Benchmarking Particle Filter Algorithms for Efficient Velodyne-Based Vehicle Localization

Jose Luis Blanco-Claraco [1] , **Francisco Mañas-Alvarez** [2] , **Jose Luis Torres-Moreno** [1,*] ,
Francisco Rodriguez [2] and **Antonio Gimenez-Fernandez** [1]

[1] Engineering Department, University of Almería, 04120 Almería, Spain
[2] Computer Science Department, University of Almería, 04120 Almería, Spain
*   Correspondence: jltmoreno@ual.es; Tel.: +34-950-214-232

Abstract: Keeping a vehicle well-localized within a prebuilt-map is at the core of any autonomous vehicle navigation system. In this work, we show that both standard SIR sampling and rejection-based optimal sampling are suitable for efficient (10 to 20 ms) real-time pose tracking without feature detection that is using raw point clouds from a 3D LiDAR. Motivated by the large amount of information captured by these sensors, we perform a systematic statistical analysis of how many points are actually required to reach an optimal ratio between efficiency and positioning accuracy. Furthermore, initialization from adverse conditions, e.g., poor GPS signal in urban canyons, we also identify the optimal particle filter settings required to ensure convergence. Our findings include that a decimation factor between 100 and 200 on incoming point clouds provides a large savings in computational cost with a negligible loss in localization accuracy for a VLP-16 scanner. Furthermore, an initial density of $\sim$2 particles/m^2 is required to achieve 100% convergence success for large-scale ($\sim$100,000 m^2), outdoor global localization without any additional hint from GPS or magnetic field sensors. All implementations have been released as open-source software.

Keywords: global positioning system; mobile robots; simultaneous localization and mapping; particle filter; district

1. Introduction

Autonomous vehicles require a robust and efficient localization system capable of fusing all available information from different sensors and data sources, such as metric maps or GIS databases. Metric maps can be automatically built by means of Simultaneous Localization and Mapping (SLAM) methods onboard the vehicle or retrieved from an external entity in charge of the critical task of map building. At present, some companies already have plans to prepare and serve such map databases suitable for autonomous vehicle navigation, e.g., Mapper.AI or Mitsubishi's Mobile Mapping System (MMS). However, at present, most research groups build their own maps by means of SLAM methods or, alternatively, using precise real-time kinematic (RTK)-grade global navigation satellite system (GNSS) solutions. For benchmarking purposes, researchers have access to multiple public datasets including several sensor types in urban environments [1–3].

In the present work, we address the suitability of particle filter (PF) algorithms to localize a vehicle, equipped with a 3D LiDAR (Velodyne VLP-16, Velodyne Lidar, San Jose, CA, USA),within a previously-built reference metric map. An accurate navigation solution from Novatel (SPAN IGN INS, Novatel, Calgary, AB, Canada) is used for RTK-grade centimeter localization, whose purpose is twofold:

(i) to help build the reference global map of the environment without the need to apply any particular SLAM algorithm (see Figure 1), and (ii) to provide a ground-truth positioning to which the output of the PF-based localization can be compared in a quantitative way.
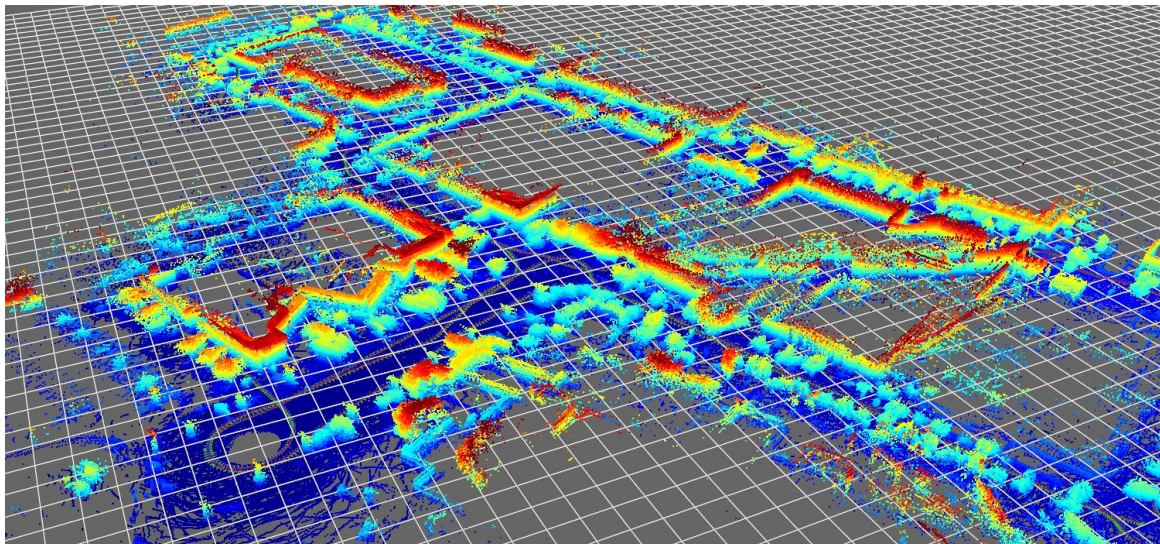
Figure 1. Overview of the ground-truth 3D map used in the benchmarks, representing $\sim$100,000 m^2 of the campus of the University of Almeria, points colorized by height.

The main contribution of this work is twofold: (a) providing a systematic and quantitative evaluation of the trade-off between how many raw points from a 3D-LiDAR must be actually used in a PF, and the attainable localization quality; and (b) benchmarking the particle density that is required to bootstrap localization, i.e., the "global relocalization" problem. For the sake of reproducibility, the datasets used in this work have been released online (refer to Appendix A).

The rest of this paper is structured as follows. Section 2 provides an overview of related works in the literature, as well as some basic mathematical background on the employed PF algorithms. Then, Section 3 proposes an observation model for Velodyne scans, applicable to raw point clouds. Next, Section 4 provides mathematical grounds of how a decimation in the input raw LiDAR scan can be understood as an approximation to the underlying likelihood function, and it is experimentally verified with numerical simulations. The experimental setup is discussed in Section 5. Next, the results of the benchmarks are presented in Section 6 and we end with some brief conclusions in Section 7.

2. Background

This section provides the required background to put the present proposal in context. We first review related works in Section 2.1, and next, Section 2.2 provides further details on the relevant particle filtering algorithms.

2.1. Related Works

As has been known by the mobile robotics community for more than a decade, SLAM is arguably more efficiently addressed by means of optimizing large sparse graphs of observations (some representative works are [4–7]) rather than by means of PF methods. The latter remain being advantageous only for observation-map pairs whose observation model is neither unimodal nor easy to evaluate in closed form, e.g., raw 2D range scans and occupancy grid maps.

On the other hand, localization on a prebuilt map continues to be a field where PFs find widespread acceptance, although there are few works in the literature where PFs are applied to the problem of 3D laser range finder (LRF)-based localization of vehicles, with many works favoring the extraction of features instead of using the raw sensor data. For example, in [8], the authors propose extracting a subset of features (subsets of the raw point cloud) from Velodyne (HDL-32E) scans, which are then matched using an iterative closest point algorithm (ICP) against a prebuilt map. Typical positioning errors obtained with this method fall in the order of one meter. Plane features are also extracted by Moosmann and Stiller in [9] from Velodyne (HDL-64E) scans to achieve a robust SLAM method. Interestingly, this work proposes randomly decimating (sampling) the number of features such that a maximum of 1000 planes are extracted per scan, although no further details are given regarding the optimality of this choice. Dubé et al. proposed a localization framework in [10] where features are first extracted from raw 3D LiDAR scans and a descriptor is computed for them. This approach has the advantage of a more compact representation of large/scale maps, enabling global re-localization faster than with particle filters, although their computational cost is higher due to the need to segment point clouds, compute descriptors, and evaluate the matching between them. As shown in our experimental results, particle filters with decimated 3D LiDAR scans can track a vehicle pose in ∼10 ms, whereas [10] takes ∼800 ms.

Among the previous proposals to use PFs in vehicle localization and SLAM, we find [11], where a PF is used for localizing a vehicle using a reflectance map of the ground and an associated observation likelihood model. A modified PF weight-update algorithm is presented in [12] for precise localization within lanes by fusing information from visual lane-marking and GPS. Their method is able to handle probability density functions that mix uniform and Gaussian distributions; such a flexibility would be also applicable to the optimal-sampling method [13] used in the present work, which is explained in Section 2.2. The integration of the localization techniques into vehicle systems' architecture demands more computationally efficient techniques [14,15]. When high level tasks are demanded, as cooperative driving, real-time performance is critical [16]. Rao-Blackwellized particle filters (RBPF) have been proposed for Velodyne-based SLAM in [17], using a pre-processing stage where vertical objects are detected in the raw scans such that RBPF-SLAM can be fed with a reduced number of discrete features. The flexibility of observation models in particle filters is exploited there to fuse information from two different metric maps simultaneously: a grid-map and the above-mentioned feature map. As can be seen, PF based methods still remain popular in SLAM. Nowadays, much research in this field focus on reducing their algorithms' computational cost. Thus, in [18], a map-sharing strategy is proposed in which the particles only have a small map of the nearby environment. In [19], the Rao-Blackwellized particle filter is executed online by using Hilbert maps. Other fields of interest are the cooperative use of particle filters for multi-robot systems [20] and the development of more efficient techniques for the integration of Velodyne sensors [21].

2.2. Particle Filter Algorithms

Particle filtering is a popular name for a family of sequential Bayesian filtering methods based on importance sampling [22]. Most commonly, PF in the mobile robot community is used as a synonymous for the Sequential Importance Sampling (SIS) with resampling (SIR) method, which is a modification of the Sequential Importance Sampling (SIS) filter to cope with particle depletion by means of an optional resampling step [23].

If we let x_t denote the vehicle pose for time step t, the posterior distribution of the vehicle pose can be computed sequentially by (the full derivation can be found elsewhere [24]):

$$p(\boldsymbol{x}_t|z^t,u^t) \propto \overbrace{p(z_t|\boldsymbol{x}^t,u^t)}^{\text{Observation likelihood}} \overbrace{p(\boldsymbol{x}_t|z^{t-1},u^t)}^{\text{Prior}}, \tag{1}$$

where z^t and u^t represent the sequences of robot observations and actions, respectively, for all past timesteps up to t. The posterior is approximated by means of a discrete set of $i = 1, \ldots, N$ weighted samples $x_t^{[i]}$ (called particles) that represent hypotheses ofthe current vehicle pose. To propagate particles from one timestep to the next, a particular proposal distribution $q(x_t | x^{t-1,[i]}, z^t, u^t)$ must be used, then the weights are updated accordingly by:

$$\omega_t^{[i]} \propto \omega_{t-1}^{[i]} \frac{p(z_t | x_t, x^{t-1,[i]}, z^{t-1}, u^t) p(x_t | x_{t-1}^{[i]}, u_t)}{q(x_t | x^{t-1,[i]}, z^t, u^t)}. \tag{2}$$

Most works in robot and vehicle localization assume that $q(\cdot)$ is the robot motion model $p(x_t | x_{t-1}^{[i]}, u_t)$ for convenience, since, in that case, Equation (2) simplifies to just evaluating the sensor observation likelihood function $p(z_t | x_t, \cdots)$. Following [13], we will refer to this choice as the "standard proposal" function. Despite its widespread use, it is far from the optimal proposal distribution [25], which by design minimizes the variance of particle weights, i.e., it maximizes the representativity of particles as samples of the actual distribution being estimated. Unfortunately, the optimal solution does not have a closed-form solution in many practical problems, hence our former proposal of a rejection sampling-based approximation to the optimal PF algorithm in [13]. In the present work, we will evaluate both PF algorithms, the "standard" (SIR with $q(\cdot)$ the motion model from vehicle odometry) and the "optimal" proposal distributions (as described in [13]), applied to the problem of vehicle localization. It is worth highlighting that the latter method is based on the general formulation of a PF, avoiding the need to perform scan matching (ICP) between point clouds and hence preventing potential localization failures in highly dynamic scenarios or in feature-less areas, where information from other sensors (e.g., odometry) is seamlessly fused in the filter leading to a robust localization system.

Our implementation of both PF algorithms features dynamic sample size, using the technique introduced in the seminar work [26], to adapt the computation cost to the actual needs depending on how much uncertainty exists at each timestep.

3. Map and Sensor Model

A component required by both benchmarked algorithms is the pointwise evaluation of the sensor likelihood function $p(z_t | x_t, m)$, hence we need to propose one for Velodyne 360° scans (z_t) when the robot is at pose $x \in SE(3)$ along a trajectory $x(t)$ given a prebuilt map m.

Regarding the metric map m, we will assume that it is represented as a 3D point cloud. We employed a Novatel's inertial RTK-grade GNSS solution to build the maps for benchmarking and also to obtain the ground-truth vehicle path to evaluate the PF output. This solution provides us with accurate WGS84 geodetic coordinates, as well as heading, pitch and roll attitude angles. Using an arbitrary nearby geodetic coordinate as a reference point, coordinates are then converted to a local ENU (East-North-Up) Cartesian frame of reference. Time interpolation of $x(t)$ is used to estimate the ground-truth path of the Velodyne scanner and the orientation of each laser LED as they rotate to scan the environment; this is known as de-skewing [9] and becomes increasingly important as vehicle dynamics become faster. From each such interpolated pose, we compute the local Euclidean coordinates of the point corresponding to each laser-measured range, then project it from the interpolated sensor pose in global coordinates. Repeating this for each measured range over the entire data set leads to the generation of the global point cloud of the campus employed as ground-truth map in this work.

Once a global map is built for reference, we evaluate the likelihood function $p(z_t | x_t, m)$ as depicted in Algorithm 1. First, it is worth mentioning the need to work with log-likelihood values when working with a particle filter to extend the valid range of likelihood values that can be represented within machine

precision. The inputs of the observation likelihood (line 1) are the robot pose $x(t)$, a decimation parameter, the list of all N points $\mathbf{p}_l^i$ in local coordinates with respect to the scanner, the reference map as a point cloud, a scaling σ value that determines how sharp the likelihood function is, and a smoothing parameter d_{max} that prevent underflowing. Put in words, from a decimated list of points, each point is first projected to the map coordinate frame (line 6), and the nearest neighbor is searched for within all map points using a K-Dimensional tree (KD-tree) (line 7). Next, the distance between each such local point and its candidate match in the global map is clipped to a maximum d_{max} and the squared distances accumulated into d^2. Finally, the log likelihood is simply $-d^2/\sigma^2$, which implies that we are assuming a truncated (via d_max) Gaussian error model as a likelihood function. Obviously, the decimation parameter linearly scales the computational cost of the method: larger decimation values provide faster computation speed at the price of discarding potentially valuable information. A quantitative experimental determination of an optimal value for this decimation is presented later on.

Algorithm 1 Observation likelihood.

Input: x, decim, $\{\mathbf{p}_l^i\}_{i=1}^{i=N}$, **map**, σ, d_{max}
Output: loglik

1: begin
2: loglik $\leftarrow 0$
3: $d^2 \leftarrow 0$
4: foreach i in 1:decim:N
5:   $\mathbf{p}_g^i \leftarrow x \oplus \mathbf{p}_l^i$
6:   $\mathbf{p}_{map\ closest}^i \leftarrow$ map.kdtree.query$(\mathbf{p}_g^i)$
7:   $d^2 \leftarrow d^2 + \min\{d_{max}^2, ||\mathbf{p}_g^i - \mathbf{p}_{map\ closest}^i||^2\}$
8: end
9: loglik $\leftarrow -d^2/\sigma^2$
10: return loglik
11: end

It is noteworthy that the proposed likelihood model in Algorithm 1 can be shown to be equivalent to a particular kind of robustified least-square kernel function in the framework of M-estimation [27]. In particular, our cost function is equivalent to the so-called truncated least squares [28,29], or trimmed-mean M-estimator [30,31].

With $x \in SE(3)$ the vehicle pose to be estimated, a least-squares formulation to find the optimal pose $x^\star$ that minimizes the total square error between N observed points and their closest correspondences in the map reads:

$$x^\star \quad = \quad \arg\min_{\mathbf{x}} \sum_{i=1}^{N} c_i^2(\mathbf{x}), \tag{3}$$

$$\text{with:} \quad c_i(\mathbf{x}) \quad = \quad \left|\left|\left(\mathbf{x} \oplus \mathbf{p}_l^i\right) - \mathbf{p}_{map\ closest}^i\right|\right|. \tag{4}$$

However, this naive application of least-squares suffers from a lack of robustness against outliers: it is well known that a single outlier ruins a least-squares estimator [30]. Therefore, robust M-estimators are preferred, where Equation (3) is replaced by:

$$x^\star \quad = \quad \arg\min_{\mathbf{x}} \sum_{i=1}^{N} f(c_i(\mathbf{x})) \tag{5}$$

with some robust kernel function $f(c)$. Regular least-squares correspond to the choice $f(c) = c^2$, while other popular robust cost functions are the Huber loss function [27] or the truncated least-squares function:

$$f(c) = \begin{cases} c^2 & |c| < \theta, \\ \theta^2 & |c| \geq \theta, \end{cases} \tag{6}$$

which is illustrated in Figure 2. The parameter θ establishes a threshold for what should be considered an outlier. The insight behind M-estimators is that, by reducing the error assigned to outliers in comparison to a pure least-squares formulation, the optimizer will tend to ignore them and "focus" on minimizing the error of inliers instead that is of those observed points that actually do correspond to map points.

(a) (b)
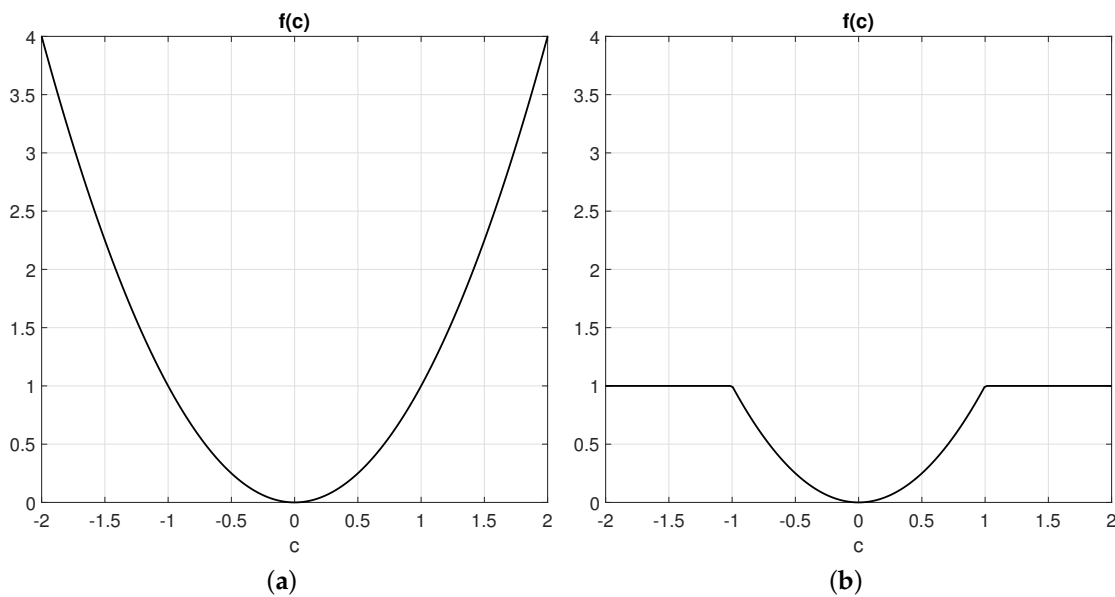
Figure 2. The cost function for regular least squares (**a**) and for truncated least squares (**b**) with $\theta = 1$. The latter significantly reduces the associated cost of outliers, thus removing their contribution to the actual cost function to be optimized.

Furthermore, this robust least-squares formulation can be shown to be exactly equivalent to an maximum a posteriori (MAP) probabilistic estimator if observations are assumed to be corrupted with additive Gaussian noise. To prove this, we start from the formulation of a MAP estimator:

$$x^\star = \arg\max_{\mathbf{x}} p(z|x) = \arg\max_{\mathbf{x}} \log(p(z|x)), \tag{7}$$

where, for simplicity of notation, we used $z = \{z_1, \ldots, z_N\}$ and x to refer to the set of N observed points and the vehicle pose for an arbitrary time step of interest, and we took logarithms (a monotonic function that does not change the found optimal value) for convenience in further derivations. Assuming the following generative model for observations:

$$z_i \sim \bar{\mathbf{m}}^i + \mathcal{N}(\mathbf{0}, \Sigma^i), \tag{8}$$

$$\bar{\mathbf{m}}^i = \mathbf{p}^i_{map} \ominus \mathbf{x}, \tag{9}$$

$$\Sigma^i = diag(\sigma^2, \sigma^2, \sigma^2), \tag{10}$$

where $p \ominus x$ means the local coordinates of point p as seen from the frame of reference x, $\mathcal{N}(\mathbf{m}, \Sigma)$ is the multivariate Gaussian distribution with mean $\mathbf{m}$ and variance Σ, and σ is the standard deviation of the assumed additive Gaussian error in measured points. Then, replacing Equation (8) into Equation (7), using the known exponential formula for the Gaussian distribution, we find:

$$
\mathbf{x}^{\star} = \arg\max_{\mathbf{x}} \{\log(p(z|\mathbf{x}))\} \tag{11}
$$

$$
= \arg\max_{\mathbf{x}} \left\{ \log \left(\prod_{i=1}^{N} p(z_i|\mathbf{x}) \right) \right\} \quad \text{(Statistical independence of } N \text{ observations)} \tag{12}
$$

$$
= \arg\max_{\mathbf{x}} \sum_{i=1}^{N} \{\log(p(z_i|\mathbf{x}))\} \tag{13}
$$

$$
= \arg\max_{\mathbf{x}} \sum_{i=1}^{N} \log \left(\left((2\pi)^3 |\Sigma| \right)^{-1/2} \exp\left(-\frac{1}{2}(z_i - \bar{\mathbf{m}}^i)^\top \Sigma (z_i - \bar{\mathbf{m}}^i) \right) \right) \tag{14}
$$

$$
= \arg\max_{\mathbf{x}} \sum_{i=1}^{N} \underbrace{\log \left(\left((2\pi)^3 |\Sigma| \right)^{-1/2} \right)}_{\text{Does not depend on } \mathbf{x}} + \log \left(\exp\left(-\frac{1}{2}(z_i - \bar{\mathbf{m}}^i)^\top \Sigma (z_i - \bar{\mathbf{m}}^i) \right) \right) \tag{15}
$$

$$
= \arg\max_{\mathbf{x}} \sum_{i=1}^{N} -\frac{1}{2}(z_i - \bar{\mathbf{m}}^i)^\top \Sigma (z_i - \bar{\mathbf{m}}^i) \tag{16}
$$

$$
= \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \underbrace{\frac{1}{2}}_{\text{Constant}} (z_i - \bar{\mathbf{m}}^i)^\top \Sigma (z_i - \bar{\mathbf{m}}^i) \tag{17}
$$

$$
= \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \left\| \frac{z_i - \bar{\mathbf{m}}^i}{\sigma} \right\|^2 \quad \text{(Since } \Sigma \text{ is diagonal).} \tag{18}
$$

Identifying the last line above with Equation (3), it is clear that the MAP statistical estimator is identical to a least squares problem with error terms $c_i = \frac{z_i - \bar{\mathbf{m}}^i}{\sigma}$. By using a truncated Gaussian in Equation (8), i.e., by modeling outliers as having a uniform probability density, one can also show that the corresponding MAP estimator becomes the robust least-squares problem in Equation (5).

Therefore, the proposed observation likelihood function in Algorithm 1 enables an estimator to find the most likely pose of a vehicle while being robust to outlier observations, for example, from dynamic obstacles.

4. Justification of Decimation as an Approximation to the Likelihood Function

A key feature of the proposed likelihood model in Algorithm 1, and which is being benchmarked in this work, is the decimation ratio, that is, how many points from each observed scan are actually considered, with the rest being plainly ignored.

The intuition behind this simple approach is that information in point clouds is highly redundant, such that, by using only a fraction of the points, one could save a significant computational cost while still achieving good vehicle localization. From the statistical point of view, justifying the decimation is only possible if the resulting likelihood functions (which in turn are probability density functions, p.d.f.) are

still similar. From Equations (11)–(18) above, solving the decimated problem is finding optimal pose $\hat{x}^\star$ to the approximated p.d.f. with decimation ratio D:

$$\hat{x}^\star = \arg\min_{\mathbf{x}} \sum_{i=1,D,2D,\ldots} \left\| \frac{z_i - \bar{\mathbf{m}}^i}{\sigma} \right\|^2. \tag{19}$$

Numerically, the decimated and the original p.d.f. are clearly not identical, but this is not an issue for we are mostly interested in the location of the global optimum and the shape of the cost function in its neighborhood. The sum of convex functions is convex. In our case, we have truncated square cost functions (recall Figure 2b), but the overall p.d.f. will still be convex near the true vehicle pose. Note that, since associations between observed and map points are determined based solely on pairwise nearness, in practice, the observation likelihood is not convex when evaluated far from the real vehicle pose. However, this fact can be exploited by the particular kind of estimator used in this work (particle filters) to obtain multi-modal pose estimations, where localization hypotheses are spread among several candidate "spots"—for example, during global relocalization, as will be shown experimentally.

As a motivational example, we propose measuring the similarity between the decimated $\hat{p}(z|x_i)$ and the original p.d.f. $p(z|x_i)$ using the Kullback–Leibler divergence (KLD) [32] using the following experimental procedure. Given a portion of the reference point cloud map, and a scan observation from a Velodyne VLP-16, we have numerically evaluated both the original and the decimated likelihood function in the neighborhood of the known ground-truth solution for the vehicle pose. In particular, we evaluated the functions in a 6D grid (since SE(3) poses have six degrees of freedom) within an area of ± 3 m for translation in (x, y, z), $\pm 7.5°$ for yaw (azimuth), and $\pm 3°$ for pitch and roll, with spatial and angular resolutions of 0.15 m and 5°, respectively. For such discretized model of likelihood functions, we applied the discrete version of KLD that is:

$$KLD(p||\hat{p}) = -\sum_i p(z|x_i) \log \frac{\hat{p}(z|x_i)}{p(z|x_i)}, \tag{20}$$

which has been summed for the 3.1×10^6 grid cells around the ground truth pose, for a set of decimation ratio values. The result KLD is shown in Figure 3, and some example planar slices of the corresponding likelihood functions are illustrated in Figure 4. Decimated versions are clearly quite similar to the original one up to decimation ratios of roughly ~ 500, which closely coincides with statistical localization errors presented later on in Section 6.
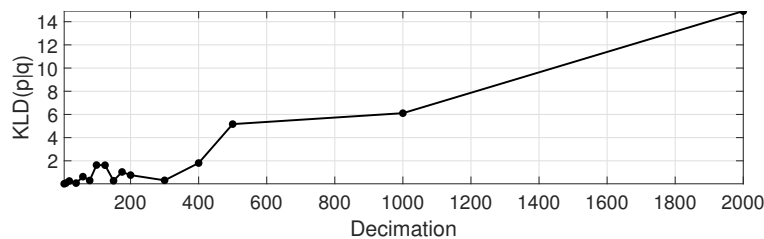
Figure 3. Kullback–Leibler divergence (KLD) between the original and decimated version of the proposed likelihood model for point cloud observations. Note that the decimated versions are remarkably similar to the original one up to decimation ratios of roughly ~ 500.
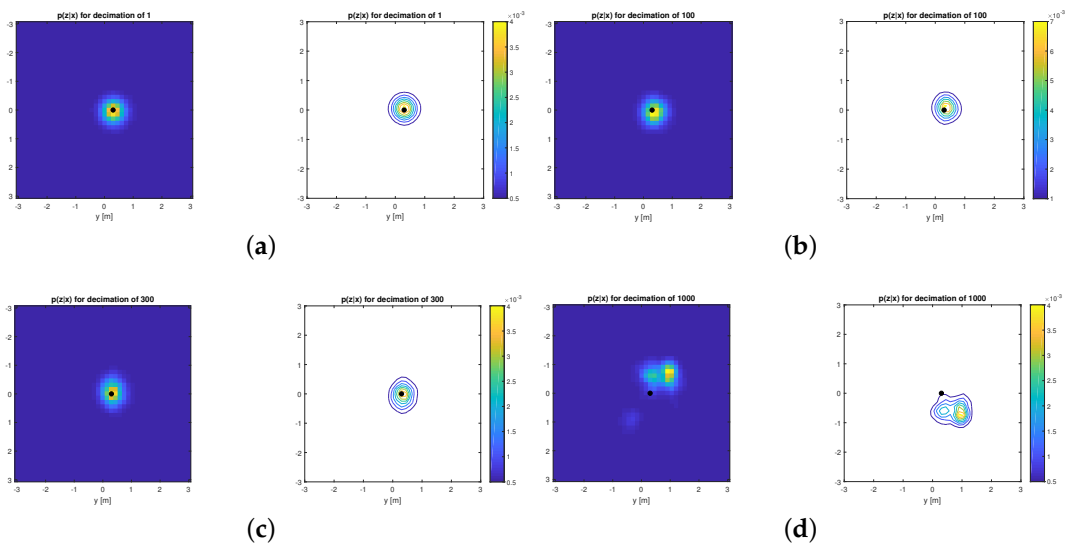
Figure 4. Some planar slices of 6D likelihood functions evaluated for different decimation ratios, as both an intensity color plot and a contour diagram. Ground-truth vehicle pose is marked as a large block dot in all the figures. (**a**) Original p.d.f. $p(z|x)$; (**b**) Approximate $\hat{p}(z|x)$ for D = 100; (**c**) Approximate $\hat{p}(z|x)$ for D = 300; (**d**) Approximate $\hat{p}(z|x)$ for D = 1000.

5. Experimental Platform

In order to perform the experimental test of this work, a customized urban electric vehicle has been used (Figure 5). Among the sensors and actuators that have been installed in the vehicle are a steering-by-wire system, which comprises a DC motor (Maxon RE50, Maxon Motor AG, Sachseln, Switzerland, diameter 50 mm, graphite brushes and 200 Watts) coupled to the conventional steering mechanism, commanded by a Pulse width modulation (PWM) signal and two encoders to close the control loop. The main feedback sensor is an incremental encoder HEDL5540 with a resolution of 500 pulses per revolution, and a redundant angle measurement is performed by an absolute encoder EMS22A with 10 bits of resolution. Another couple of encoders are mounted in the rear wheels to serve as odometry. Finally, the prototype is equipped with a Novatel SPAN ING GNSS solution and a Velodyne VLP-16 3D LiDAR. More details about both mechanical characteristics and sensors placement are provided in Appendix A.

Figure 5. The autonomous vehicle prototype employed in this work.

The software architecture runs on top of a PC (64 bit Ubuntu GNU/Linux) and under Robotics Operative System (ROS) [33] (version Kinetic), with additional applications from the Mobile Robot Programming Toolkit (MRPT) (https://www.mrpt.org/), version 1.9.9.

We provide open-source C++ implementations for all the modules employed in this work. Sensor acquisition for Novatel GNSS and Velodyne scanners were implemented as C++ classes in the MRPT project. We also plan to release ROS wrappers in the repository `mrpt_sensors` (https://github.com/mrpt-ros-pkg/mrpt_sensors). Our implementation supports programmatically changing all Velodyne parameters (e.g., rpm), reading in dual range mode, etc. Odometry and low-level control are implemented in an independent (https://github.com/ual-arm-ros-pkg/ual-ecar-ros-pkg) ROS repository. Particle filter algorithms are also part of the MRPT libraries and have ROS wrappers in `mrpt_navigation` (https://github.com/mrpt-ros-pkg/mrpt_navigation).

6. Results and Discussion

Next, we discuss the results for each individual experiment and benchmark. All experiments ran within a single-thread on an Intel i5-4590 CPU, (Intel Corporation, Santa Clara, CA, USA) @ 3.30 GHz. Due to the stochastic nature of PF algorithms, statistical results are presented for all benchmarks, which have been evaluated a number of times feeding the pseudorandom number generators with different seeds.

6.1. Mapping

We acquired a dataset in the UAL campus (see Figure 6) with the purpose of serving to build a reference metric map and also to benchmark PF-based localization algorithms. As described in former sections, we used centimeter-accurate GPS positioning and Novatel SPAN INS attitude estimation for orientation angles. Poses were recorded at 20 Hz along the path shown in Figure 6b. Since time has been represented in the vertical axis, it is easy to see how the vehicle was driven through the same areas several times during the dataset. In particular, we manually selected a first fragment of this dataset to generate a metric map (segment A–B in Figure 6b), then a second non-overlapping fragment (segment C–D) to test the localization algorithms as discussed in the following. The global map obtained for the entire dataset is depicted in Figure 6a, whereas the corresponding ground-truth path can be seen in Figure 7.
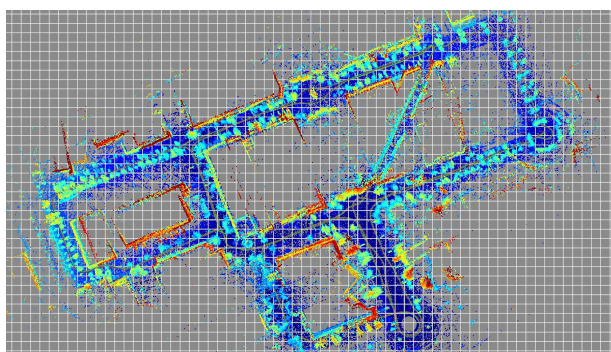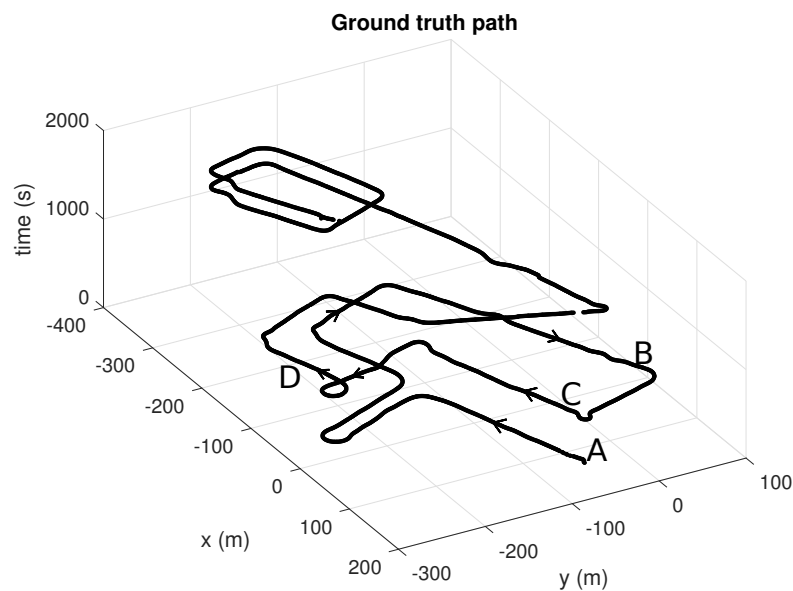

(**a**)

Figure 6. *Cont.*

(**b**)

Figure 6. Bird-eye view of (**a**) the ground-truth map used in the benchmark, along with (**b**) a corresponding satellite image of the UAL campus.

Figure 7. The whole ground-truth path of the presented dataset, with time used as vertical axis to help identifying the loops. Please compare with Figure 6 for reference.

6.2. Relocalization, Part 1: Aided by Poor-Signal GPS

Global localization, the "awakening problem" or "relocalization" are all names of specific instances of the localization problem: those in which uncertainty is orders of magnitude larger than during regular operation. Depending on the case and available sensors, uncertainty may span a few square meters within one room, or an entire city-scale area. Since our work addresses localization in outdoor environments, we will assume that a consumer-grade, low-cost GPS device is available during the initialization of the localization system. To benchmark such a situation, we initialize the PF with different number of particles (ranging from 20 to 4000) spread over an area of 30×30 m^2 that includesthe actual vehicle pose. No clue is given for orientation (despite the fact that it might be easy to obtain from low-cost magnetic sensors) and no GPS measurements are used in subsequent steps of the PF, whose only inputs are Velodyne scans and

odometry readings. The size of this area has been chosen to cover a typical worst-case GPS positioning data with poor precision, that is, with a large dilution of precision (DOP). Such a situation is typically found in areas where direct sight of satellites is blocked by obstacles (e.g., trees, buildings). Refer to [34] for an experimental measurement of such GPS positioning errors.

Notice that the particle density is small even for the largest case (N = 4000, density is $4000/900 \approx 4.4$ particles/m^2), but the choice of the optimal-sampling PF algorithm makes it possible to successfully converge to the correct vehicle pose within a few timesteps.

We investigated what is the minimum particle density required to ensure a high probability of converging to the correct pose, since oversizing might lead to excessive delays while the system waits for convergence. Relocalization success was assessed by running a PF during 100 timesteps and checking whether (i) the average particle pose is close to the actual (known) ground truth solution (closer than two meters), and (ii) the determinant of the covariance fitting all particles is below a threshold ($|\Sigma| < 2$). Together, these conditions are a robust indicator of whether convergence was successful. The experiment was run 100 times for each initial population size N, using a point cloud decimation of 100, and automatic sample size was in effect in the second and subsequent time steps. The success ratio results can be seen in Figure 8a, and demonstrate that the optimal-sampling PF requires, in our dataset, a minimum of 4000 particles (4.4 particles/m^2) to ensure convergence. Obviously, the computational cost grows with N as Figure 8b shows, hence the interest in finding the minimum feasible population size. Note that the computational cost is not linear with N due to the complex evolution of the actual population size during subsequent timesteps. Normalized statistics regarding number of initial particles per area are also provided in Table 1, where it becomes clear that an initial density of ∼2 particles/m^2 seems to be the minimum required to ensure convergence for the proposed model of observation likelihood.
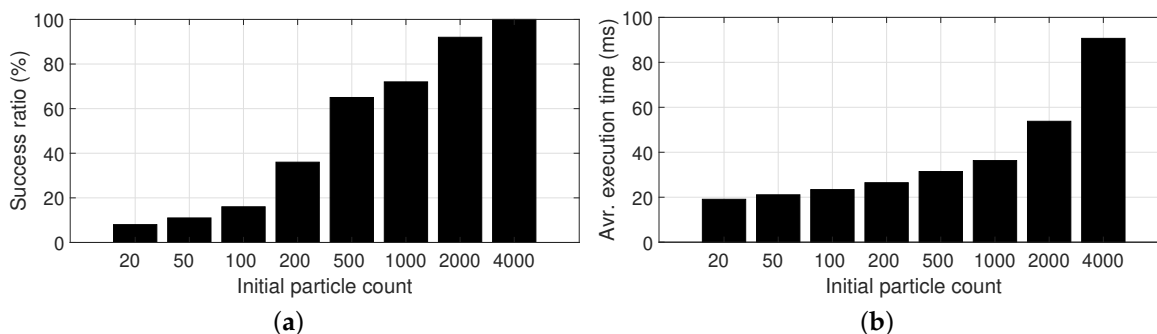
Figure 8. Statistical results for the relocalization benchmark. Refer to Section 6.2 in the text for further details. (**a**) Success ratio of relocalization; (**b**) Computation cost.

6.3. Relocalization, Part 2: LiDAR Only

Next, we analyze the performance of the particle filter algorithm to localize, from scratch, our vehicle without any previous hint about its approximate pose within the map of the entire campus.

For that, we draw N random particles following a uniform distribution (in x, y, and also in the vehicle azimuth ϕ) as the initial distribution, with different values of N, and after 100 time steps, we detect whether the filter has converged to a single spot, and whether the average estimated pose is actually close to the ground truth pose. The experiment has been repeated 150 times for each initial particle count N. The area where particles are initialized has a size of $420 \times 320 \, \text{m}^2 = 134,400 \, \text{m}^2$. Notice that the dynamic sample size algorithm ensures that computational cost quickly decreases as the filter converges, hence the higher computational cost associated with a larger number of particles only affects the first iterations (typically, less than 10 iterations).

Table 1. Statistical results for the relocalization benchmark with an initial uncertainty area of 30×30 m^2. Refer to Section 6.2.

Initial Density (Particles/m^2)	Convergence Ratio
0.02	21.7%
0.08	50.0%
0.17	68.3%
0.33	80.0%
0.83	96.7%
1.67	100%
3.33	100%
6.67	100%

The summary of results can be found in Table 2, and are consistent with the relationship between initial particle densities and convergence success ratio in Table 1. A video for a representative run of this test is available online for the convenience of the reader (Video available in: https://www.youtube.com/watch?v=LJ5OV-KMQLA).

Table 2. Statistical results for the relocalization benchmark with an initial uncertainty area of the entire campus. Refer to Section 6.3.

Initial Particle Count	Initial Density (Particles/m^2)	Convergence Ratio
1000	0.007	2.0%
2000	0.014	12.0%
5000	0.037	24.0%
10,000	0.074	32.0%
20,000	0.149	56.6%
30,000	0.223	63.3%
40,000	0.298	66.6%
50,000	0.373	75.3%
60,000	0.447	80%
70,000	0.522	81.3%
80,000	0.597	86.0%
100,000	0.746	89.3%
125,000	0.932	89.3%
150,000	1.119	92.6%
175,000	1.305	92.6%
200,000	1.492	96.0%

6.4. Choice of PF Algorithm

In this benchmark, we analyzed the pose tracking accuracy (positioning error with respect to ground-truth) and efficiency (average computational cost per timestep) of a PF using the standard proposal distribution in contrast to another using the optimal proposal. Please refer to [13] for details on how this algorithm achieves a better random sampling of the target probability distribution, by simultaneously taking into account both the odometry model and the observation likelihood $p(z_t|x_t, m)$ in Algorithm 1.

Experiments were run 25 times and average errors and execution times were collected for each algorithm, then data fitted as a 2D Gaussian as represented in Figure 9. The minimum population size of the standard PF was set to 200, while it was 10 for the optimal PF. However, their "effective" number of particles are equivalent since each particle in the optimal algorithm was set to employ 20 terations in the internal sampling-based stage. The optimal PF achieves a slightly better accuracy with a relatively higher computational cost, which still falls below 20 ms per iteration. Therefore, the conclusion is that the optimal

algorithm is recommended, but with a small practical gain, a finding in accordance with previous works that revealed that the advantages of the optimal PF become more patent when applied to SLAM, while only representing a substantial improvement for localization when the sensor likelihood model is sharper [13].
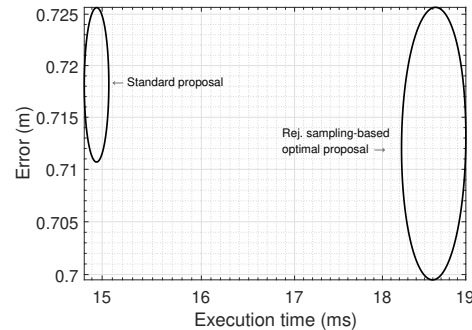
Figure 9. Execution time and average pose tracking error for two different PF algorithms. Ellipses represent 95% confidence interval as reconstructed from data of 25 repetitions of the same pose tracking experiment with different random seeds. Point cloud decimation was set to 100 in both algorithms. See Section 6.4 in the text for a discussion.

6.5. Tracking Performance

To demonstrate the suitability of the proposed observation model, we run 10 instances of a pose tracking PF using the standard proposal distribution, point cloud decimation of 100, and a dynamic number of samples with a minimum of 100. We evaluated the mean and 95% confidence intervals for the localization error over the vehicle path, and compared it to the error that would accumulate from odometry alone in Figure 10. As can be seen, the PF keeps track of the actual vehicle pose with a error median of 0.6 m (refer to Table 3).
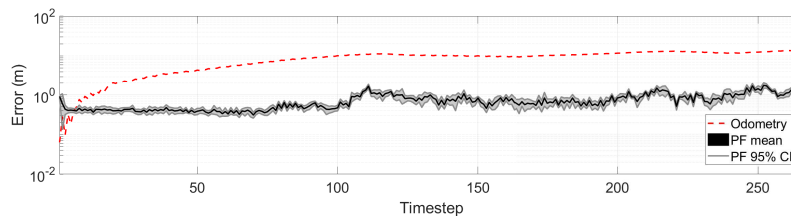
Figure 10. Pose tracking error and odometry-only error. See Section 6.5 in the text for more details.

*6.6. Decimating Likelihood Evaluations*

Finally, we addressed the issue of how much information can be discarded from each incoming scan while preventing the growth of positioning error. Decimation is the single most crucial parameter regarding the computational cost of pose tracking with PF, hence the importance of quantitatively evaluating its range of optimal values. The results, depicted in Figure 11, clearly show that decimation values in the range 100 to 200 should be the minimum choice since error is virtually unaffected. In other words, Velodyne scans apparently have so much redundant information that we can keep only 0.5% of them and still remain well-localized. Statistical results of these experiments, and the corresponding error histograms, are shown in Table 3 and Figure 12, respectively. As can be seen from the results, the average error is relatively stable for decimation values of up to 500, and quickly grows afterwards.
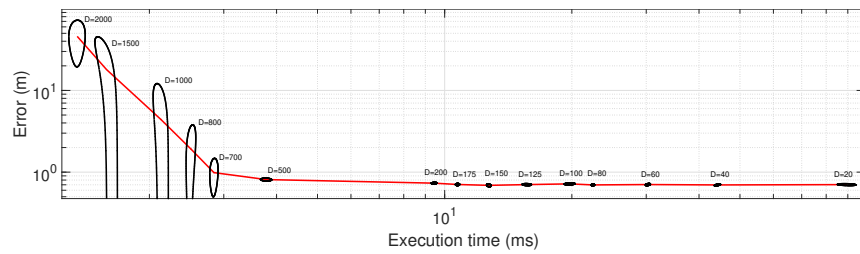
Figure 11. Positioning error and computational cost per timestep for different values of the likelihood function decimation parameter. Black: 95% confidence intervals (ellipses deformed due to the logarithmic scale) for 25 experiments, red: mean values. Note that the initial error and uncertainty are larger than during the steady state of the tracking algorithm, since particles are initially uniformly-distributed over an area of 30×30 m^2. See Section 6.6 in the text for more details.

Table 3. Localization error statistics for different decimation ratios applied to the input sensory data.

Decim.	Mean (m)	Median (m)	Standard Deviation (m)
20	0.615	0.585	0.250
40	0.620	0.585	0.257
60	0.626	0.590	0.261
80	0.635	0.579	0.282
100	0.635	0.587	0.279
150	0.641	0.591	0.290
200	0.656	0.586	0.315
300	1.207	0.623	3.528
500	0.951	0.641	1.182
700	8.356	0.746	20.023
800	6.497	0.765	17.832
1000	8.915	0.892	17.405
1500	21.674	4.325	30.847
2000	30.895	9.638	37.628

(a)

(b)

(c)

(d)

(e)

(f)
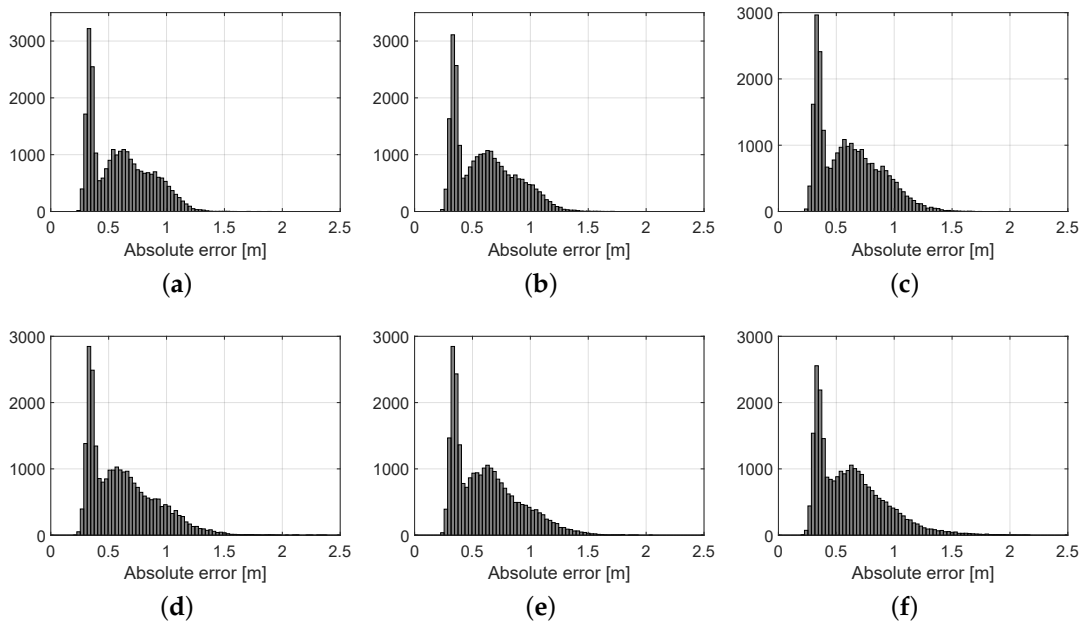
Figure 12. *Cont.*
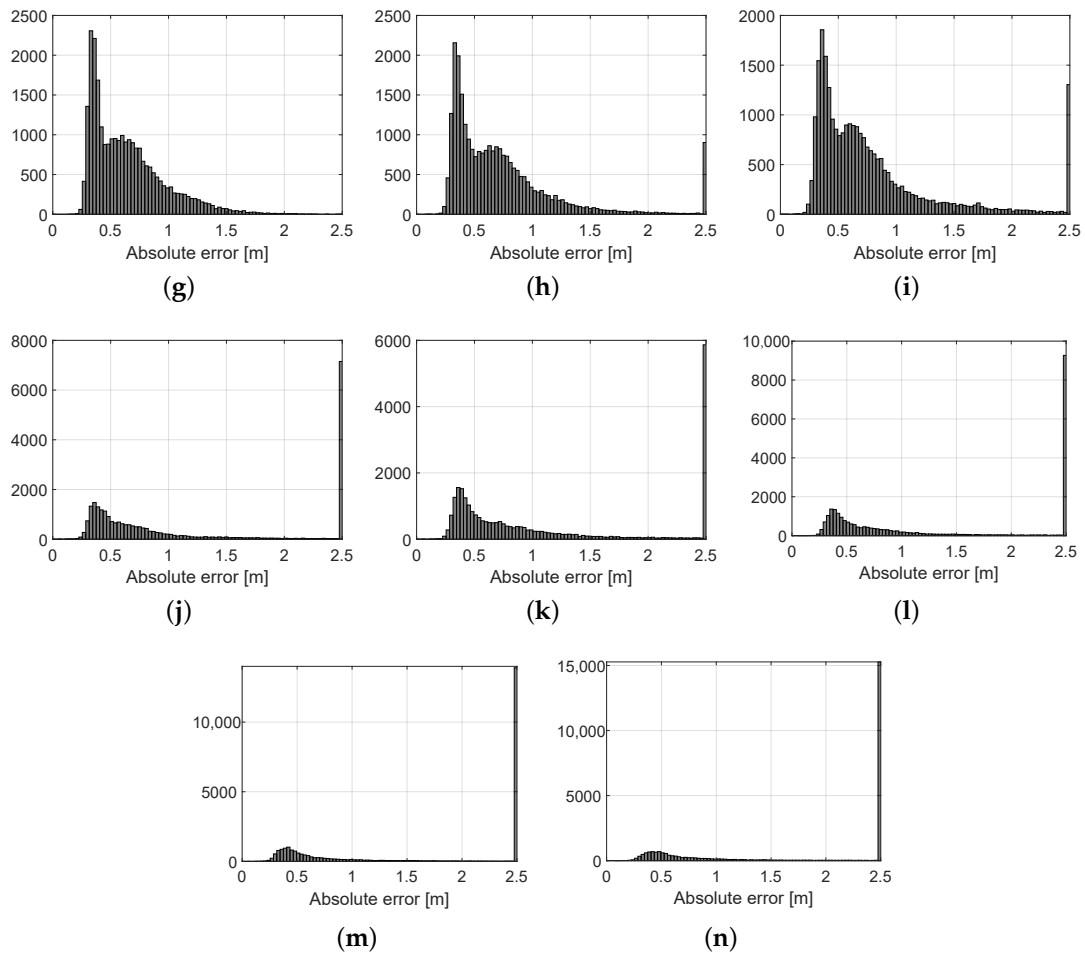
Figure 12. Histograms of the pose tracking localization error for different decimation ratios. Note that errors larger than 2.5 m are clipped into one single bin for the largest decimation ratios, for the sake of providing a uniform horizontal scale in all plots. (**a**) Decimation = 20; (**b**) Decimation = 40; (**c**) Decimation = 60; (**d**) Decimation = 80; (**e**) Decimation = 100; (**f**) Decimation = 150; (**g**) Decimation = 200; (**h**) Decimation = 300; (**i**) Decimation = 500; (**j**) Decimation = 700; (**k**) Decimation = 800; (**l**) Decimation = 1000; (**m**) Decimation = 1500; (**n**) Decimation = 2000.

7. Conclusions

In this work, we proposed an observation model for Velodyne scans, suitable for use within a PF, which has been successfully validated experimentally. Benchmarks showed that the optimal-PF algorithm is preferable in general due to its superior accuracy during pose tracking and its suitability to cope with the relocalization problem with an exiguous density of particles. Furthermore, one of the most remarkable results is the finding that PFs are robust enough to keep track of a vehicle pose while decimating the input point cloud from a Velodyne sensor by factors of two orders of magnitude. Such an insight, together with the use of a KD-tree for efficient querying the reference map, allows for running an entire localization update step within 10 to 20 ms.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Vehicle Description and Raw Dataset

Table A1. Main characteristics of the prototype vehicle.

Mechanic Characteristics	Value
Lenght $\times$ Width $\times$ Height	$2680 \times 1525 \times 1780$ mm^3
Wheelbase	1830 mm
Front/rear track width	1285/1260 mm
Weight without/with batteries	472/700 kg

Electric Characteristics	Value
DC motor XQ $-$ 4.3	4.3 kW
Batteries (gel technology)	8×6 V -210 Ah
Autonomy	90 km
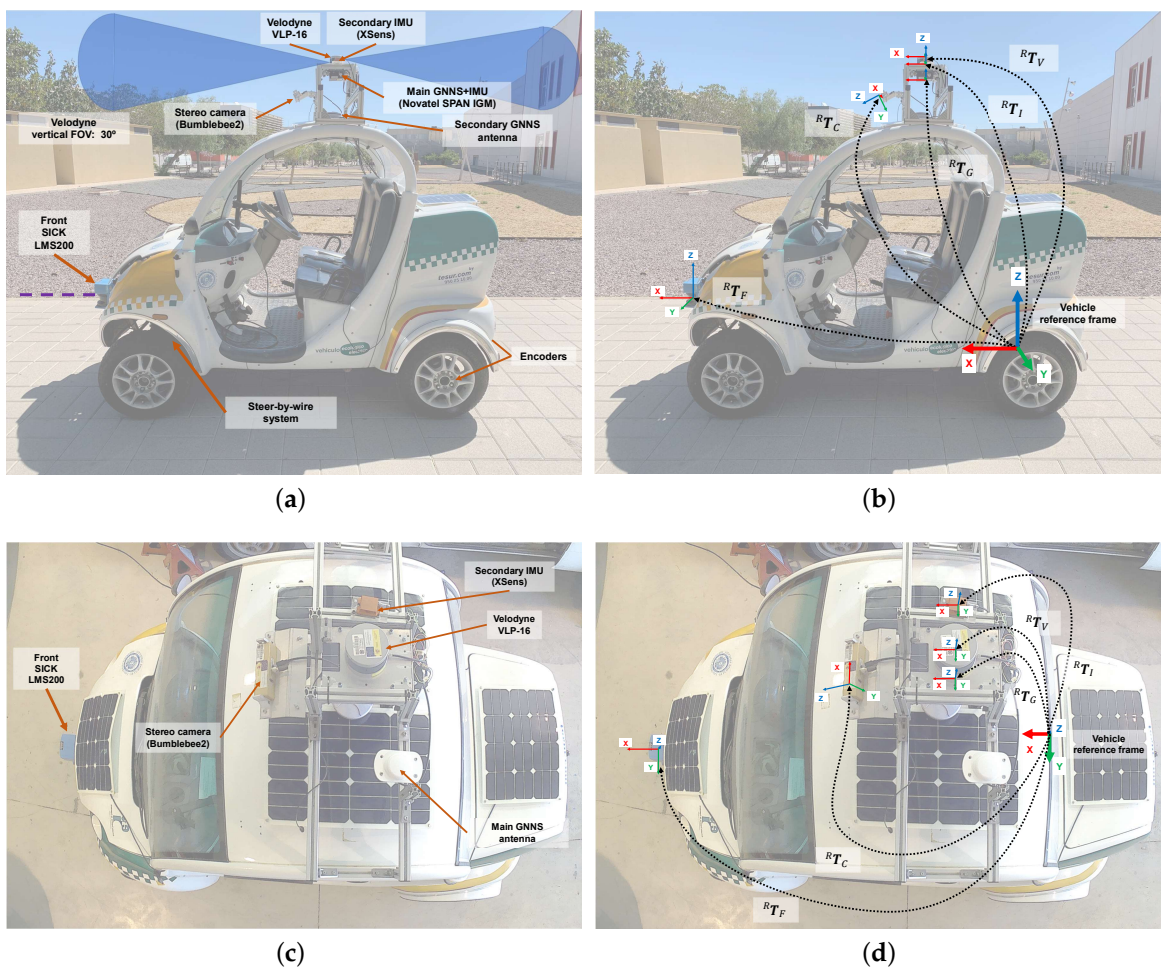
Figure A1. Electric vehicle prototype used in the experimental tests. (**a**) Side view: sensors; (**b**) Side view: frames; (**c**) Top view: sensors; (**d**) Top view: frames.

The main characteristics of the experimental vehicle used are summarized in Table A1. A pack of eight batteries Trojan TE35-Gel 210Ah 6V propels the vehicle ensuring an autonomy of 90 km at a maximum travel speed of 45 km/h by means of a 48 V DC motor controlled by a permanent magnet motor. Speed is controlled by a Curtis PMC controller (model 1268-5403). Three voltmeters are employed to measure the voltage in the rotor, the field, and the batteries. In addition, the prototype is equipped with three ampere-meters (LEM DHR 100, LEM, Fribourg, Switzerland) to measure instantaneous current consumption, at the same three elements.

Figure A1 shows all the installed sensors, together with their relative poses with respect to the vehicle frame of reference. Approximate values for each such poses, together with the raw dataset, are available online (https://ingmec.ual.es/datasets/lidar3d-pf-benchmark/).

References

1. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]

2. Blanco-Claraco, J.L.; Moreno-Dueñas, F.Á.; González-Jiménez, J. The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. *Int. J. Robot. Res.* **2014**, *33*, 207–214. [CrossRef]

3. Gaspar, A.R.; Nunes, A.; Pinto, A.M.; Matos, A. Urban@CRAS dataset: Benchmarking of visual odometry and SLAM techniques. *Robot. Auton. Syst.* **2018**, *109*, 59–67. [CrossRef]

4. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. g^{2}o: A general framework for graph optimization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 3607–3613.

5. Ila, V.; Polok, L.; Solony, M.; Svoboda, P. SLAM++—A highly efficient and temporally scalable incremental SLAM framework. *Int. J. Robot. Res.* **2017**, *36*, 210–230. [CrossRef]

6. Kaess, M.; Johannsson, H.; Roberts, R.; Ila, V.; Leonard, J.J.; Dellaert, F. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Int. J. Robot. Res.* **2012**, *31*, 216–235. [CrossRef]

7. Blanco, J.L. A Modular Optimization Framework for Localization and Mapping. In Proceedings of the Robotics: Science and Systems, Freiburg im Breisgau, Germany, 22–26 June 2019.

8. Yoneda, K.; Tehrani, H.; Ogawa, T.; Hukuyama, N.; Mita, S. Lidar scan feature for localization with highly precise 3D map. In Proceedings of the IEEE Intelligent Vehicles Symposium, Ypsilanti, MI, USA, 8–11 June 2014; pp. 1345–1350.

9. Moosmann, F.; Stiller, C. Velodyne SLAM. In Proceedings of the IEEE Intelligent Vehicles Symposium, Baden-Baden, Germany, 5–9 June 2011; pp. 393–398.

10. Dubé, R.; Dugas, D.; Stumm, E.; Nieto, J.; Siegwart, R.; Cadena, C. Segmatch: Segment based place recognition in 3D point clouds. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5266–5272.

11. Levinson, J.; Montemerlo, M.; Thrun, S. Map-Based Precision Vehicle Localization in Urban Environments. In Proceedings of the Robotics: Science and Systems, Atlanta, GA, USA, 27–30 June 2007; Volume 4, p. 1.

12. Rabe, J.; Stiller, C. Robust particle filter for lane-precise localization. In Proceedings of the 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Vienna, Austria, 27–28 June 2017; pp. 127–132.

13. Blanco, J.L.; González-Jiménez, J.; Fernández-Madrigal, J.A. Optimal Filtering for Non-Parametric Observation Models: Applications to Localization and SLAM. *Int. J. Robot. Res.* **2010**, *29*. [CrossRef]

14. Song, W.; Yang, Y.; Fu, M.; Kornhauser, A.; Wang, M. Critical Rays Self-Adaptive Particle Filtering SLAM. *J. Intell. Robot. Syst. Theory Appl.* **2018**, *92*, 107–124. [CrossRef]

15. Maalej, Y.; Sorour, S.; Abdel-Rahim, A.; Guizani, M. Vanets Meet Autonomous Vehicles: Multimodal Surrounding Recognition Using Manifold Alignment. *IEEE Access* **2018**, *6*, 29026–29040. [CrossRef]

16. Xu, P.; Dherbomez, G.; Hery, E.; Abidli, A.; Bonnifait, P. System Architecture of a Driverless Electric Car in the Grand Cooperative Driving Challenge. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 47–59. [CrossRef]

17. Choi, J. Hybrid Map-Based SLAM Using a Velodyne Laser Scanner. In Proceedings of the 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 3082–3087.

18. Jo, H.; Cho, H.M.; Jo, S.; Kim, E. Efficient Grid-Based Rao-Blackwellized Particle Filter SLAM with Interparticle Map Sharing. *IEEE/ASME Trans. Mechatron.* **2018**, *23*, 714–724. [CrossRef]

19. Vallicrosa, G.; Ridao, P.; Vallicrosa, G.; Ridao, P. H-SLAM: Rao-Blackwellized Particle Filter SLAM Using Hilbert Maps. *Sensors* **2018**, *18*, 1386. [CrossRef]

20. Wen, S.; Chen, J.; Lv, X.; Tong, Y. Cooperative simultaneous localization and mapping algorithm based on distributed particle filter. *Int. J. Adv. Robot. Syst.* **2019**, *16*. [CrossRef]

21. Grant, W.S.; Voorhies, R.C.; Itti, L. Efficient Velodyne SLAM with point and plane features. *Auton. Robots* **2019**, *43*, 1207–1224. [CrossRef]

22. Doucet, A.; De Freitas, N.; Gordon, N. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 3–14.

23. Arulampalam, M.S.; Maskell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]

24. Blanco-Claraco, J.L. Contributions to Localization, Mapping and Navigation in Mobile Robotics. Ph.D. Thesis, Universidad de Malaga, Malaga, Spain, 2009.

25. Doucet, A.; Godsill, S.; Andrieu, C. On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.* **2000**, *10*, 197–208. [CrossRef]

26. Fox, D. Adapting the sample size in particle filters through KLD-sampling. *Int. J. Robot. Res.* **2003**, *22*, 985–1003. [CrossRef]

27. Huber, P.J. Robust estimation of a location parameter. In *Breakthroughs in Statistics*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 492–518.

28. Audibert, J.Y.; Catoni, O. Robust linear least squares regression. *Ann. Stat.* **2011**, *39*, 2766–2794. [CrossRef]

29. Yang, H.; Carlone, L. A Polynomial-time Solution for Robust Registration with Extreme Outlier Rates. In Proceedings of the Robotics: Science and Systems, Freiburg im Breisgau, Germany, 22–26 June 2019. [CrossRef]

30. Ruppert, D.; Carroll, R.J. Trimmed least squares estimation in the linear model. *J. Am. Stat. Assoc.* **1980**, *75*, 828–838. [CrossRef]

31. Chen, J.H. M-estimator based robust kernels for support vector machines. In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; Volume 1, pp. 168–171.

32. MacKay, D.J.; Mac Kay, D.J. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.

33. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-Source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.

34. Dutt, V.S.I.; Rao, G.S.B.; Rani, S.S.; Babu, S.R.; Goswami, R.; Kumari, C.U. Investigation of GDOP for Precise User Position Computation with All Satellites in View and Optimum Four Satellite Configurations. *J. Ind. Geophys. Union* **2009**, *13*, 139–148.