Global Outer-Urban Navigation with OpenStreetMap

Benjamin Suger Wolfram Burgard

***Abstract*— Publicly available map services are widely used by humans for navigation and nowadays provide almost complete road network data. When utilizing such maps for autonomous navigation with mobile robots one is faced with the problem of inaccuracies of the map and the uncertainty about the position of the robot relative to the map. In this paper, we present a probabilistic approach to autonomous robot navigation using data from OpenStreetMap that associates tracks from OpenStreeetMap with the trails detected by the robot based on its 3D-LiDAR data. It combines semantic terrain information, derived from the 3D-LiDAR data, with a Markov-Chain Monte-Carlo technique to match the tracks from OpenStreetMap with the sensor data. This enables our robot to utilize OpenStreetMap for navigation planning and to still stay on the trails during the execution of these plans. We present the results of extensive experiments carried out in real world settings that demonstrate the robustness of our system regarding the alignment of the vehicle pose relative to the OpenStreetMap data.**

Fig. 1. Our approach addresses robot navigation in outer-urban environments and aims to match a path given from OpenStreetMap (red) to the part in the sensor data that may most likely corresponds to the street (black). It is particularly designed to cope with map and GPS errors.

I. Introduction

Robust navigation is an important pre-requisite for truly autonomous mobile robot systems. A large class of state-of-the-art solutions for large-scale navigation uses solutions to the simultaneous localization and mapping (SLAM) problem to create an accurate map of the environment. Some of them generate dense occupancy grid maps plus a topo-metrical graph from it. The topo-metrical high-level graph is then used for global path planning purposes and the fine grained map is loaded dynamically as map tiles for the local vicinity of the robot. This has the drawback that unknown areas need to be explored and mapped first and then, at operation time, an active localization module needs to run online. Moreover, planning is only possible in areas that have been explored before and a lot of data needs to be stored.

In this work, we will replace the need of creating a global map in advance with data from OpenStreetMap[1] (OSM) and model the vicinity of the robot online, which enables an autonomous robot to navigate in previously unseen environments. This requires two capabilities: first, we need to know where the robot is in the OSM and, second, we need t know how the path should be computed from OSM and be transformed to project it onto the real trail in case of OSM erros. Our idea is inspired by the way how humans use maps provided by cartographic services for global path planning and navigation purposes. The cartographic data provided by OpenStreetMap is based on "Volunteered Geographical Information" (VGI), see Haklay and Weber [5], and is publicly available under a "Open Database License (ODbL)." The problem in the context of robotics is that VGI maps have a limited accuracy and that they cannot be directly compared to maps acquired by SLAM. To use them for robot navigation, we need to take several sources of errors into account, *e.g.*, nodes may be placed in wrong positions by errors of the contributors, inaccurate GPS estimates (during map creation and at operation time) and the sparse approximation of trails by line segments. Interestingly, humans can easily resolve these inaccuracies. We hypothesize this originates from the ability to classify tracks and associate them with the map, instead of relying purely on the GPS estimate.

In this paper, we provide a novel approach to imitate this human ability and make maps like OpenStreetMap useful for global path planning in outer-urban environments, to autonomously navigate on small streets, dirt- and forest roads. To achieve this, we compare the shape of the trails from the map with semantic terrain classification and use a Markov-Chain Monte-Carlo technique to determine the most likely position of the trail in our local frame. With this correction it is possible to use data from OSM for global path planning on a real autonomous robot which sends short-term subgoals to a local planner that navigates the robot in its nearby vicinity.

II. Related Work

In recent years, publicly available maps like OSM have gained interest in robotic applications and several authors have utilized such maps for localization purposes by match-

All authors are with the University of Freiburg, Institute for Computer Science, 79110 Freiburg, Germany. This work has been partly supported by the European Commission under the grant numbers ERC-AG-PE7-267686-LifeNav and FP7-610603-EUROPA2

[1]https://openstreetmap.org

ing the road network against data observed by the robot. For example, Floros et al. [3] use the Chamfer-distance to compare chunks of visual-odometry trajectories from the robot with the network structure of OSM. Mandel and Birbach [8] fuse GPS and odomtery data for OSM localization. Ruchti et al. [11] segment single 3D-LiDAR scans into *street* and *no street* areas and match the street segments against the road-network, while also taking into account the negative information of *no street*. Whereas these approaches show an impressive robustness for global localization, they only provide limited accuracy with localization errors typically exceeding several meters, which is not sufficient to align goal points with the tracks as done by our approach.

Other approaches use maps of geotagged images as a source for localization and match information retrieved from images against the images in the map. There are also feature-based approaches that match panoramas, *e.g.*, from GoogleStreetView, against the local camera images [1, 14, 13]. However, these approaches assume a rich image-based map, which is not available everywhere, especially not in outer-urban and forested environments. Another, camera-based, approach by Radwan et al. [10] extracts textual information from images and matches it against georeferenced text of the map for localization, a source of information that is sparse in outer-urban environments.

More similar to our approach, Máttyus et al. [9] combine monocular aerial images with stereo images from a robot, estimating their pose in OSM and enhancing the map with semantic information that is derived from deep learning image classification. The authors assume the error of the OSM alignment to be perpendicular to the road direction, an assumption that we use in our approach as well. However, they furthermore assume that the error is within fixed bounds of four meters, which seems sufficient for the urban dataset from KITTI. In our case we cannot restrict the error to fixed bounds, since in forested environments we face errors of up to ten meters and more.

Another approach by Irie et al. [7] extracts boundaries from the structural representation of GoogleMaps and projects them to images recorded by the robot. The matching score is based on squared-loss mutual information between the image features and projected labels from the map. This approach is dedicated to urban environments and needs manual preprocessing for map conversion. An approach for OSM navigation was recently presented by Hentschel and Wagner [6] where 3D-LiDAR data is matched against the rich information of landmarks also provided by the map, *e.g.*, the compendium of buildings. The approach is particularly suited for urban environments and cannot directly be applied to outer-urban environments as they are not guaranteed to contain a sufficient amount of the required features.

From the related work, it is clear that the uncertainty in the quality of OSM is a well-known problem. In the context of urban environments approaches often assume error bounds ranging from two to four meters, which is typically a strong assumption for less frequently used outer-urban environments. Haklay [4] evaluated the quality of OSM with a focus on London and England. His analysis shows that positions recorded by Ordnance Survey (OS) at different areas in London are on average six meters away from the corresponding positions in OSM. He also compared the overlap of smaller roads in OS and OSM with results varying from 5% to 100%, which indicates substantial errors in such maps and also provides a clear motivation for our approach.

III. Preliminaries

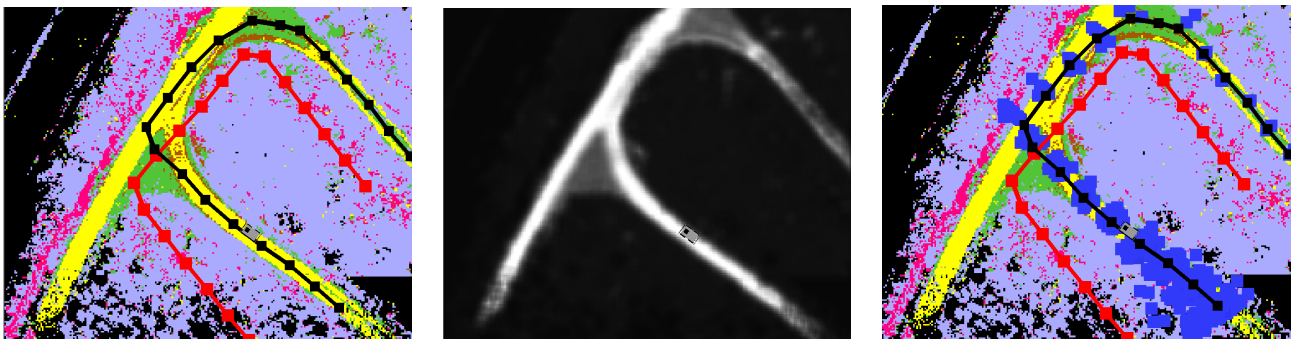
In this section, we briefly introduce the framework upon which we build the work presented in this paper. To achieve a homogeneous system we aim to connect two components, global planning on OpenStreetMap and the local frame of the robot, which is represented by semantic classification of terrain, derived from 3D-LiDAR sensor readings. We will use the road network from OpenStreetMap as a global metric topological map in order to provide short term subgoals to a planner that models only the local vicinity of the robot using sensor information provided by GPS, IMU, odometry and a 3D-LiDAR sensor.

A. Planning on OpenStreetMap

For efficient planning on the street-network graph, we use a standard A*-planner to calculate a path to the desired goal location on the OSM graph. Assuming that the vehicle is located on a road we deterimine the starting point by the orthogonal projection of the GPS pose to the closest street in OSM. To provide incremental subgoals to the local planner, we sub-sample the subgoals equidistantly along the path generated by the A* algorithm. Accordingly, we obtain a sequence of proposed subgoals $g_0^G, \ldots, g_n^G$ in the global UTM coordinate system. Whenever a subgoal is almost reached, we send the next one, until the final goal-point is sent. An example of this sub-sampled path is visualized in red on the left in Fig. 2 (a). However, this naive approach itself can not be used directly, due to diverse sources of errors, such as inaccuracies in the map or the GPS pose estimate. As Máttyus et al. [9] described, these errors mostly matter in the direction perpendicular to the road. To correct this error and thus make the navigation plan useful, our approach includes a dedicated classifier for the terrain in the vicinity of the robot to identify the road.

B. Semantic Terrain Information

To integrate the global subgoals consistently into the local frame of the robot, we will rely on a 3D-LiDAR-based semantic classification of the terrain type. For this, we use the same random forest classifier as in our previous work [12], in which we used a rolling grid map of fixed resolution to register the 3D-points based on the odometry measurements for a fixed distance d. Then we compute for each cell a five-dimensional feature vector, based on the remissions and the local geometry of the points registered to the cell. We train a random forest classifier as a fuzzy terrain type classifier, and integrate multiple measurements assuming a static terrain map, the independence of individual cells and



(a) (b) (c)

Fig. 2. Subfigure (a) visualizes the semantic classification, distinguishing *street* (yellow), *dirtroad* (brown), *grass* (green), *vegetation* (purple) and *other* (pink). The red line visualizes the OSM representation without correction and the black line the positions estimated by our approach. Subfigure (b) visualizes the weighting function, where bright colors correspond to high weights. In subfigure (c), blue markers represent the distribution of the samples for possible subgoals.

the independence of consecutive observations. Therefore, we can maintain a probability distribution over the terrain type given the features f for each individual cell

$$P(t \mid f_1, \ldots, f_n) = \eta \frac{P(t \mid f_n) P(t \mid f_1, \ldots, f_{n-1})}{P(t)}, \quad (1)$$

with a normalization constant η that is independent of the terrain class t and a prior $P(t)$, which is uniform in our current implementation. For more details we kindly refer the reader to Suger et al. [12]. We distinguish between the terrain classes *street, dirtroad, grass, vegetation* and *other*, see Fig. 2 (a) for a visualization of the terrain class map.

IV. Subgoal Alignment

In this section, we explain how our approach combines the information from the two independent maps. From the global planner on OSM we receive a series of subgoals $g_0^G, \ldots, g_n^G$, which are represented by their UTM-coordinates. We transform the received subgoals to our local odometry frame, leading to $g_0^L, \ldots, g_n^L$. With the subgoals in the robot frame, we calculate the difference $G_k^L = g_k^L - g_{k-1}^L$, which we represent by the orientation $\theta_k = \operatorname{atan2}(G_k^L(y), G_k^L(x))$ and the distance $d_k = \|G_k^L\|_2$, and model it as values of a random variable u_k. The goal of our work is to find the best configuration of the local subgoals, taking the observations from the 3D-LiDAR into account. To achieve this goal, we derive a probabilistic formulation in Sec. IV-A and provide details of its components in Sec. IV-B and Sec. IV-C. Finally to estimate the positions of the most recent subgoals we use this formulation in a Markov-Chain Monte-Carlo approach in Sec. V.

A. Probabilistic Formulation

Our goal is to find the most likely arrangement of the local subgoals $\hat{g}_0, \ldots, \hat{g}_n$, such that every subgoal is on the street given the relative positions $u_1, \ldots, u_n$ and the observations, integrated in the terrain class map $\mathcal{Z}$.

$$\hat{g}_{0:n} = \operatorname{argmax} \; p\left(g_{0:n} \mid \mathcal{Z}, u_{1:n}\right) \quad (2)$$

This results in a $2(n+1)$-dimensional optimization problem and to our knowledge no efficient method to directly solve it exists. Therefore, we aim for a recursive solution in order to reduce the dimensionality of the problem. By construction we can assume that the subgoals $g_{0:n}$, given $u_{1:n}$, satisfy the Markov property and therefore form a Markov-Chain.

$$p(g_n \mid g_{0:n-1}, u_{1:n}) = p(g_n \mid g_{n-1}, u_n) \quad (3)$$

Moreover, we made the assumption that the cells of the terrain map are independent and therefore, given the subgoals g_i are well separated, the observations $z_{0:n}$ made at the subgoals are independent given the subgoals.

$$p(z_{0:n} \mid g_0, \ldots, g_n) = \prod_{i \leq n} p\left(z_i \mid g_i\right) \quad (4)$$

Now we can iteratively apply Bayes rule and combine it with Eq. (3) and Eq. (4) to derive the following factorization of the posterior.

$$\hat{g}_{0:n} = \underset{g_{0:n}}{\operatorname{argmax}} \; p\left(g_0\right) \prod_i p\left(z_i \mid g_i\right) p\left(g_i \mid g_{i-1}, u_i\right) \quad (5)$$

Here, $p\left(g_0\right)$ is the prior distribution, $p\left(z_t \mid g_t\right)$ is the observation likelihood and $p\left(g_t \mid g_{t-1}, u_t\right)$ is the process model. Even though each factor can be computed efficiently now, the search space for a full trajectory remains too large. Therefore, we use an iterative solution to overcome this.

B. Process Model

We assume that the main source of error occurs perpendicular to the street direction. We reflect this in the modeling of the process model, which has the form

$$g_k = h\left(g_{k-1}, u_k, \Delta\right), \quad (6)$$

with

$$h\left(g_{k-1}, u_k, \Delta\right) = g_{k-1} + \frac{d_k + \Delta d}{\cos(\Delta\theta)} \begin{pmatrix} \cos(\theta_k + \Delta\theta) \\ \sin(\theta_k + \Delta\theta) \end{pmatrix}, \quad (7)$$

where Δ is a zero mean Gaussian noise variable.

$$(\Delta d, \; \Delta\theta) \sim \mathcal{N}\left((0, \; 0), \begin{pmatrix} \sigma_d & 0 \\ 0 & \sigma_\theta \end{pmatrix} \right) \quad (8)$$

The additional factor $(cos(\Delta\theta))^{-1}$ accounts for the angular deviation and adjusts the distance, unfolding the commonly used banana-shaped distributions perpendicular to the street direction. For the initial distribution we assume that the direction of the street is known and therefore model

$$g_0 \sim \mathcal{N}\left(g_0^L, \Sigma_0\right) \tag{9}$$

with an appropriately rotated covariance matrix. The visualization of the sample distribution is shown in Fig. 2(c).

C. Measurement Model

To calculate the observation likelihood we use the terrain class map, that we derive from the measurements as described in Sec. III-B. For a quantification of the semantic information given a local subgoal g, $P(\mathcal{Z} \mid g)$, we map the terrain classes to costs, which increase as it becomes more unlikely to see a class on a road. More precisely, we define a function $f : \mathcal{T} \rightarrow \mathbb{R}^+_{>0}$ from the set of classes $\mathcal{T} = \{street, dirtroad, grass, vegetation, other\}$ to a discrete subset of the real half-line. Given the mapping f we can calculate a value for each cell $C \in \mathcal{Z}$ using the results of the fuzzy classification, by calculating the expected value of the function f over all terrain classes of cell C as

$$f^E(C) = \sum_{t \in \mathcal{T}} f(t) P(t \mid C), \tag{10}$$

where $P(t \mid C)$ is maintained by the terrain map, see Sec. III-B. Now, as we aim to find evidence for a part of the map belonging to a street, and since our map has a fixed resolution r, we do not only use a single cell but a square grid neighborhood with the center cell at the subgoal g, which we refer to as $\mathbf{N}(g)$, of size $(2N+1) \times (2N+1)$. With respect to this we compute the final costs as an equally weighted average of f^E over the cells in $\mathbf{N}(g)$.

$$f^{\mathbf{N}}(g) = \frac{1}{|\mathbf{N}(g)|} \sum_{C \in \mathbf{N}(g)} f^E(C) \tag{11}$$

To transform this cost value into a likelihood we use an exponential distribution with parameter λ in our current implementation, stating the observation likelihood as

$$P(z \mid g) \sim \lambda \exp\left(-\lambda f^{\mathbf{N}}(g)\right). \tag{12}$$

Recall that we used the term of well separated subgoals in the derivation of Eq. (4), now it is clear that this can be defined as $\mathbf{N}(g_i) \cap \mathbf{N}(g_j) = \emptyset$, $\forall i \neq j$. This also shows the tight connection of the assumption that observations are independent given the subgoals and the independence of the cells in the terrain map $\mathcal{Z}$. Fig. 2(b) visualizes the observation likelihood for every cell of the map.

V. Sequential Markov-Chain Monte-Carlo Sampling

To solve the optimization problem stated in Eq. (5) we employ a MCMC method for an incremental solution. In the real world, one of the main problems in our scenario is visibility. Especially in forested environments, an intersection is often not observed before the robot actually arrives there.

Mooswald Schauinsland

Fig. 3. GPS tracks of the datasets we used for evaluation on an aerial image. Autonomous trajectories are colored in yellow, remotely controlled trajectories are colored in red. Mooswald datasets are collected nearby our campus and Schauinsland datasets on a mountain near Freiburg.

Moreover, the process model, given in Eq. (7), models the incremental, moderate error and cannot resolve a larger error, which typically occurs at intersections, in a single step. On the other hand, we are interested in subgoals that are as far away as possible, which is in favor of the local planner. Therefore, we do not sample the most recent subgoal only but a sequence of the most recent K subgoals.

$$\begin{aligned} &p\left(g_{n-K+1:n} \mid \mathcal{Z}, u_{1:n}\right) = \\ &\int \Bigg(\prod_{i=n-K+1}^{n} p\left(z_i \mid g_i\right) p\left(g_i \mid g_{i-1}, u_i\right) \cdot \\ &p(g_{n-K+1} \mid g_{n-K}, u_{n-K}) p\left(g_{n-K} \mid \mathcal{Z}, u_{1:n-K}\right) \Bigg) \mathrm{d}g_{n-K} \end{aligned} \tag{13}$$

Unfortunately, this increases the complexity of the distribution and especially in online scenarios computation time is crucial. Therefore, we need to find a proper trade-off between accuracy and computational time, which we analyze in our experimental section. To complete the description of our approach we briefly state the needed steps, which are *sampling*, *weighting* and *resampling*. In our case, each sample G_j contains a sample of the most recent K subgoals, sequentially sampled from the proposal distribution

$$G_j \sim \prod_{i=n-K+1}^{n} p\left(g_{i,j} \mid g_{i-1,j}, u_i\right). \tag{14}$$

In practice we sample a series of Δ_i from the distribution given in Eq. (8) and consecutively apply Eq. (6) to sample hypothesis G_j. Furthermore, we calculate the importance weight for G_j as

$$\mathbf{w}_j \propto \prod_{i=n-K+1}^{n} p\left(z_{i,j} \mid g_{i,j}\right). \tag{15}$$

Both computations can be done efficiently due to the recursive structure. However, the high dimensionality of the state space requires substantially more samples. After the weighting we calculate the so-called number of effective particles $n_{eff} = \left(\sum \mathbf{w}_j^2\right)^{-1}$. If n_{eff} is less than half the number of samples we perform a resampling procedure, in which we use stochastic universal sampling as introduced

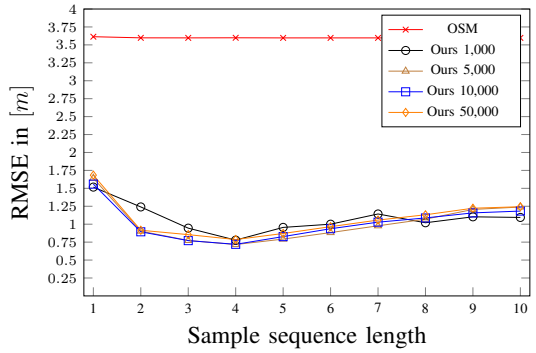


Fig. 4. Average RMSE over ten instances for our approach (N) and the uncorrected data from OSM, where N denotes the number of samples. Our approach achieves similar RMSE for different sample sizes and the minimum for the sequence length of four.

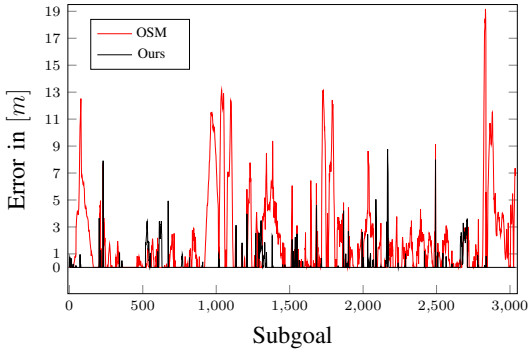


Fig. 5. Error of our approach compared to uncorrected data from OSM for every subgoal used in our evaluation set. For our approach we used 1,000 samples and a sequence length of four in this case.

by Baker [2]. For the current state estimate we use the expectation of the sampled trajectory.

VI. Experiments

We evaluated our approach using real world experiments with the robot shown in Fig. 1, which was navigated both autonomously and via remote control. First, we give a detailed overview of our system settings in Sec. VI-A. To measure the quality of our approach, we need to evaluate whether or not the subgoals are located on the trails in the local frame of the robot, such that the robot can follow the desired path, which was globally planned on OSM. In Sec. VI-B, we will investigate the influence of the sample sequence length K and the number of samples use to approximate the distribution to this measure. Especially in online settings, runtime becomes important and results concerning this are discussed in Sec. VI-C. Finally, we reason about limitations of the approach and discuss cases in which our approach may fail to find the correct path in Sec. VI-D.

A. System Setting

For all experiments we used data from the same robot, equipped with a Velodyne HDL-64E 3D-LiDAR sensor, Gyro sensors, IMU sensors, Kalman-filtered GPS and internal odometry. The terrain classification accumulates measurements for a driven distance of $d = 0.5m$ between consecutive feature computations. The resolution of the grid was set to $0.2m$ with a size of 300×300 to model the vicinity of the robot, see Sec. III-B. Subgoals were generated every four meters and the map service we used was OpenStreetMap. For all experiments we set the neighborhood size $N = 2$, which corresponds to a size of $1m \times 1m$, and $\lambda = 4$, see Sec. IV-C. The terrain cost function was set to $\{$*street*: 0.1, *dirt*: 0.2, *grass*: 0.4, *vegetation*: 0.8, *other*: 1.0$\}$. In the case when a cell without observation was requested, a default value equal to that of *other* was returned. For the evaluation we considered datasets from two locations, one nearby our campus (Mooswald) and one from a mountain near Freiburg, the Schauinsland. The GPS-tracks of the trajectories are visualized on aerial images in Fig. 3. All in all, we evaluated data with an overall trajectory length of 14 km.

B. Performance

First to mention, we successfully applied our approach to a real robot, navigating autonomously in both areas, see Fig. 3, with an overall trajectory length of 5.2 km, in which we used 1,000 samples and a sample sequence length of four. Our approach made the difference between the robot leaving the road and getting stuck and successfully reaching its goal. This runs contained challenging situations including Y-shaped intersections as well as nodes at intersections that were several meters off. To quantify this qualitative results we calculate the root mean squared error (RMSE) measuring the distance of the subgoals to the middle of the street in the local frame. Due to a lack of ground truth data, we approximate this error measure utilizing the knowledge that the robot was driving near the middle of the street most of the time. Given a sequence of robot poses $p_{1:m}$ and a sequence of subgoals $g^K_{1:n}$, corrected with a sample sequence length K, and assuming a medium street-width $s = 3m$, we calculate the RMSE as:

$$RMSE_K = \sqrt{\frac{\sum_{i \leq n} \max\left(\left|\min_{l \leq m} \|g_i^K - p_l\|_2 - \frac{s}{2}\right|, \ 0\right)^2}{n}} \quad (16)$$

For the computation of the RMSE we concatenate the poses and subgoals of all trajectories. We examine samples sizes between 1,000 and 50,000 and vary the sequence length from one to ten. Due to the randomized nature of our approach we compare the average RMSE over ten instances for each setting. The results along with the RMSE of the subgoals from uncorrected OSM data is depicted in Fig. 4. The sequence length of four achieves the minimum RMSE for all the sample sizes, ranging from $0.78m$ for 1,000 samples to $0.72m$ for 10,000 samples. Compared to the raw subgoals from OSM, which exhibits a RMSE of $3.6m$, our method yields an error reduction of up to 80%.

In Fig. 5, we compare the errors at each subgoal, corrected with our approach using 1,000 samples and a sequence length of four to the errors of the raw subgoals from OSM. The

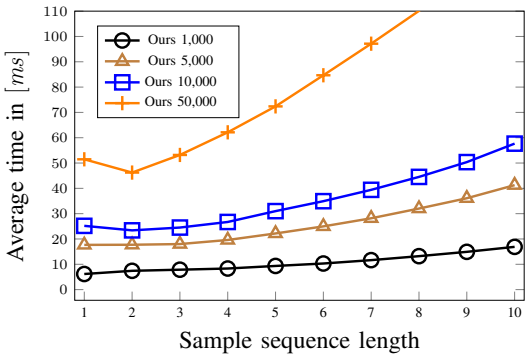


Fig. 6. Average runtime for calculating an update for our approach N, where N denotes the number of samples.

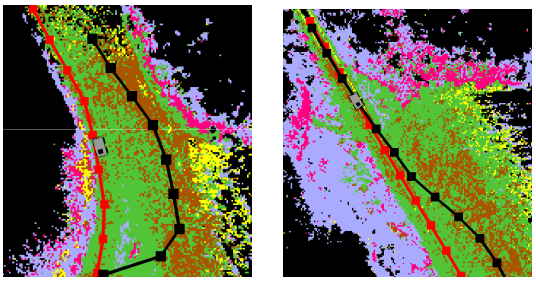


Fig. 7. An example where our approach fails to find the correct path (left), due to an ambiguous classification outcome. The path from OSM (red) matches precisely. Our method (black) estimates the subgoals at an area that is classified as dirt road, whereas the actual path is classified mostly as grass. As soon as the classification improves our method recovers (right).

errors of OSM exceed ten meters several times, while the errors of our approach are typically below $1m$. Except for one situation, which we explain in Sec. VI-D, the single peaks that exhibits a higher error for our approach stems from intersections, at which the area of possible street points is large and the trajectory often takes a shortcut relative to the shape representation in OSM, which is not considered by our error measurement. Nevertheless, comparing the errors from our approach to OSM in a single tailed paired student's-t-test, the improvement using our approach was significant ($\alpha = 0.05$) for all settings and instances.

C. Runtime

Execution time of the approach is important, especially when deploying it on a real autonomous system. Our approach is efficiently implemented in C++ and the runtime was measured for the different settings on a single core of an Intel(R) Core(TM) i7-2700K CPU @ 3.50 GHz. The terrain map updates, which are computed every 0.5 m in a separate thread, took on average 0.3 s and are not further included in this evaluation. For our approach, the runtime is almost linear in the number of samples and the sequence length, see Fig. 6. With 1,000 samples and a sequence length of four, the average update time is 8.3 ms, whereas this increases to 62 ms with 50,000 samples. Our choice of parameters is fully supported by our runtime and accuracy evaluation. This setting of the system yields realtime performance which is integral for autonomous navigation in real world scenarios.

D. Limitations

The performance of our approach depends on the quality and correctness of the terrain classification. In the remotely controlled experiments we encountered a situation, in which the path was covered by grass and several meters away the classifier found evidence for a dirt road, accordingly the estimation of the subgoals drifts towards that region, see the left image in Fig. 7. Nevertheless, as soon as the path is distinguishable from the surroundings some meters later, our approach can successfully recover from that failure, see the right image in Fig. 7. When navigating autonomously this means that the robot may deviate from the desired path in that region and return to it as soon as our method recovers from its failure.

VII. Conclusions

In this paper, we presented an approach to use publicly available map data from OpenStreetMap as global map for autonomous navigation in outer-urban environments. Our approach relies only on perceptions of the local vicinity of the robot, as the global map is represented by OSM, and therefore requires constant memory only. It furthermore is highly efficient and can operate online on a robot navigating autonomously in real world settings. The experimental evaluation shows the robustness of our approach.

References

[1] P. Agarwal, W. Burgard, and L. Spinello. Metric localization using google street view. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2015.

[2] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proc. of the Second Int. Conf. on Genetic Algorithms*, 1987.

[3] G. Floros, B. van der Zander, and B. Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2013.

[4] M. Haklay. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and planning B: Planning and design*, 37(4), 2010.

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

[6] M. Hentschel and B. Wagner. Autonomous robot navigation based on openstreetmap geodata. In *Intelligent Transportation Systems (ITSC)*, 2010.

[7] K. Irie, M. Sugiyama, and M. Tomono. A dependence maximization approach towards street map-based localization. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2015.

[8] C. Mandel and O. Birbach. Localization in urban environments by matching sensor data to map information. In *European Conference on Mobile Robots (ECMR)*, 2013.

[9] G. Máttyus, S. Wang, S. Fidler, and R. Urtasun. Hd maps: Fine-grained road segmentation by parsing ground and aerial images. In *IEEE Conf. on Comp. Vis. and Patt. Rec. (CVPR)*, 2016.

[10] N. Radwan, G. D. Tipaldi, L. Spinello, and W. Burgard. Do you see the bakery? leveraging geo-referenced texts for global localization in public maps. In *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, Stockholm, Sweden, 2016.

[11] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard. Localization on openstreetmap data using a 3d laser scanner. In *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2015.

[12] B. Suger, B. Steder, and W. Burgard. Terrain-adaptive obstacle detection. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2016.

[13] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *IEEE Conf. on Comp. Vis. and Patt. Rec. (CVPR)*, 2015.

[14] A. R. Zamir and M. Shah. Accurate image localization based on google maps street view. In *European Conference on Computer Vision*. Springer, 2010.