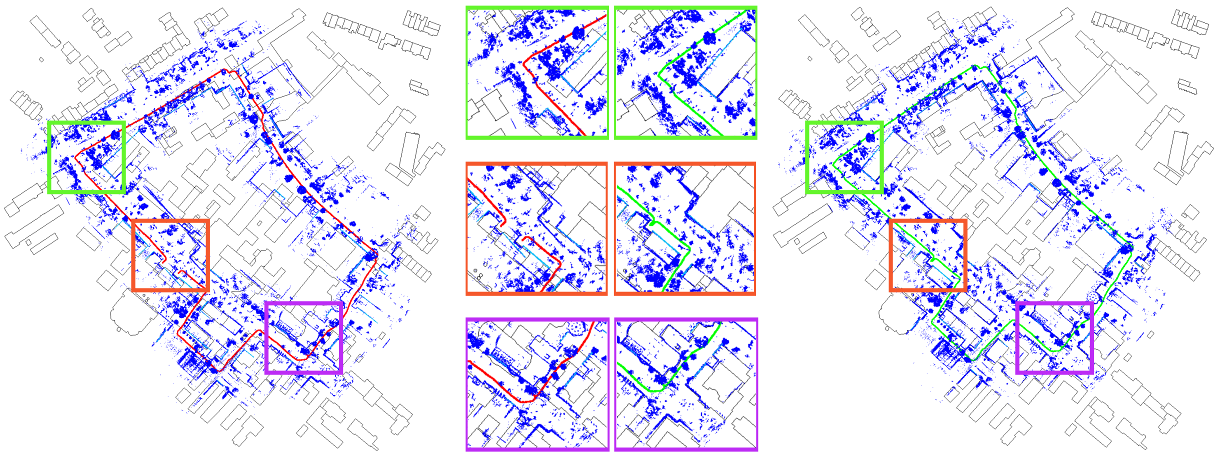

Fig. 5: Left: Overlayed trajectory before the optimization. Right: Trajectory after optimization. Middle: Zoom in parts of the trajectory; Top: Case of corrected misalignment at the beginning of the trajectory; Bottom: fixed misalignment at the end of the trajectory; Center: successfully closed loop.

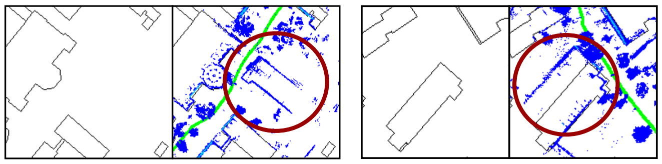

Fig. 6: Cases with map inconsistency. Left: the building is not in the map. Right: the building was extended.

B. Map Inconsistencies

The second experiment is designed to show that our approach is also robust against the map inconsistencies at least up to some degree. These inconsistencies may result from different sources, for example a building was wrongly mapped, a building was demolished, but is still present in the map, or a building that was built after the time of the map creation. All these cases may lead to inconsistencies in the resulting map built by the robot. As we take into account the matching-dependent uncertainty, the information about the inconsistencies is incorporated into the optimization process. Fig. 6 depicts two examples of the map inconsistencies that are successfully handled by our approach. The left image depicts a situation in which the building is visible in the scan and not present in the map and the image to right shows the case, where the building is wrongly mapped (building in the map is too small). Our system can deal with inconsistencies through the use of a robust kernel function. Fig. 7 shows the effect of disabling the robust kernel function. As it can be seen, the map gets distorted near the wrongly mapped buildings, see zoomed views in Fig. 6 and circles in Fig. 7.

C. Different Sensor Setups

Next, we illustrate that our approach works rather independently from a specific laser scanner. In our experiments, we are using three types of laser scanners: a 2D Hokuyo UTM-30, a 3D Velodyne Puck (VLP-16), and a Zoller+Fröhlich Profile scanner without any manual postprocessing of the data. Fig. 3 depicts the robot setups used in our experiments.

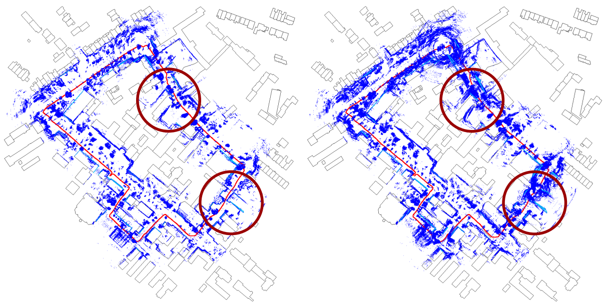

Fig. 7: Enabling / Disabling robust kernel function (DCS). Left: optimization using DCS. Right: optimization without robust kernel functions.

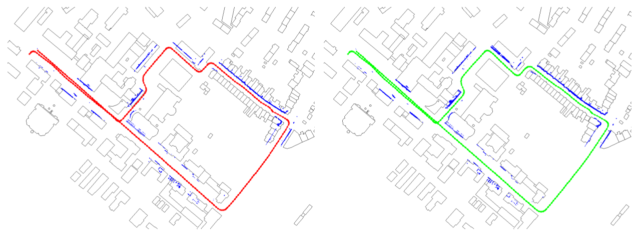

Fig. 8: The alignment of the robot's trajectory created using Zoller+Fröhlich scanner.

Figures 4, 5, 8 show the datasets obtained using three different sensors. As can be seen, all setups allow for an appropriate alignment of observed buildings with the open street map data.

D. Sensitivity to the Initial Guess

In our optimization procedure, the alignment starts with the initial guess about the first pose of the robot in the map. The initial pose is assumed to be roughly known. This can happen through GPS or can be provided manually. In this experiment, we show the sensitivity and stability of the system with respect to different initial robot pose. Depending on the structure of the environment and the result of the building detection algorithm in different datasets, we obtain

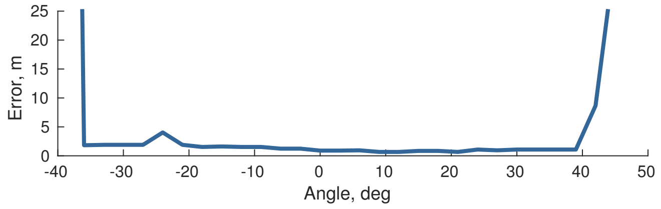

Fig. 9: Ability to correct the alignment given an initial deviation from the ground truth in the robot's heading. For deviations of more that $\pm 45°$, the alignment error increases substantially.

TABLE I: Timing results for processing the whole dataset (full) and processing a chunk (per chunk) of the dataset of $\approx 30\,m$.

		full		per chunk	
	dist	pose-graph	osm	pose-graph	osm
dataset 1	168 m	9.89 s	0.9 s	1.75 s	0.16 s
dataset 2	336.6 m	62 s	0.83 s	5.52 s	0.074 s
dataset 3	579.6 m	41.5 s	4.93 s	2.14 s	0.25 s
dataset 4	1040 m	86 s	4.1 s	2.48 s	0.11 s

that the maximum allowed error in orientation varies between $\pm 40°$ and $\pm 50°$, whereas the inaccuracy for the position may vary between $5\,m$ to 10. An example for the resulting alignment error depending on the deviation from the true heading is shown in Fig. 9. As can be seen, from deviation of more than $\pm 45°$, our approach does not find a proper alignment. In general, the approach is more sensitive to deviations in the orientation than to deviations in the x/y location.

E. Runtime

For mobile robots to operate in real world environments the computational load of the underlying algorithms should be small. In this experiment, we show that our approach adds only a small, if not negligible, computational overhead to the simultaneous localization and mapping process. We have run our algorithm on several different datasets with various size and complexity and summarize the runtime results in the Tab. I. As it can be seen, the time needed to process additional map knowledge (osm, 4th and 6th columns) is almost negligible in comparison to the time needed for the pose graph SLAM (pose-graph, 2d and 5th columns). This means that our extension can be included into the optimization procedure without adding any significant computational overhead.

V. Conclusion

In this paper, we proposed a novel extension to graph-based SLAM that allows for exploiting existing map information such as OpenStreetMap data. We extended the standard graph-based SLAM formulation by introducing an additional term in the error function. This term relates the nodes of pose-graph with building information from publicly available maps. Our methods has a minimal computation overhead compared to standard graph-based SLAM and has the possibility to improve the map quality. We implemented our approach within the g2o framework, evaluated it on real world data taken in urban environments, and illustrated the benefits of our method for the mapping process.

Acknowledgment

The authors would like to thank Michael Ruhnke from the University of Freiburg for valuable feedback and fruitful discussions as well as Lasse Klingbeil and Markus Wieland from the University of Bonn for the dataset recorded with the Zoller+Fröhlich profile scanner.

References

[1] P. Agarwal, W. Burgard, and L. Spinello. Metric localization using google street view. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2015.

[2] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[3] T. Bailey and H.F. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I. *Robotics & Automation Magazine*, 13(2):99–110, June 2006.

[4] O. Bengtsson and A.-J. Baerveldt. Robot localization based on scan-matching estimating the covariance matrix for the IDC algorithm. *Robotics and Autonomous Systems*, 44(1):29 – 40, 2003.

[5] P.J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[6] I. Bogoslavskyi, M. Mazuran, and C. Stachniss. Robust homing for autonomous robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

[7] A. Censi. An accurate closed-form estimate of icp's covariance. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[8] G. Floros, B. van der Zander, and B. Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1054–1059, 2013.

[9] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2010.

[10] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[11] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symp. on Comput. Intell. in Rob. and Aut. (CIRA)*, 1999.

[12] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[13] M. Hentschel, O. Wulf, and B. Wagner. A gps and laser-based localization for urban and non-urban outdoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 149–154, 2008.

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[15] R. Kümmerle, B. Steder, C. Dornhege, G. Kleiner, A.and Grisetti, and W. Burgard. Large scale graph-based slam using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.

[16] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4, 1997.

[17] A. L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza. Mav urban localization from google street view data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3979–3986, 2013.

[18] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard. Localization on openstreetmap data using a 3d laser scanner. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 5260–5265, 2015.

[19] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer, 2016.

[20] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[21] O. Wulf, K. O Arras, H. I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4204–4209, 2004.