Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors

Fan Yan Olga Vysotska Cyrill Stachniss

***Abstract*— Localization is an essential capability of mobile vehicles such as robots or autonomous cars. Localization systems that do not rely on GNSS typically require a map of the environment to compare the local sensor readings to the map. In most cases, building such a model requires an explicit mapping phase for recording sensor data in the environment. In this paper, we investigate the problem of localizing a mobile vehicle equipped with a 3D LiDAR scanner, driving on urban roads *without* mapping the environment beforehand. We propose an approach that builds upon publicly available map information from OpenStreetMap and turns them into a compact map representation that can be used for Monte Carlo localization. This map requires to store only a tiny 4-bit descriptor per location and is still able to globally localize and track a vehicle. We implemented our approach and thoroughly tested it on real-world data using the KITTI datasets. The experiments presented in this paper suggest that we can estimate the vehicle pose effectively only using OpenStreetMap data.**

I. Introduction

Long-term operation is an important requirement for mobile robots to be considered as truly autonomous. One of the prerequisites for reliable navigation is the ability to correctly localize the robot. Typical robot operation routines consist of building a map of the environment with the robot's sensors and then localizing with respect to that map. Collecting this map data, however, can be tedious work, especially for outdoor robot localization, since the mapping procedure needs to cover a vast area to support continuous localization. Moreover, it is also a time- and resource-consuming operation.

Recently, different researchers proposed approaches that use the publicly available maps, like Google Maps or OpenStreetMap (OSM), to perform the robot localization [1], [5], [19]. The main advantage of these maps is that they already exist for a large portion of the world and are already available, partially free of charge. The disadvantage is that they were made to be human-readable and are not directly suitable in the robotics context.

Several approaches propose ways to incorporate the information from publicly available maps into a robot's simultaneous localization and mapping pipelines. For example, Floros *et al.* [5] extract the road network as a graph of line segments and use it as an additional cue in the observation model of the Monte Carlo localization framework. Afterwards, the robot pose estimates are tracked by a Kalman Filter for consistency of the visual odometry estimates. Vysotska *et al.* [18] propose to use buildings information from OpenStreetMap to correct the drift accumulated within 2D graph-based SLAM in outdoor environments.

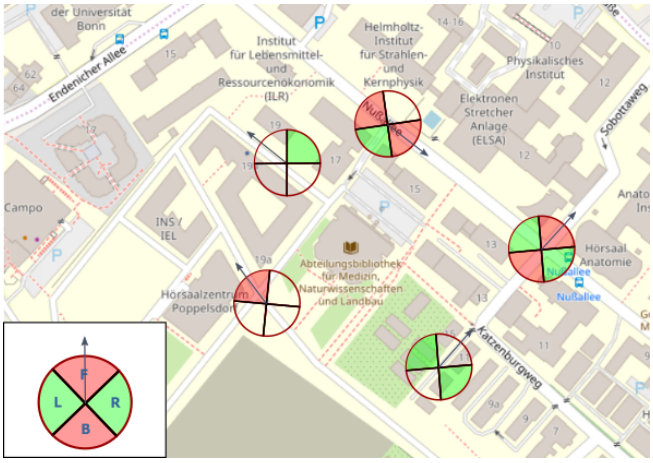


Fig. 1: Compact feature descriptor used for localization on OSM data without requiring an explicit mapping phase using the robot's sensors. Given robot pose (circle center) and sensor visibility range (circle radius), the extracted descriptor shows if the intersection is detected in front (F) and back (B) area, yes (red) / no (white), and if the building gap was detected in right (R) or left (L) section, yes (green) / no (white).

In this paper, we propose a novel global localization algorithm that uses building and road information extracted from OSM data to estimate the robot's pose. In our map representation, the estimated robot's pose can be directly converted to the GPS coordinates given the georeferenced OSM data and thus result in global positioning. The main novelty of our paper lies in deploying a 4-bit semantic descriptor originally proposed by Panphattarasap *et al.* [13] within the particle filter framework and not requiring a mapping phase, instead, using OSM data. This compact 4-bit descriptor indicates if a street intersection can be detected in the front or behind the robot as well as if there are any gaps in between the buildings to the right or to the left from it, see Fig. 1 for an illustration. We furthermore propose an algorithm to detect street intersections and building gaps from LiDAR range images. Different from the work of Panphattarasap *et al.* [13], who train a specific neural network that detects intersections and gaps from image data, we rely on semantic labels obtained from existing networks trained for semantic segmentation [12] as well as cost-efficient geometric checks.

In sum, we make the following key claims: Our approach

All authors are with the University of Bonn, Germany.

This work has partly been supported by the German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324 (PhenoRob).

978-1-7281-3605-9/19/$31.00 ©2019 IEEE

is able to (i) perform global localization using publicly available data, (ii) while being computationally efficient, since the building gap and intersection detection need to be done once per scan and all further comparison involves only 4-bit descriptors, (iii) and requires a small amount of memory to store the environment model. These three claims are backed up by the paper and our experimental evaluation.

II. Related Work

Localizing a robot in the environment is one of the fundamental problems for autonomous navigation and has been extensively studied in the past [4], [3], [9], [8]. Most of these approaches, however, require a precomputed map built by a robot. Only recently, we see the emergence of approaches that use publicly available information to facilitate the robot localization. For example, Floros *et al.* [5] propose to use the information from street graphs in OpenStreetMap in the form of edge maps to perform robust localization. Their main idea is to accumulate the robot's measurements from a continuous acquisition until a robot's trajectory forms a particular 2D pattern and align this pattern to the edge map using Champfer matching. In our approach, we form a descriptor from a single measurement and thus are able to perform global localization from the first measurement. Suger *et al.* [17] deploy the road and trail information from the OpenStreetMap to localize a robot in outer-urban environments. They use semantic terrain information to align the robot's pose to the tracks extracted from OpenStreetMap using the particle filter framework.

The work that gave inspiration to our approach is the work of Panphattarasap *et al.* [13], in which the authors propose a simplistic binary semantic descriptor that can be efficiently used in the urban areas with OpenStreetMap for global localization. The authors extract the descriptors from the visual input by training the neural network to recognize intersection and gaps between the buildings whereas in our approach we deal with 3D LiDAR data from KITTI datasets [6]. In our approach, we deploy an existing, publicly available network for semantic point cloud segmentation by Milioto *et al.* [12] to eliminate the need to train a specific network and perform simplistic geometric checks to determine road intersections and building gaps. Further approaches for semantic segmentation are also evaluated by Behley *et al.* [2]. Another approach that also uses 3D laser scanners is proposed by Ruchti *et al.* [14]. Here, the authors detect the road surface in the 3D scans and align their observations to the road network of OpenStreetMap using the particle filter approach. Further applications of OpenStreetMap data include correcting the drift of the 2D graph-based slam approaches as in [18] as well as using further features as poles from OpenStreetMap to perform robust localization in urban environments [15]. Another advantage of using the 4-bit semantic descriptor is its small size, which results in efficient memory usage, the problem also addressed by Milford *et al.* [11]. Another group of researcher use imagery from publicly available sources to perform robust localization [1], [10].

III. Our Approach

The main focus of our approach lies in performing global localization on OpenStreetMap using the particle filter framework in combination with a compact 4-bit semantic descriptor. In the remainder of this section, we first briefly discuss general Monte Carlo localization before explaining the detail of our approach. We describe how to obtain 4-bit semantic descriptors from the OSM as well as from the LiDAR range images and how this information can be incorporated into an MCL sensor model.

A. Monte Carlo Localization

To estimate the pose x_t of the vehicle at time t, we rely on Monte Carlo localization or MCL, originally proposed by Dellaert *et al.* [4]. The key idea of MCL is to represent the belief about the robot's pose by a set of weighted particles, which enables arbitrary distributions rather than restricting the belief to be a Gaussian distribution, as assumption a Kalman filter does. MCL recursively estimates the posterior about the robot's pose:

$$bel(x_t) = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}, \quad (1)$$

with η being the normalizer from Bayes' rule. The term $u_{1:t}$ denotes the sequence of motion commands and $z_{1:t}$ is the sequence of observations. The term $p(x_t \mid x_{t-1}, u_t)$ is the motion model describing the uncertainty of the motion execution. The observation model $p(z_t \mid x_t)$ refers to the likelihood of obtaining the observation z_t given the robot's current pose x_t. This work differs in comparison to most MCL realizations as we reduce the observations to 4-bit descriptor vectors and are still able to localize in an OpenStreetMap network.

For resampling, we use low variance resampling (also called stochastic universal resampling) in combination with adaptive resampling using the effective number of particles, see [7], [16], to reduce the risk of particle depletion. To do that, we compute

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2}, \quad (2)$$

where N is the number of particles and $w^{(i)}$ the weight of sample i. N_{eff} estimates how well the current particle set represents the true posterior and, thus, we resample only if N_{eff} drops below a given threshold (here $3N/4$).

As an initial distribution, we sample particles uniformly on the streets with uniformly distributed orientation. If further prior knowledge about the robot's pose is available, this can be considered in the initial belief.

B. Computing an Expected Observation Given the Map

One of the key steps to realizing the particle filter is to define the suitable expected observation. We estimate the expected observation from an OpenStreetMap representation given a pose and orientation of the robot. A particular part of the world can be exported from the OpenStreetMap web interface as a simple XML file where every road is stored as

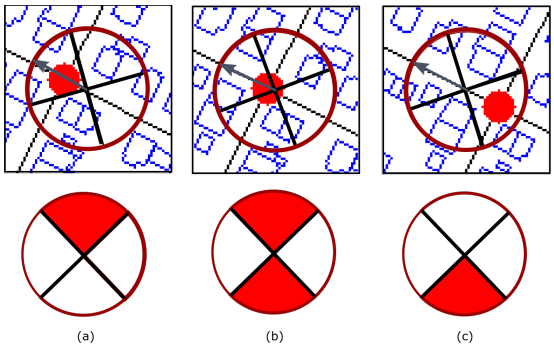


Fig. 2: Upper row: robot pose overlaid onto the map, arrow indicates the heading. Bottom row: corresponding descriptor: Robot (a) approaches the intersection, (b) at the intersection, (c) leaving the intersection. The red sector of the descriptor corresponds to the detected intersection.

a *way* element and a building is stored as a closed polygon. By parsing the XML file, we render roads and buildings onto an orthophoto-style 2D image. In this way, every pixel in the map directly corresponds to a global world coordinate, e.g., a GPS coordinate, and thus localizing a robot on such an image results in obtaining the geo-referenced position of the robot.

Our semantic binary descriptor encodes information about the visibility of road intersections and building gaps. Thus, we need to construct a corresponding expected observation given a possible position and orientation of the robot (a particle) and the sensor range. As a preprocessing step, we estimate the road intersections by checking for every GPS street coordinate if in its vicinity there exists a GPS coordinate from another street. The resulting intersections are marked as red circles in Fig. 2.

1) Road Intersections: Given a robot location and the visibility range of the sensor, we check for every pixel in front (F) and back (B) sector if it belongs to the intersection in the map. Whenever the number of pixels in the corresponding sector reaches a threshold θ_{iters}, we consider the intersection to be seen and set the corresponding bit of the descriptor to 1, illustrated with red color in the sectors F and/or B in Fig. 2 (bottom), cases (a) and (c). When the robot is located within the intersection both, front and back sectors, are getting activated, thus both bits get the value 1 as in case (b) in Fig. 2. In our experiments, we set the radius of the visibility circle on the map to be 25 m.

2) Building Gaps: A building in the real world is represented by a polygon in the OpenStreetMap map as well as in our map image, see Fig. 3 blue polygons. When parsing the XML file, we assign an individual index to each separate building stored in OpenStreetMap. This allows us to efficiently detect building gaps within the side sector of the descriptor, since we only need to check whether there exist two different building IDs in the corresponding sector of the visibility circle, as is the case for Fig. 3 (a) and (b). Sometimes due to the properties of the cities, several buildings can be connected together, but are represented as individual elements in OpenStreetMap as in Fig. 3 (c).

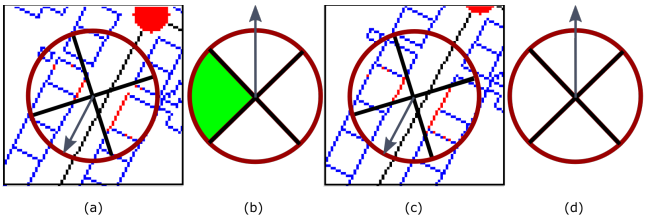


Fig. 3: Building gap information generated from the map: (a) R sector gap is occluded (a) L sector gap is visible. The images (c) and (d) illustrated the absence of observable building gaps.

Fig. 4: Semantic range image. Roads are visualized in purple and buildings in yellow.

This leads to the fact that the descriptor for the expected observation would show a building gap, although there is no visible gap observable in the real world. To robustly deal with such a situation, we assign the same ID to both buildings, whenever they share common polygon points given the OpenStreetMap data. As a result of this, the corresponding side sectors stay inactive, as depicted in Fig. 3 (d).

We furthermore distinguish if a building within the visibility circle is occluded by another building by performing ray casting between the robot's location and the building pixel. Whenever the ray crosses another building a corresponding part is considered unobservable and the corresponding bit stays inactive, as in Fig. 3 (a) right sector.

C. Descriptor from a Labeled Range Image

In addition to computing the descriptor for the expected observation, we also need to define how to obtain the descriptor from the sensor readings. The combination of expected and real observation is then used to define the observation model for MCL.

In this work, we rely on a 3D LiDAR as the robot's sensor that provides a 360-degree scan of the surroundings. Each scan is reduced to a 4-bit description, which is then in turn used for localization. Since our descriptor requires a basic notation of semantics (buildings and street intersections), we first deploy the approach of Milioto *et al.* [12] for semantic point cloud segmentation to obtain a semantic mask for a range image, an example is given in Fig. 4. For our purposes, we are only interested in roads, marked in purple, and buildings, marked in yellow. We use this approach as a black box, for further detail, see the work by Milioto *et al.* [12], and process the labeled range image to extract building gaps and street intersections as outlined below.

1) Road Intersections: For detecting intersections, we start with analyzing a labeled range image that contains road pixels only, as depicted in Fig. 5. Analogously, to the sectors of the descriptor, we divide a range image into four parts,

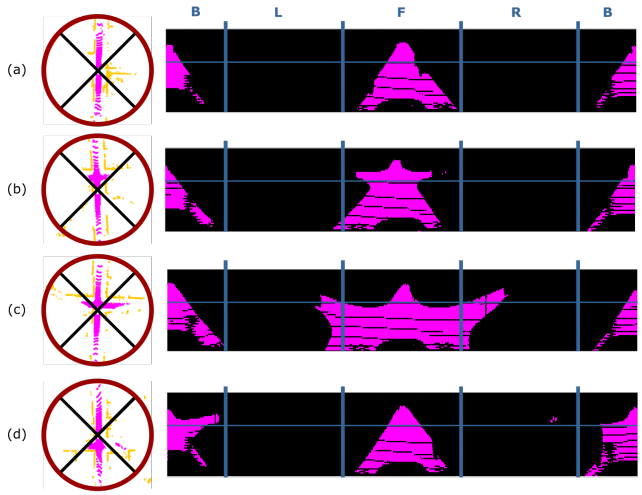


Fig. 5: Four possible patterns of intersection bits: (a) Robot is on a straight road; (b) approaching an intersection; (c) at the intersection; (d) leaving the intersection. Point clouds in the left column show the ground truth. Road range images in the right column are divided into F, R, B and L sections, where F and B sections are further divided into three sub-parts in order to detect the possible branch.

i.e. sets of range image columns, representing the front view (F), right side (R), back part (B) and left part (L) of the robot respectively. Then, we consider four different robot's location types with respect to a road network.

First, the robot is located on the street with no intersections, see Fig. 5 (a). The corresponding road pattern in the range image only shows road pixels in the top area (first 25 of 64 rows of the range image) of the front and back view. Since no intersection is detected in F or B area, these sectors of the descriptor stay inactive.

Second, the intersection appears ahead of the robot, see Fig. 5 (b). Then multiple roads form a different pattern scattered further away from the front part (F) in comparison to the first case. To detect multiple roads in the front part, we further subdivide it into three sub-parts: front-left, front-middle, and front-right. Whenever the number of road pixels in the top area of at least two subparts becomes larger than a threshold θ_{road}, we consider an intersection to exist and set the front bit of a descriptor to 1. In our experiments, the threshold θ_{road} was set to 100 pixels.

Third, the robot is at the intersection as in Fig. 5 (c). In this case, we check whether there exist road pixels in the top of either right (R) or left (L) part. Whenever the θ_{road} is reached and there also exist road pixels in the bottom area, we set both front and back descriptor bits to 1.

Forth, in case the robot is leaving an intersection as in Fig. 5 (d), multiple roads can be seen in the back part of the range image. Analogously to the second situation, we check if there is more than one road in the back and in this case set the back descriptor bit to one. To perform these checks, we only need to iterate once over the range image, which results in an efficient street intersection detection approach.

A potential weak point of this counting scheme, however, can be wrong intersection detections whenever we encounter

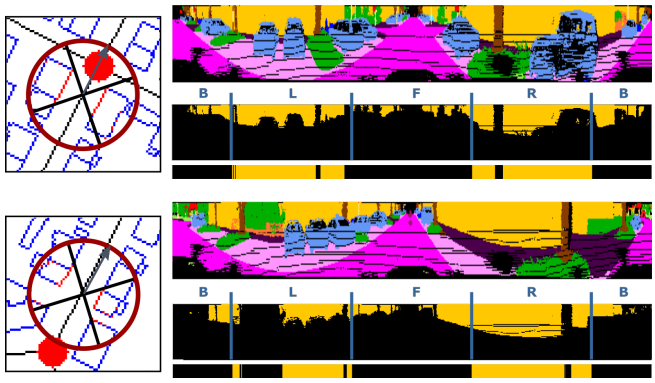


Fig. 6: Two examples of building gap detections. From top to bottom: Semantic range image, corresponding buildings mask, a row measurement composed by storing the minimum depth value along the respective column.

a strongly curved street within one scan. If a street has a high curvature, a number of pixels may land not only in the front-middle part but also the front-left or front-right part, causing an intersection detection in the front to be triggered, although in reality there is none. This situation, however, happens rarely and is successfully covered by the considered noise model in the observation model of the particle filter.

2) Building Gaps: To detect the gaps between the buildings from a 3D laser scan, we analyze the semantic range image in a similar manner as for the intersection detection, but in this case, we only look at the pixels that are labeled as buildings, see yellow pixels in Fig. 6. As before, we consider the range image to be divided into four parts (F, B, L, R). Since the descriptor only considers gaps to the sides of the robot, we further analyze only the left (L) and right (R) parts. As can be seen from the figure, parts of buildings are often hidden behind the obstacles, like tree trunks, vegetation, or cars. To robustly deal with these types of occlusions, we project 2D building range image to 1D "row-like" representation, by iterating over every column of the L and R area of the range image. Whenever pixels of a column are labeled as building, the corresponding row entry is labeled as building, all other entries correspond to the potential gaps. Simultaneously, we associate with every row pixel the minimum depth of the building seen in the corresponding column. An example of a 1D representation extracted from a labeled range image is depicted in Fig. 6.

To detect the building gaps, we iterate over each pixel of the row representation and count the number of consecutive buildings and gaps pixels. This results in obtaining potential hypothesis for the gaps. If the gap size is less than a threshold θ_{gap}, the gap is considered to be a sensor noise and is neglected, as in Fig. 6 (top) left sector. In general, we consider a potential gap to be a building gap if there exists a consistent gap, i.e., gap larger than θ_{gap}, between two building components in the row measurement, as is the case in Fig. 6 (bottom) left sector. The presence of the obstacle in front of a building also generates a potential gap in row measurement. To deal with this situation, we can further inspect the semantic label of a gap. If it has a consistent label

of an obstacle, e.g., tree trunk, we further neglect this gap, as depicted in Fig. 6 (top) right sector. We simultaneously track the depth consistency of the pixels in a building component. The depth is consistent whenever the depth change between the adjacent pixels is less than a threshold θ_{depth}. Whenever a building stands behind another building from a scanner perspective, the gap between the buildings cannot be detected solely based on semantic information. However, this situation shows a rapid depth change and violates the depth consistency assumption. In this case, we also consider the gap to exist, see Fig. 6 (bottom) right sector.

D. Observation Model for MCL

In previous sections, we described how to compute the 4-bit descriptors from OpenStreetMap as well as from an individual LiDAR scan. In this section, we describe how to obtain particle weights given a pair of descriptors, e.g., real and expected. The observation model is central in MCL as it computes particles weight given a particular pose. Overall, it is used to estimate which particle is more likely to represent the true position of the robot. To compute this likelihood, we compare the descriptor obtained from sensor readings with the expected descriptor from the map by estimating their Hamming distance d_{ham}.

Since our compact descriptor contains only 4 bits, there are five possibilities of descriptor distances, from zero to four, where zero results from perfectly similar descriptor and four corresponds to complete opposite descriptor. The weight of every particle w_p then can be computed as:

$$w_p = 1 - 0.2\, d_{ham}. \tag{3}$$

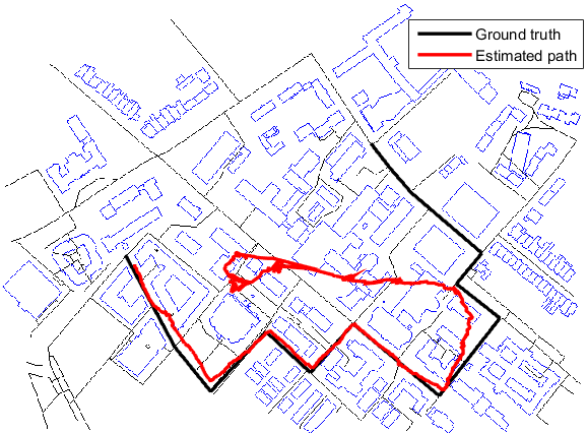
The particle with the largest distance gets the weight 0.2. In this way, we ensure that particles with higher Hamming distance are penalized more but are not eliminated completely.

IV. Experimental Evaluation

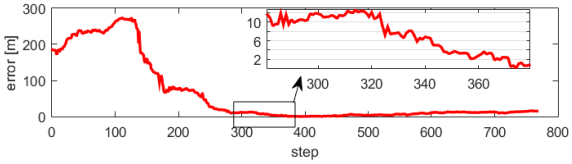
Our evaluation is designed to illustrate the properties of our proposed approach for localization on OpenStreetMap data. We support our claims by providing experiments in simulations as well as using real world data. To evaluate the accuracy of the particle filter, we compute mean localization error as a distance from the mean particle to the ground truth estimate for each individual step of the filter. For the real world experiments, we used the LiDAR measurements and ground truth poses from the KITTI dataset.

A. Simulated Data

The first experiment is designed to show the performance of our global localization system in a controlled, simulated environment. We computed a map of the city of Bonn, Germany, from the OpenStreetMap and generate a potential robot trajectory within this map as shown in Fig. 7a. The size of the map corresponds to around 500 m by 700 m and the trajectory is 767 m long. By using the simulated trajectory, we obtain noise-free measurements as well as ground truth poses. This allows us to confirm that the compact binary descriptor is powerful enough to perform global localization,



(a) Global localization paths



(b) Localization error

Fig. 7: Localization paths and errors using noise-free measurements for illustration purposes. Localization is achieved with an accuracy of 12 m after around 300 steps. The red trajectory illustrates the weighted mean of the particles. During the first 250 steps, the filter was not converged yet and thus the weighted particle poses are a suboptimal estimate trajectory and do not overlay with the roads. After convergence, however, the pose can be tracked well.

as can be seen from Fig. 7b. For the simulations, we added Gaussian noise with mean zero and standard deviation 0.01 deg for rotation and 0.2 m for translation to the motion model. Real world data will impose perceptual aliasing for such simplistic descriptors since there are a lot of places in the 2D map that generate the same descriptors and it is hard within one observation to disambiguate the poses—even if the measurements would be noise-free, which they never are in reality. Nevertheless, as can be seen in Fig. 7b, by using the semantic descriptor within the particle filter framework, the robot is successfully localized within approx. 300 steps with a mean localization error reduced to 12 m.

The second experiment is designed to show that the proposed system provides robust global localization under added random as well as systematic noise. Here, we use the same trajectory with noise-free measurements and the same motion noise from the previous experiment. We randomly flip 10, 20, 25, 30, 40 and 50 percent of all the bits in the measurement descriptors. As Fig. 8 suggests, we are able to globally localize the robot with an accuracy of 30 m for noise levels up to 30%.

To model more realistic noise scenarios, we furthermore added systematic noise to the measurements. This situation typically occurs in the real world whenever the detection algorithm fails for several continuous frames to detect an intersection or a building gap. Such a systematic noise can be simulated by flipping the totally perfect 4-bit descriptors for several continuous steps. Note that the influence of

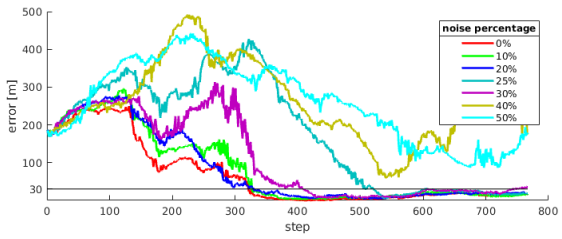


Fig. 8: Localization error using measurements with random noise.

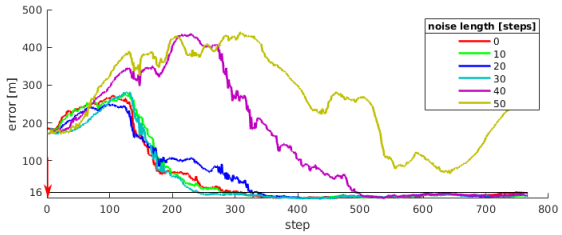


Fig. 9: Localization error using measurements with systematic noise. Systematic noise is added from the first step.

the systematic noise depends not only on its length but also where the noise is added. Since at the initial steps there are fewer particles nearby the true pose, this makes the localization more prone to fail. So we chose to add a systematic noise with the length of 10, 20, 30, 40 and 50 meters starting from the first step. As Fig. 9 shows, a systematic noise with length up to 30 m at the first step shows a similar, solid localization performance. From our experience adding similar noise after the filter convergence has little influence on localization result.

B. Real-World Data from KITTI

This experiment is designed to show the similarity of the descriptors extracted from the map and from the real world scans. For that, we use the KITTI dataset, sequence 00, which contains 4541 Velodyne LiDAR observations. Every observation leads to a descriptor with 4 bits. We compare all bits from real observations against bits extracted from the map and estimate the percentage of different bits. As Tab. I shows, by applying our descriptor extraction scheme from Sec. III-C, the bits difference for the intersection detection is 16.8% and 28.9% for detecting building gaps. The total difference in bits is 22.8%, which is according to previous experiments in a tolerable noise level for our particle filter to deal with. Among the different bits, we also distinguish between the false negative observations and the false positive ones. As we can see, there are more undetected building gaps (22.6%) than the wrongly identified ones (6.3%). This might be because some occluded building pixels on the map are detected as observable from the perspective of the laser scanner, and thus, there are more building gaps detected from the map than from the range images.

The next experiment is designed to show the localization accuracy on real world data. Fig. 10 depicts the map of the KITTI sequence 00 generated from OpenStreetMap with 1 meter per pixel resolution with the ground truth path and our

TABLE I: Total bit difference between descriptors detected from real data (KITTI 00) and extracted from the map.

Type	Different bits	False negatives	False positives
Intersection	16.8%	9.3 %	7.5%
Gap	28.9%	22.6%	6.3%
Total	22.8%	15.9%	6.9%

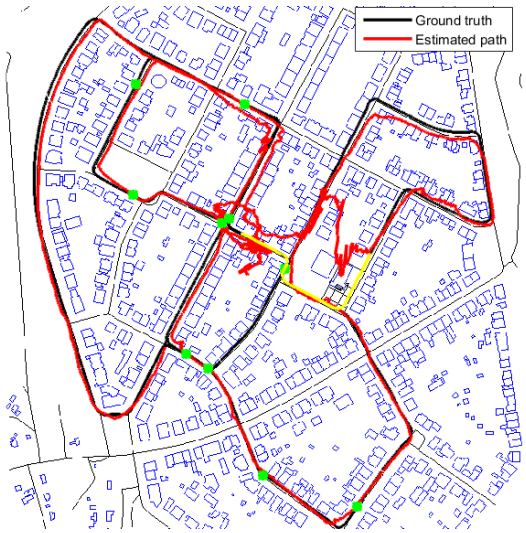


Fig. 10: Global localization paths. Green: sequences starting frames. Yellow: Area in which the OSM data is inconsistent. Red: weighted mean of the particles. During filter initialization, the weighted particle poses are a suboptimal estimate trajectory and do not overlay with the roads. After convergence, however, the pose can be tracked well.

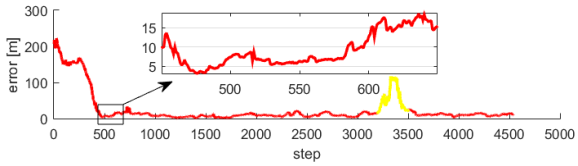


Fig. 11: Localization for KITTI sequence 00 is achieved within 20 m accuracy after approximately 500 steps. Yellow corresponds to the divergence points as in Fig. 10.

estimated path. As the corresponding error curve in Fig. 11 shows, our approach converges at around 500 steps with the localization accuracy of 20 m. The localization accuracy of our system degrades around timestamp $3200 - 3500$ since the sidewalk segments are marked as roads. This leads to inconsistencies between the expected and real descriptors. Nevertheless, our system is able to recover in subsequent time stamps.

To provide a more quantitative evaluation, we subdivide this sequence into 10 parts with the starting points from the 200^{th} time step with the length of 2000 steps and then creating new sequences with a start increment of 200 steps, which are marked as the green asterisks in the Fig. 10. When starting from different points our localization system is able to localize the robot with 50 m accuracy within 500 frames, with 20 m within 600 frames and with 10 m within 1000 frames on average as shown in Fig. 12.

We have further conducted experiments on more KITTI

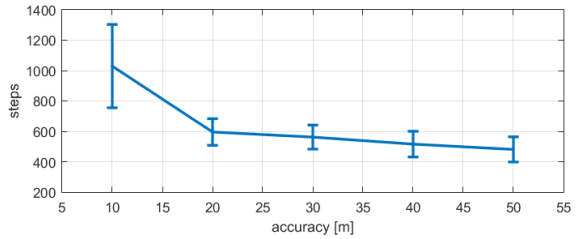


Fig. 12: Global localization accuracy versus the required number of steps. 95% confidence intervals are indicated by the bars.

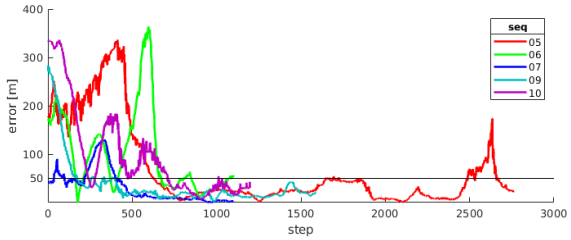


Fig. 13: Localization results of several KITTI sequences.

sequences that run in residential areas (05, 06, 07, 09, 10). The results are displayed in Fig. 13. Localization is achieved after around 500 to 800 steps within an accuracy of 50 m for most sequences. However, the reduction of measurement into such a simple feature comes at a price of lower accuracy and longer convergence time when compared to the result of [14], where the authors take advantage of full laser scans and successfully localize a robot with an accuracy of 10 meters after around 250 steps.

C. Computational Efficiency

In the last experiment, we show that by deploying the compact binary descriptor, we need only a small amount of memory for storage. We can furthermore precompute all possible expected observations for a given map for all possible robot locations with 5 degrees angular resolution. This results in 5.02 MB of space for an area of approx. $650 \times 600\,m$ in Karlsruhe and 2.64 MB for an area of approx. $500 \times 700\,m$ in Bonn, both with a map resolution of 1 m per pixel. Due to the small size of the descriptors, the particle filter update can be performed efficiently. The descriptor from the real observation should only be computed once per measurement which takes around $232\,ms$ on an intel core i7 notebook. Afterwards, updating the weight of each particle involves only comparing two descriptors from real and expected observations, which is done highly efficiently by computing the Hamming distance between the two 4-bit vectors.

V. Conclusion

In this paper, we presented a novel approach for global robot localization to estimate a robot's pose given publicly available maps using a compact semantic 4-bit descriptor. Our method exploits the results of a semantic segmentation system and detects intersections and building gaps from a range image using a set of geometric checks. This information is then integrated into the observation model used for localization. We implemented and evaluated our approach in simulations as well as on real world datasets and illustrated the localization performance of our approach. We show that it is possible to localize a robot in publicly available maps using MCL requiring only a compact 4-bit descriptor to describe a place as well as an observation.

References

[1] P. Agarwal, W. Burgard, and L. Spinello. Metric Localization using Google Street View. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. *arXiv*, 2019.

[3] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer Tracts in Advanced Robotics*, pages 143–157. Springer Verlag, 2006.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

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

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[7] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34, 2007.

[8] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Speeding-up rao-blackwellized SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 442–447, Orlando, FL, USA, 2006.

[9] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A Navigation System for Robots Operating in Crowded Urban Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, Karlsruhe, Germany, 2013.

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

[11] M. Milford. Vision-based place recognition: how low can you go? *Intl. Journal of Robotics Research (IJRR)*, 32(7):766–789, 2013.

[12] A. Milioto and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. Accepted for publication.

[13] P. Panphattarasap and A. Calway. Automated Map Reading: Image Based Localisation in 2-D Maps Using Binary Semantic Descriptors. *arXiv preprint*, 2018.

[14] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard. Localization on OpenStreetMap Data Using a 3D Laser Scanner. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[15] R. Spangenberg, D. Goehring, and R. Rojas. Pole-Based Localization for Autonomous Vehicles in Urban Scenarios. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[16] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard. Analyzing Gaussian Proposal Distributions for Mapping with Rao-Blackwellized Particle Filters. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[17] B. Suger and W. Burgard. Global Outer-Urban Navigation with OpenStreetMap. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[18] O. Vysotska and C. Stachniss. Exploiting building information from publicly available maps in graph-based slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[19] O. Vysotska and C. Stachniss. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, 2019.