Localization on OpenStreetMap Data using a 3D Laser Scanner

Philipp Ruchti   Bastian Steder   Michael Ruhnke   Wolfram Burgard

***Abstract*— To determine the pose of a vehicle is a fundamental problem in mobile robotics. Most approaches relate the current sensor observations to a map generated with previously acquired data of the same system or by another system with a similar sensor setup. Unfortunately, previously acquired data is not always available. In outdoor settings, GPS is a very useful tool to determine a global estimate of the vehicles pose. Unfortunately, GPS tends to be unreliable in situations in which a clear view to the sky is restricted. Yet, one can make use of publicly available map material as prior information. In this paper, we describe an approach to localize a robot equipped with a 3D range scanner with respect to a road network created from OpenStreetMap data. To successfully localize a mobile robot we propose a road classification scheme for 3D range data together with a novel sensor model, which relates the classification results to a road network. Compared to other approaches, our system does not require the robot to actually travel on the road network. We evaluate our approach in extensive experiments on simulated and real data and compare favorably to two state-of-the-art methods on those data.**

I. Introduction

One essential prerequisite for autonomous navigation for cars or mobile robots is to know the pose of the vehicle in the world. For example, without this information a mobile robot would not be able to plan a path to a desired goal. The most common localization method in outdoor settings is the global positioning system (GPS). While GPS provides a global position, the accuracy of the pose estimate depends on the number and distribution of visible satellites. Especially in cities with high buildings or under tree canopies GPS can suffer from severe outages. The goal of this work is to enable mobile robots to perform robust navigation even under such circumstances.

To overcome this problem, many autonomous mobile robots or self-driving cars localize themself within highly accurate maps of the environment, built with range or vision sensors [15], [9]. While this approach yields highly accurate results, it requires a substantial effort to obtain such maps from the sensor data of the mobile robot in advance and to maintain them afterwards.

In this paper, we propose an alternative solution and present an approach to localize a mobile robot given publicly available maps, like OpenStreetMap [12], which provide a dense description of the public road network around the planet. To achieve this, our approach performs a classification of the observations obtained with a 3D laser scanner. In addition, it contains a dedicated sensor model for a probabilistic approach based on a particle filter. Compared to other approaches that perform localization on publicly available maps [2], [4], we do not make the assumption that the robot has to drive on the road network. It only has to be in a reasonable proximity to be able to observe the road network from time to time. Figure 1 illustrates the basic principle of our approach.

All authors are members of the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany. {ruchtip,steder,ruhnke,burgard}@informatik.uni-freiburg.de

This work has been partly supported by the European Commission under the grant numbers ERC-AG-PE7-267686-LifeNav and FP7-610603-EUROPA2

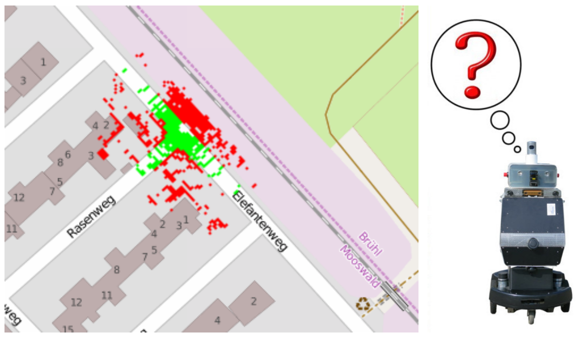


Fig. 1. Our system localizes a mobile robot equipped with a 3D range scanner in a publicly available OpenStreeMap.

II. Related Work

Localization in road networks has received quite some attention in the context of autonomous navigation with cars. For example during the DARPA urban challenge all teams had to localize their vehicle on a road network [8], [11], [16]. Most of the systems in this context used a fused localization estimate based on GPS, odometry, and a very accurate inertial measurement unit (IMU). Such methods typically provide locally very accurate motion estimates and can determine the global pose within a few meters, but rely on very expensive, specialized hardware to do so. One of the first methods using a Monte Carlo filter to localize a robot was proposed by Dellaert *et al.* [3]. In their experiments they used sonar readings or laser scans to localize a robot in an occupancy grid map. Since then many different approaches employed Monte Carlo Localization (MCL) to localize a robot [7], [13], [17]. Floros *et al.* [4] localize a robot on an OpenStreetMaps road network using visual odometry. They use a history of odometry poses to match against the road network using fast oriented chamfer matching. The authors assume that the shape of this odometry path resembles the shape of the road. While this approach enables fast and robust localization it is restricted to robots driving on roads. Brubaker *et al.* [2] provide a method for graph based localization on OpenStreetMaps using visual odometry. They represent the robots pose explicit on the edges of the road graph. The authors provide a probabilistic transition model to move the particles on the graph. Hentschel *et al.* [6] use OpenStreetMap data for localization, to perform path planning, and autonomous vehicle control in an urban environment. In contrast to our approach, they mostly use the shape of buildings to localize the robot using 2D laser scans. This approach allows the robot to leave the road but needs an urban environment with known shapes of the buildings. Kümmerle *et al.* [10] present an approach to localize a mobile robot equipped with a 3D range scanner in an aerial image, using Monte Carlo Localization. Compared to our approach, aerial images contain richer information of the environment, e.g., buildings or trees, compared to just a road network.

In this paper, we propose a system to localize a robot, equipped with a 3D laser scanner, with respect to a road network from OpenStreetMap. We apply a supervised classification approach to classify laser scans into road and non-road. This classification is then used in a corresponding sensor model to weight the particles of a Monte Carlo Localization. In contrast to the above-mentioned methods, which localize a robot on a road network using only odometry, our method does not require that the robot actually travels on the road network.

III. MCL-based Localization on Road Networks

The goal of our system is to find the pose of a robot relative to a given road network, based on a sequence of 3D laser scans, odometry measurements, and a rough initial position of the robot (within a few hundred meters).

In the following we will describe our Monte Carlo Localization approach, including a novel sensor model, as well as the classifier we use to distinguish road from non-road in the robot's measurements.

A. Monte Carlo Localization

In our work, we use a particle filter [14] to perform Monte Carlo Localization. A particle filter in the context of localization represents the probability distribution over the pose x_t of the robot with a finite set of weighted hypotheses called particles. To maintain a believe about the pose of the system, a particle filter performs two steps. The first step is the motion update, which modifies the pose hypothesis of each particle using the action u_t, the map m and the previous state x_{t-1} by sampling x_t according to $p(x_t \mid u_t, x_{t-1}, m)$. In the second step, we relate the measurements to the map according to our sensor model. More precisely, we weight the particles using the measurement z_t according to our sensor model $p(z_t \mid x_t, m)$. Afterwards, we resample a new set of particles from the old ones, where the chance of survival for each particle is proportional to its weight in the old particle set.

B. Our Implementation of the Monte Carlo Filter

One general problem of approximating a probability distribution with a finite set of particles is that good hypotheses might not survive the resampling step. Therefore, we use the number of effective particles to measure the quality of the particle distribution and resample only when this number is below a given threshold. The number of effective particles [5] is computed as

$$\mathrm{N}_{\text{eff}} = 1 / \sum_{i=1}^{N} \left(w^{(i)} \right)^2 . \tag{1}$$

Resampling is only performed if $\mathrm{N}_{\text{eff}} < N/2$, where N is the number of particles. For the motion update we use odometry readings from the wheel encoders or equivalent simulated data.

C. Classification

Since our map encodes only road information we first need to retrieve this information from the 3D range readings. Therefore, we have to decide whether regions in range scans observe road or non-road surfaces. Throughout our experiments we perform road classification on single scans from a Velodyne HDL-32E Lidar, but any kind of 3D laser scanner (e.g., tilting/rotating 2D scanners) can be used with our method. Our method expects 3D scans that include additional reflectance values. Our classification is defined as a function mapping from discretized cells z_i to the classes road or non-road

$$c : z_i \mapsto \{\text{road}, \neg\text{road}\} .$$

To calculate this classification, we project the scan into a two dimensional grid and classify the single cells independently. From all points falling into one cell, we calculate the following features: the mean and standard deviation of the height values, the mean of the squared intensity values, the standard deviation of the intensities, the distance to a fitted plane, the normal vector of the fitted plane, and the maximum difference in height values. Based on those local features we learn a classifier using boosting [1] in a supervised fashion. Cells with no or too few points to calculate features are neglected.

To train the classifier, we first collected a dataset (different from the one used in the experiments) with a robot and manually labeled the regions in the scans as being road and non-road. This leads to a classified set of features which we use to train our classifier.

D. Weighting / Sensor Model

The task of the sensor model is to determine the likelihood $p(z \mid x, m)$ of a measurement z, given the robot is at pose x in the map m. In our approach, the input to our sensor model is the road classification in form of a 2D grid map, in which each cell is either unobserved, road or, non-road. In the following, we explain how we relate the road classification to the provided road network.

Under the assumption that the grid cells from our sensor measurement are independent, the likelihood of a measurement z, composed of the classified grid cells $z_1, \ldots, z_{|z|}$, can be calculated as

$$p(z \mid x, m) \propto \prod_{i=1}^{|z|} f(z_i, m), \tag{2}$$

where $f(z_i, m)$ is the likelihood of one cell in the measurement given the map. We model the likelihood for each observed point as a Gaussian

$$f(z_i, m) = \mathcal{N}(\varepsilon(z_i, m), 0, \sigma_{z_i}), \tag{3}$$

with mean 0 and a user defined standard deviation that is individually defined for road and non-road measurements. The term $\varepsilon(z_i, m)$ is an error based on the distance to the next road in the map, which can be efficiently determined using pre-computed distance transform maps.

1) Cost for Road Cells: The calculation of $\varepsilon^r(z_i, m)$ for road cells is supposed to penalize measured road cells that do not correspond to the roads appearing in the map. Such regions lead to a higher distance to the road and thereby increase the error for this match. The error $\varepsilon^r(z_i, m)$ for cells classified as road is calculated for all cells z_i with $c(z_i) =$ road as follows

$$\varepsilon^r(z_i, m) = \min\left(d, \max\left(0, |z_i - s|\right)\right), \tag{4}$$

where s is the closest point on the road network and d is the maximum allowed distance to a road. The term $\varepsilon^r(z_i, m)$ therefore describes the distance between the observed cell and the closest road in the road network. Cells with a higher distance to the closest road than d will not be penalized further. The usage of the error ε^r in the particle filter leads to a situation that particles are kept in regions where most of the cells that are classified as road are close to a road in the map. The effect of this error can be seen in Figure 2[left]. Removing the dotted side road results in higher errors for all cells, classified as road (green) which are in the blue ellipse. The closest road in the map is now further away, which will increase the error for this match. On the other hand, particles

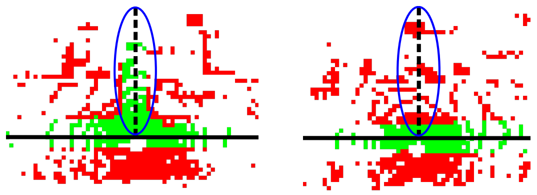


Fig. 2. Classified example scans. Red cells are classified as non-road, green cells are classified as road, and black lines show the road network. Removing the dotted road in the left figure increases the errors ε^r as the next road in the map for the cells in the blue ellipse is now further away. Adding the dotted road in the right figure increases the error ε^n as the next road in the map for the cells in the blue ellipse is now closer.

tend to stick in regions with several nearby-roads or multi-lane crossings, because many roads in the vicinity tend to decreases the error for false classifications and side roads in contrast to single lanes.

2) Cost for Non-Road Cells: For non-road cells the calculation of $\varepsilon^n(z_i, m)$ is supposed to penalize situations where there are more roads in the map than can be explained by the scan, such as a side road that does not appear in the scan. Equivalently to the case for road cells we define

$$\varepsilon^n(z_i, m) = d - \min\left(d, \max\left(0, |z_i - s|\right)\right). \tag{5}$$

This term is small for points that are far away from the closest road in the road network and large for points that are close to roads. If we have a scan with only a straight road segment as shown in Figure 2[right], we increase the error for this match whenever we try to match it against a region of the map including an additional side road (dotted line). This is due to the fact, that the cells inside the blue ellipse now have a smaller distance to the next road and thereby a higher value of $\varepsilon^n(z_i, m)$.

3) Computing the Likelihoods: Formula 2 can be efficiently computed using log-likelihoods as follows

$$p(z \mid x, m) \propto \exp\left(-\sum_{i,\text{road}} \frac{(\varepsilon_i^r)^2}{\sigma_1^2} - \sum_{i,\neg\text{road}} \frac{(\varepsilon_i^n)^2}{\sigma_2^2}\right), \tag{6}$$

whereas $\varepsilon_i^r = \varepsilon^r(z_i, m)$, $\varepsilon_i^n = \varepsilon^n(z_i, m)$, σ_1 being the assumed standard deviation for road cells, and σ_2 being the assumed standard deviation for non-road cells. Larger values of σ_1 or σ_2 lead to more peaked, thereby stricter distributions.

IV. Experimental Evaluation

To evaluate our method, we performed extensive experiments on three different datasets. In the first experimental setting, we decided to evaluate our method in simulation since this provides us with ground truth, which is hard to obtain for real world experiments. In a second experimental setting, we evaluated our method on a real world dataset providing evidence that both the classification scheme and the sensor model are robust to the typical noise introduced by real 3D range scanners. Furthermore, we compared our method against two other state-of-the-art approaches on three different datasets. In the following, we briefly explain those methods.

A. Distance to Road

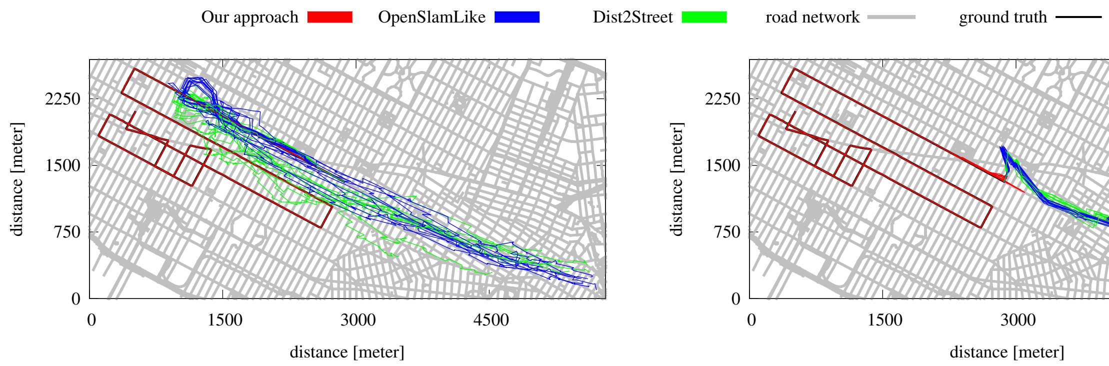
The first method calculates the weight for a particle involving its distance to the next road in the map. More precise we define the weight as $w = \exp\left(|x - s|/\sigma\right)$, where x is the position of the particle and s is the next point on a road. As in our approach, σ defines the shape of the weighting distribution. This method favors regions of the map with a lot of roads close to each other.

B. Chamfer Distance Based

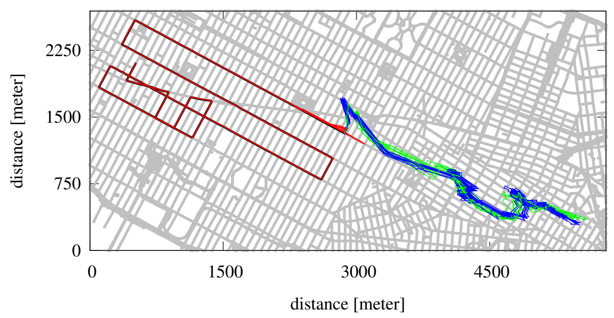
The second method is based on chamfer matching and is comparable to the method by Flores *et al.* [4]. For this method the particle filter stores the path calculated from the odometry readings over a fixed time or distance window. To weight a particle we calculate the distance from each point on this path ending at the particle pose to the road map. This is comparable to the chamfer matching score of the path (template image) to the road map (query image) at the position and orientation given by the pose of the particle.

C. Initialization and Parameters

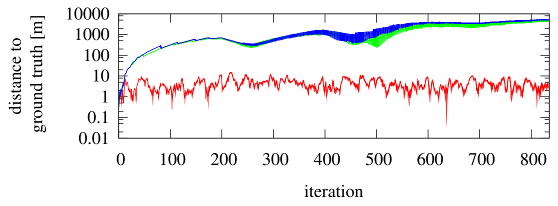
For our tracking experiments we initialize the particle filter by sampling 2,000 particles from a Gaussian with a standard deviation of 20 m around the true or GPS position. We then sample the orientation from a Gaussian with a



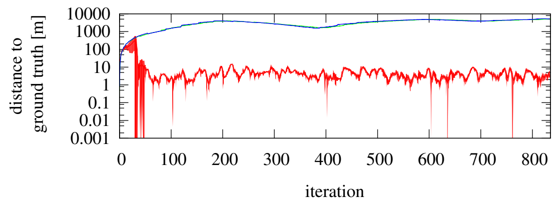
(a) tracking paths



(b) global localization paths

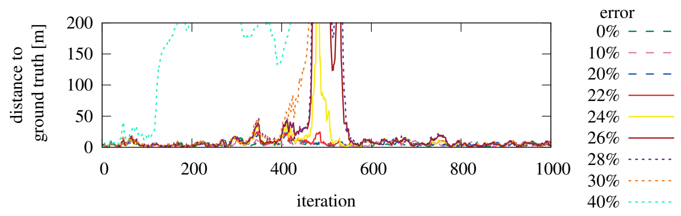


(c) tracking errors

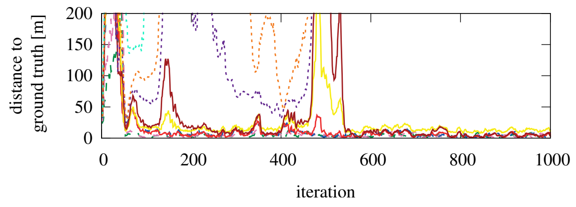


(d) global localization errors

Fig. 3. Paths and errors of the first simulated experiments. Ten runs were executed for each of the three methods. The left side shows the paths and errors with their standard deviation over the ten runs created at the task of tracking, the right side shows the same for the task of global localization. The errors are the differences between the weighted mean of all particles and the ground truth. The scale of the y-axis in (c) and (d) is logarithmic.



(a) global localization errors



(b) global localization errors

Fig. 4. Position error for the weighted mean of the particles for tracking (left) and global localization (right) shown for different rates of the classification error. Our method allows up to 22% classification error without a substantial decrease in the localization accurancy.

standard deviation of 0.15 rad around the true orientation or the measurement from the compass of the robot. For the global localization we use 20,000 particles which we distribute evenly in a radius of 250 m around the true position and sample the orientation randomly. For each experiment we use ten runs per method. For all experiments we use the same parameters. To discretize the scans we use a cell size of 1 m. We choose $1/\sigma = 1/\sigma_1 = 0.0003$ and $1/\sigma_2 = 0.00015$. For the maximum distance to the road we use $d = 10\,\text{m}$.

D. Simulation Experiments

In the simulation experiments we want to investigate the performance of the weighting given a perfect classification. We therefore create scans by copying a local surrounding in a radius of 15 m around the requested pose from the roads of an OpenStreetMap. We then generate an odometry path with additional white noise and ground truth poses which we use for evaluation. In these simulation-based experiments we want to demonstrate that a classified scan can increase the localization performance in contrast to methods using odometry only. This increase is due to the fact that we are able to observe roads that we did not (yet) drive on. These experiments are carried out on a cutout of the OpenStreetMap for Manhattan. We evaluated how the methods perform on the task of tracking the position of the robot on the map. The paths created by the different methods are shown in Figure 3(a). The trajectory estimated by our method (red) generates paths that closely resemble the ground truth path. Our method can take advantage of observations of crossings, especially with the diagonal road (running horizontal in the center of this map area), whereas the other two methods suffer from the long straight paths and the ambiguities. Figure 3(c) shows the distance from the weighted mean of the particles to the true pose over time. We also performed global localization on the same dataset. Figure 3(b) shows the paths, while Figure 3(d) shows the corresponding errors. In the error plot we can see that after around 30 iterations our approach is able to find the right road, whereas the other methods are not able to resolve the ambiguities.

In this paper we use a very basic classifier to distinguish between road and non-road in the scans. To see how much our method suffers from classification errors, we randomly flipped a defined amount of cells of the perfect classification and repeated the previous experiment. A plot of the resulting

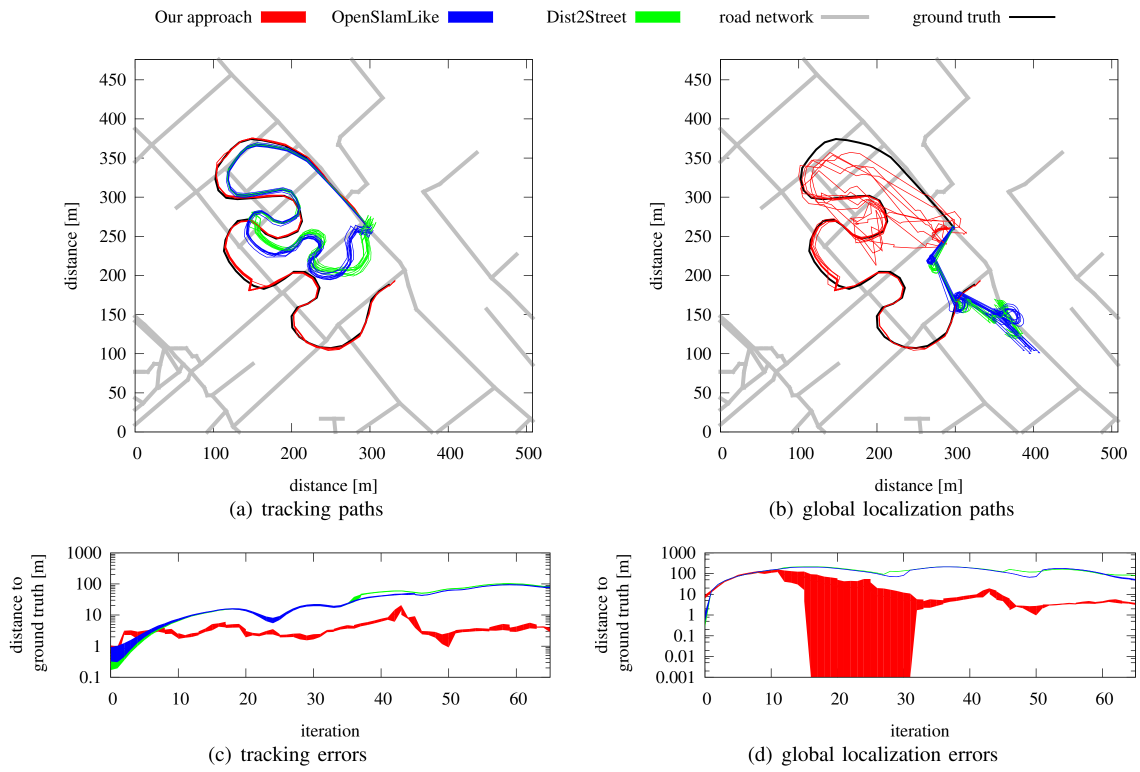


Fig. 5. Paths and errors of the simulated offroad experiments. We evaluated each of the three methods using ten runs. The left side shows the paths and errors with their standard deviation over the ten runs created at the task of tracking, the right side shows the same for the task of global localization. The errors are the differences between the weighted mean of all particles and the ground truth. The scale of the y-axis in (c) and (d) is logarithmic.

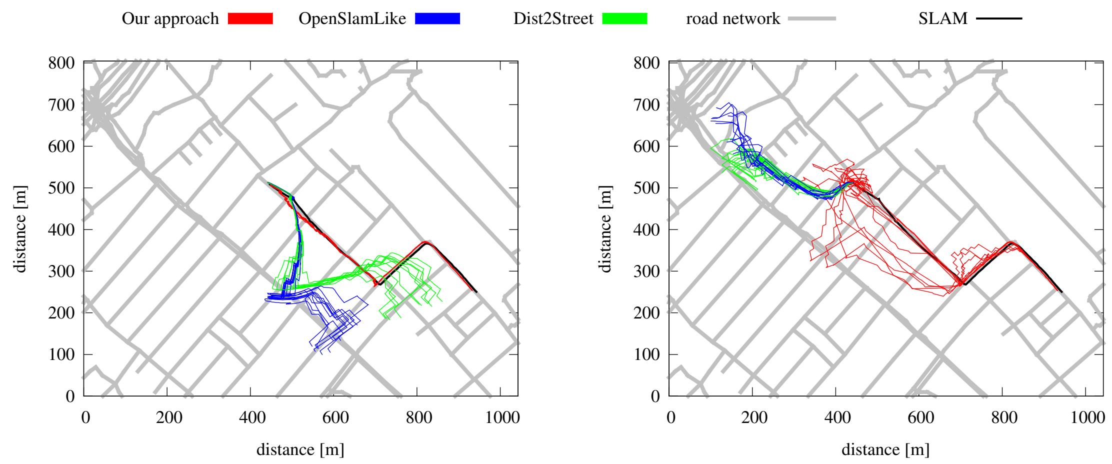
difference of the weighted mean of all particles to the ground truth is shown in Figure 4. We can see that our method performs well even if 20% of the cells are falsely classified. Since the Particle filter only relies on the classification result and the odometry of the robot, any classification method using arbitrary sensors could be used, even if the classification is sub-optimal.

E. *Simulated Offroad Data*

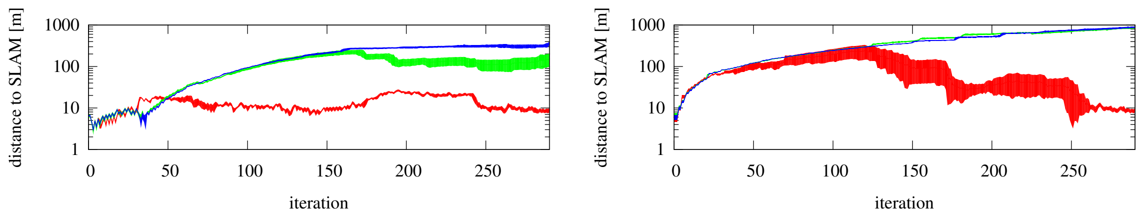
Using laser scans we are able to localize the robot even if it does not drive on the road network. In this experiment, we simulated a run on a map of Freiburg. As before we copy a local surrounding of 15 m from a road network to simulate perfect classified scans. We compare our method against two other approaches that use only odometry and assume that the robot drives on the road. This assumption is violated during this experiment. On the tracking task our method outperforms the other two methods, as shown in Figure 5(a). As expected, the other methods are able to follow the shape of the path but are always drawn towards the next roads, which leads to a higher error (see Figure 5(c)). The global localization (see Figures 5(a) and 5(c)) also performs as expected. Our method is able to converge to the correct position after roughly half of the trajectory, whereas the other two methods diverge.

F. *Robot Experiments*

Furthermore, we evaluated our approach on data from one of our robots. We use the robot *Obelix* which is shown in Figure 1 on the right side. This robot provides odometry from wheel encoders and 3D scans with reflectance from a Velodyne HDL-32E. We collected data in an urban environment. The classification was trained on a separate data set collected with the same robot. Since no ground truth is available for this real-world experiment, we use the result of a graph-based SLAM system that also incorporates GPS meassurements instead. As in the simulated setting we perform two experiments using the same data set. The first experiment evaluates the tracking performance while the second one tests the global localization. Please note that the odometry of the robot is not properly calibrated and suffers from high deterministic rotational errors. Figures 6(a) and 6(c) show the result for all three methods for tracking. We see that our approach is able to keep the correct track. The other methods suffer from the high rotational errors of the odometry and some of the particles incorrectly turn right early in the trajectory. Our method is able to recover from this situations. Figures 6(b) and 6(d) show the results for global localization. In this dataset the robot first drives on a small straight pedestrian path. On this path the robot is able to perceive two junctions. These junctions allows our method to improve the filter belief with respect to the correct position. After the first turn the filter is able to converge. The other methods just see the two turns, which is insufficient to localize the robot. The runtime of our approach was 95 seconds on a standard i7 desktop machine. Note that the robot traveled for about 13 minutes to collect the dataset. Therefore our system is applicable for online operation.



(a) tracking paths (b) global localization paths



(c) tracking errors (d) global localization errors

Fig. 6. Results for the real robot experiments. We evaluated each of the three methods using ten runs. The left side shows paths and errors with their standard deviation over ten runs created at the task of tracking, the right side shows the same for the task of global localization. The errors are the differences between the weighted mean of all particles and our SLAM result including GPS measurements. The scale of the y-axis in (c) and (d) is logarithmic.

V. Conclusion

In this paper, we presented an approach to localize a mobile robot with respect to a road network such as the one provided by OpenStreetMap using 3D range scans. In our approach we employ a classifier to distinguish road from non-road in the scans and use the classified scans as the sensory input of a Monte Carlo Localization approach. In practical experiments, both in simulation and on real world data, we showed that our method can reliably perform global localization as well as pose tracking for a robot even in challenging situations where other state-of-the-art approaches fail. The presented method has the advantage that it does not require the robot to actually travel on a road.

References

[1] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

[2] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! leveraging the crowd for probabilistic visual self-localization," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] 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)*, 2013.

[5] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[6] M. Hentschel and B. Wagner, "Autonomous robot navigation based on openstreetmap geodata," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2010.

[7] P. Jensfelt, D. J. Austin, O. Wijk, and M. Andersson, "Feature based condensation for mobile robot localization," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2000.

[8] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, *et al.*, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[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 Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[10] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Journal of Autonomous Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[11] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[12] OpenStreetMap contributors. (2014) OpenStreetMap. [Online]. Available: http://www.openstreetmap.org

[13] J. Rowekamper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[15] C. Urmson. (2014) The latest chapter for the self-driving car: mastering city street driving. [Online]. Available: http://googleblog.blogspot.de/2014/04/the-latest-chapter-for-self-driving-car.html

[16] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[17] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization by combining an image-retrieval system with monte carlo localization," *Robotics, IEEE Transactions on*, vol. 21, no. 2, pp. 208–216, 2005.