IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 January 2023, accepted 12 February 2023, date of publication 22 February 2023, date of current version 28 February 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3247503*

# Prior Distribution Refinement for Reference Trajectory Estimation With the Monte Carlo-Based Localization Algorithm

**MARK GRIGULETSKII[1,3], OLEG SHIPITKO[1,2], MAXIM ABRAMOV[1,2], EGOR PRISTANSKIY[1,4], VLADISLAV KIBALOV[1,2], AND ANTON GRIGORYEV[1,2]**

[1]Evocargo LLC, 129085 Moscow, Russia
[2]Institute for Information Transmission Problems (IITP), RAS, 127051 Moscow, Russia
[3]Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[4]Department of Applied Mathematics and Computer Science, Moscow Polytechnic University, 121205 Moscow, Russia

Corresponding author: Mark Griguletskii (mark.griguletskii@skoltech.ru)

This work was supported by the EVOCARGO LLC.

**ABSTRACT** A robot localization problem demands a fair comparison of the positioning algorithms. A reference trajectory of the robot's movement is needed to estimate errors and evaluate a quality of the localization. In this article, we propose the Prior Distribution Refinement method for generating a reference trajectory of a mobile robot with the Monte Carlo-based localization system. The proposed approach can be applied for both indoor and outdoor environments of an arbitrary size without the need for expensive position tracking sensors or intervention in the testing infrastructure. The reference trajectory is generated by running the algorithm over a so-called Particles' Transition Graph, obtained from a resampling stage of Monte Carlo localization. The prior distribution of particles is then refined by forward-backward propagation through the graph and exploring the connections between particles. The Viterbi algorithm is applied afterwards to generate a reference trajectory based on refined particles' distribution. We demonstrate that such an approach is capable of generating accurate estimates of a mobile robot's position and orientation with the only requirement of moderate quality of localization system being used as a core algorithm for iterative optimization.

**INDEX TERMS** Ground truth trajectory, reference trajectory, benchmark trajectory, monte carlo localization, particle filter, robot tracking, smoothing, localization.

## I. INTRODUCTION

To obtain a meaningful, truthful result of the research or production development, it is necessary to be able to compare the result of the work with some standard. The benchmarks, performance metrics, and datasets allow us to compare various methods, track the progress towards the goal, and verify algorithms against state-of-the-art results in the research field. This is especially true for robotics, where there is a large quantity of factors influencing any experiment. The variety of robot designs, operating environments, sensors, and actuators make it almost infeasible to compare systems in terms of performance.

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

Localization is the key component of many autonomous robotics system. The large number of factors makes measuring the performance of robot navigation systems complicated. There are many performance metrics for localization, and it is not obvious which ones should be chosen. Additionally, there are many possible environmental conditions: indoor and outdoor in the case of mobile robots, highways and urban conditions in the case of autonomous vehicles, and specific environments for logistic robots. Finally, there is no inexpensive, effortless method for measuring a robot reference trajectory. This article mainly focuses on the last problem. Henceforth, by ground truth trajectory, we mean the trajectory undertaken by a robot. By reference trajectory, we mean an estimate of the ground truth trajectory which can be used to evaluate the quality of the robot's localization.

Table 1 summarizes approaches used to obtain the reference trajectory for a robot. It can be seen that there is no methodology for testing navigation systems that suits both indoor and outdoor scenarios and does not require additional sensors and preliminary preparation of the testing area.

In this work, we propose a computational method for reference trajectory estimation for algorithms based on Monte Carlo localization [1] paradigm. The proposed method uses Particles' Transition Graph – the graph where each vertex is a particle (i.e. hypothesis of robot pose in a particular time moment) and the edges represent the resampling process: stochastic relocations of particles based on likelihood. The main hypothesis of our method is that particles that ''survived'' in the last time moment were sampled from the ones that had a high probability. If we track particles back through the Particles' Transition Graph until the first moment and distribute new particles around the ancestors of survived ones (*prior distribution*) in the subsequent run, we would improve the localization quality.

The main contributions of this paper are as follows:

- We propose a novel method for reference trajectory estimation based on iterative refinement of the prior distribution of the robot's initial state.
- We evaluate the proposed method both on simulated and real data. For this, we use a visual localization algorithm based on road-markings detection implemented in autonomous cargo vehicles. We also compare the proposed method with the Viterbi algorithm applied to reference trajectory estimation.

## II. RELATED WORK

How should the performance of two localization algorithms be compared? How to do it if the systems use different sensors to extract environmental features? An obvious approach is to build robots and run them in the same environment. That is the idea behind robotics competitions such as RoboCup [2], DARPA Grand Challenge [3], Indy Autonomous Challenge [4] and others. Unfortunately, the competitions allow us to compare algorithms only indirectly. The result of the competition depends on the level of system integration and the engineering skills of a given team, but it is not possible to evaluate the performance of a particular robot subsystem or algorithm. If an autonomous vehicle manages to complete successfully all the navigation tasks, that means the whole system works properly. In contrast, if a robot fails, the localization system could work correctly, but other subsystems caused the problem.

Over the last two decades, robotics community started to work on creating common datasets and benchmarks for comparing different algorithms. Even though there are approaches allowing the localization and mapping quality to be assessed without comparing it to ground truth [5], in general, we still need a ground truth trajectory for comparison. Since such a trajectory is not available in principle, as there is no source of information able to provide the real noise-free path of a robot, researchers try to obtain a ground truth trajectory estimate – *a reference trajectory* suitable to assess the quality of localization algorithms.

One of the first initiatives in mobile robotics was the work of Baltes [6]. He proposed a set of benchmarks to evaluate the path planning and positioning algorithms for mobile robots. The external tracking system was able to provide a reference trajectory with 3 cm precision in an area of 25 $m^2$. However, such a system was limited by design to indoor applications. Similarly, in [7] authors proposed a visual reference system for the evaluation of mobile robot navigation systems. An upward-looking camera is installed on top of the robot and visual landmarks are located throughout the indoor environment. Although such an approach gives a precise estimate of localization performance, it requires an additional sensor, landmark production, and preliminary testing area preparation as well as hard to apply to outdoor scenes.

In outdoor environments most researchers use Real-Time Kinematic (RTK) GNSS systems [8], [9] providing centimeter precision under certain conditions. This method is reliable for the outdoor environment where clear sky is available, however in urban areas many factors affect RTK solution [10]: huge buildings, bridges or tunnels corrupt GNSS measurements.

A widespread approach of evaluating algorithms in robotics is simulation. It allows reconstructing any environment and to conduct reproducible experiments. For localization, an error-free ground truth trajectory might be obtained from the simulation environment. Even though modern simulators account for many error sources and simulate many factors, they still can not account for all the real-life complexity [11]. Experiments involving sophisticated sensors such as cameras or laser scanners can only be simulated up to a certain level of accuracy, e.g., capturing environments must regard surface properties such as materials, local structures, and reflections, which is hardly possible in modern simulators.

A more recent initiative was directed toward comparing various simultaneous localization and mapping (SLAM) methods. It resulted in the creation of the widely used SLAM-benchmarks [12], [13]. Factor Graph-based smoothing methods [14] are widely used for the reference trajectory estimation due to ability to compute the path using all the available observations simultaneously solving the over constrained weighted least-squares problem in offline. Thus, the data sets [15], [16] include a reference trajectory generated by the application of an offline graph SLAM algorithm. However, filter-based localization algorithms are still widely used in industry and researches [17], [18] due to fast computations and comparable (to Graph-based algorithms) level of accuracy [19], [20]. This allows the filter to produce an estimation with high frequency, which is significant for the safety requirements.

For Monte Carlo-based localization systems, Bayesian filtering and smoothing techniques are often used for this purpose. These approaches use observations $Z_{1:T} = [Z_1, \ldots, Z_T]$, control signals $U_{1:T} = [U_1, \ldots, U_T]$ and

**TABLE 1.** Summary of reference trajectory generation methods.

| Method | Advantages | Disadvantages |
|---|---|---|
| Existing datasets | • Algorithms can be compared on the same input data and use the same reference.<br>• Already available for some problems. | • Hard and expensive to collect if not available for a particular problem.<br>• There is often no dataset with the exact set of sensors or environment features required.<br>• Absence of datasets for specific tasks: warehouse navigation, public roads. |
| Simulations | • Relatively low cost: doesn't involve building new hardware, but high-fidelity simulation requires software engineering, which can be costly.<br>• Arbitrary simulated environment. | • Can be difficult to create<br>• Does not take into account many factors from the real world (imperfect calibration, sensor errors, weather conditions) |
| Reference tracking systems | • High precision (down to few centimeters). | • Expensive.<br>• There might be a need for an additional algorithm (localization & mapping system in case of LIDAR).<br>• Limitations by the capabilities of sensor: motion capture systems in the outdoors, LIDAR in foggy weather conditions.<br>• Limited by indoor applications: RTK GNSS. |
| Smoothing methods | • No need for additional sensors.<br>• No need for additional infrastructure.<br>• Suits any environment. | • Less precise than simulated data and reference tracking systems in some cases. |

knowledge of initial distribution over the hidden state $P(X_0)$, as well as motion model $P(X_t|X_{t-1}, U_t)$ and observation model $P(Z_t|X_t)$ to estimate current state distribution $P(X_t|Z_{1:t}, U_{1:t})$, smoothed distribution $P(X_t|Z_{1:T}, U_{1:T})$ or maximum a posteriori (MAP) sequence of states $X^{MAP}_{1:T} \triangleq \underset{X_{1:T}}{\operatorname{argmax}} P(X_{1:T}|Z_{1:T}, U_{1:T})$. Without loss of generality, we will further include control $U$ in the other observations $Z$, since they are usually measured, for instance, from wheeled odometry. The works [21], [22], and [23] provide an overview of Bayesian smoothing techniques. For instance, they describe the Forward-Backward (FB) smoother:

$$P(X_t|Z_{1:T}) = \underbrace{P(X_t|Z_{1:t})}_{\text{filtering}} \int \frac{\overbrace{P(X_{t+1}|Z_{1:T})}^{\text{smoothed}} \overbrace{P(X_{t+1}|X_t)}^{\text{dynamics}}}{\underbrace{\int P(X_{t+1}|X_t)P(X_t|Z_{1:t}), dX_t}_{\text{state prediction}}} dX_{t+1}, \quad (1)$$

or Two-Filter smoother:

$$P(X_t|Z_{1:T}) \propto \underbrace{P(X_t|Z_{1:t})}_{\text{filter 1}} \underbrace{P(Z_{t+1:T}|X_t)}_{\text{filter 2}}. \quad (2)$$

The maximum a posteriori sequence of states can be found with the dynamic programming Viterbi algorithm [24]. It estimates the most likely sequence of hidden states in the context of hidden Markov model with finite state space. To apply the Viterbi algorithm for reference trajectory estimation, one needs to perform space discretization. However, robot state-space discretization leads to an intractable number of states, since the Viterbi algorithm has $O(N^2T)$ complexity, where $N$ is the number of possible robot's states, $T$ – the number of time steps i.e. lengths of trajectory in the time. Note thin time is usually discretized. To overcome computational complexity, the Particle Filter (Monte Carlo localization) can be applied to estimate the most probable states efficiently, as proposed in [24] and [25]. The Viterbi algorithm is then applied only to the states represented by particles. The same approach is used in this work. Note that

the approach could be further improved for high-dimensional spaces as proposed in [26], where the dual-tree recursion algorithm reduces computational complexity to $O(N \log N)$.

The prior distribution $P(X_0)$ is usually considered to be known and dense around the ground truth pose of the robot in time $t = 0$ (e.g. in [25]). However, in the real world, this is not always the case, since the initial position of the robot can be completely unknown as in ≪ robot problem≫ [27] or known only approximately. For the latter, one can imagine an autonomous car, the position of which is initialized with a GNSS signal with a margin of error that can reach tens of meters in an urban environment [28].

In this work, we propose a novel method to improve reference trajectory estimation via prior distribution refinement in a state space represented by the Monte Carlo localization algorithm. This leads to a significant improvement in the entire trajectory. A Particles' Transition Graph is introduced to store the information about filtering distribution. We then apply the Viterbi algorithm to the refined distribution to obtain the most likely sequence of states and estimate a reference trajectory of the robot. We also demonstrate the limitations of the Viterbi algorithm applied to Particle Filter localization with an imprecise prior distribution.

## III. REFERENCE TRAJECTORY ESTIMATION

This section consists of 4 parts which describe our approach to reference trajectory estimation: (A) a revision of Monte Carlo localization with the Particle Filter, (B) Viterbi smoothing technique, (C) Particles' Transition Graph for filtering state representation, and (D) Prior Distribution Refinement (PDR) method.

### A. MONTE CARLO LOCALIZATION AND PARTICLE FILTER

Before talking about smoothing and reference trajectory reconstruction, let us briefly revise Monte Carlo localization and the underlying Particle Filter (PF) algorithm. Having a robot with a state of $X_t$ in time $t$ is represented by belief $P(X_t)$ – a continuous probability density function distributed over the state space, control/odometry signals $U_{1:t}$ and set of measurements $Z_{1:t}$ the Particle Filter estimates a posterior (joint probability distribution) $X_{0:t}$ overall state sequences as follows:

$$\begin{aligned} P(X_{0:t}) &= P(X_{0:t}|U_{1:t}, Z_{1:t}) \\ &= \eta P(Z_t|X_{0:t}, U_{1:t}, Z_{1:t-1}) P(X_{0:t}|Z_{1:t-1}, U_{1:t}) \\ &\cong \eta P(Z_t|X_t) P(X_t|X_{t-1}, U_t) P(X_{0:t-1}|Z_{1:t-1}, U_{1:t-1}) \end{aligned} \tag{3}$$

With the Monte Carlo paradigm, the belief of $X_t$ is represented by a discrete probability density function with a set of $N$ particles $X_t = \{x_t^1, x_t^2, \ldots, x_t^N\}$. As described in [3], the approach consists of 4 main steps: (1) initialize $X_0$, (2) sample particles $X_1^{\{N\}} \sim P(X_1|X_0^{\{N\}}, U_1)$ after $U_1$ control/odometry signal is applied/measured, (3) get measurements $Z_1$ and calculate $P(Z_1|X_1)$, and (4) resample particles $X_1^{\{N\}} \sim P(X_1|X_0^{\{K\}}, U_1, Z_1)$. By recursive repetition of these steps with each new signal $U_t$ and measurements $Z_t$, PF estimates a posterior $X_{0:t}$. Including control $U$ into other observations $Z$, the joint distribution may be rewritten as follows:

$$P(X_{0:t}|Z_{0:t}) = P(X_{0:t-1}|Z_{0:t-1}) P(Z_t|X_t) P(X_t|X_{t-1}) \tag{4}$$

### B. VITERBI SMOOTHING

The Viterbi algorithm [24] is a dynamic programming algorithm that can be used to obtain the maximum a posteriori (MAP) probability estimate of the most likely sequence of hidden states in the Hidden Markov Model (HMM):

$$X_{0:t}^{MAP} \triangleq \underset{X_{0:t} \in \otimes_{t=0}^{T} \{X_t^{(i)}\}_1^N}{\operatorname{argmax}} P(X_{0:t}|Z_{0:t}), \tag{5}$$

with the joint probability distribution as equation 4.

Adapting to the robot localization problem, assume a tuple $X_{0:T} = [X_0, .., X_T]$ of $T$ elements, where each $X_t$ is a robot's state in time $t$ and its belief $P(X_t)$ (probability density function) is represented by $N$ particles. For planar 2D, each particle $X_t^i \in R^4$ is represented by $[x_t^i, y_t^i, \theta_t^i, \omega_t^i]$ – position, orientation and the weight of a particle proportional to the likelihood of an observation $P(Z_t|X_t^i)$.

An expectation of robot state in time $t$ can be expressed as $X_t = \mathbb{E}_{[X_t|Z_{1:t}]} = \sum_{n=1}^{N} \omega_t^i \cdot X_t^i$. Let $\Delta X_t = \begin{matrix} [r]\Delta X_{t,t-1}^{trans} \\ \Delta X_{t,t-1}^{rot} \end{matrix} = \begin{matrix} [r]\left|\left|\mathbb{E}_t^{[x,y]} - \mathbb{E}_{t-1}^{[x,y]}\right|\right|_2 \\ \mathbb{E}_t^{\theta} - \mathbb{E}_{t-1}^{\theta} \end{matrix}$, $\theta_t \in [-\pi : \pi]$. The equivalent notations will be used for every pair of particles: $\Delta X_t^{ij} = \begin{matrix} [r]\Delta X_{t,t-1}^{ijtrans} \\ \Delta X_{t,t-1}^{ijrot} \end{matrix} = \begin{matrix} [r]||X_t^{i[x,y]} - X_{t-1}^{j[x,y]}||_2 \\ ||X_t^{i[\theta]} - X_{t-1}^{j[\theta]}|| \end{matrix}$. Then, the probability of particle $j$ for time $t-1$ to be transited to the position of particle $i$ in time $t$ is represented as follows:

$$\begin{cases} P_{trans}(X_t^i|X_{t-1}^j) = \eta \exp\left(\Delta X_{t,t-1}^{ij_{trans}} \text{-} \Delta X_{t,t-1}^{trans}, \sigma_{trans}^2\right), \\ P_{rot}(X_t^i|X_{t-1}^j) = \eta \exp\left(\Delta X_{t,t-1}^{ij_{rot}} \text{-} \Delta X_{t,t-1}^{rot}, \sigma_{rot}^2\right), \\ P(X_t^i|X_{t-1}^j) = P_{trans}(X_t^i|X_{t-1}^j) P_{rot}(X_t^i|X_{t-1}^j), \end{cases} \tag{6}$$

where $\sigma_{rot}^2 = \sigma_\theta^2$, $\sigma_{trans}^2 = \sigma_x^2 + \sigma_y^2$ – variances of motion model noise. By multiplying probabilities, we assume that the rotation and translation parts of motion are uncorrelated. Fig. 1 illustrates an example of calculating the MAP trajectory of a robot for time interval $[0{:}t]$ by maximazing the joint distribution.

### C. PARTICLES' TRANSITION GRAPH

Let us introduce the notion of a Particles' Transition Graph (PTG) which represents the filtering distribution $P(X_t|Z_{0:t})$. Mathematically speaking, it is a directed forest – a union of directed trees [29]. Each vertex of PTG (see Fig. 2) is a particle that stores information about its position, likelihood, and the resampling index – the index of the particle at the site of which the current particle was transited after the resampling process. For example, if there is a particle $i$ in

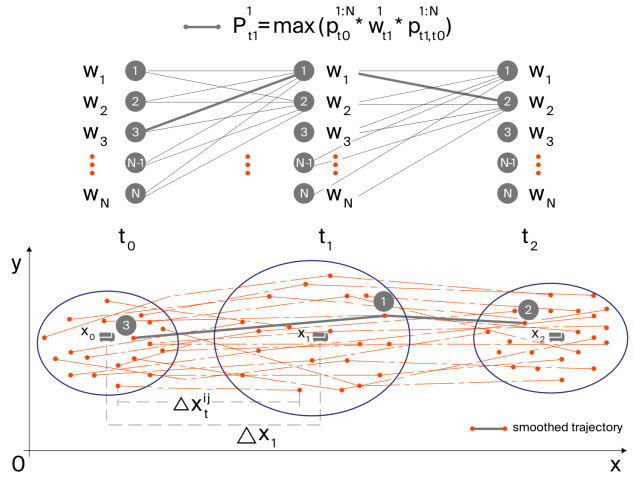



**FIGURE 1.** Graphical example of maximum a posteriori trajectory calculation with the Viterbi smoothing technique based on particles' poses. The green line (smoothed trajectory) indicates the most likely path. The orange dots are particles' poses, the green triangle in the middle of an ellipsoid – the estimated robot pose. $w_t^i$ - weight (likelihood) of a particle in time $t$, $P_{t_1}^1$ – the most likely path between state $t_0$ and state $t_1$ in if a MAP trajectory goes through particle 1 in time $t_1$.

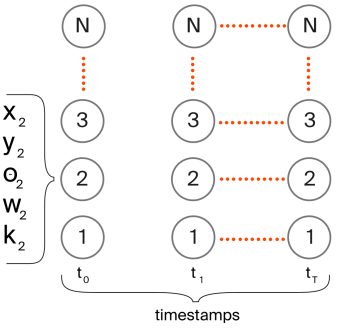



**FIGURE 2.** Scheme of the Particles' Transition Graph vertices at different timestamps $t_i$. Each vertex represents a particle $p_i$ with an information vector: position $[x_i, y_i, \theta_i]$, weight $w_i$ and resampling index $k_i$.

time $t$ with resampling index $k = j$, it means that after the resampling procedure, a particle $i$ has been relocated to the position of a particle $j$. However, the edges of vertices in PTG can be defined differently. Fig. 3 (a) and (b) illustrate that there are 2 ways of constructing PTG for the same filtering distribution. The similarity between them is in the connection of vertices when particle $i$ has a resampling index $k = i$: after the resampling procedure, a particle did not change its position. The difference is the way the vertices are connected when the resampling index $k \neq i$. To make it clear, let us see an example in Fig. 3 (a), which illustrates the first way of PTG construction. Particle 7 in time $t = 1$ spawned particles 4, 5, 6, 7, 8. It means that those five particles had the same resampling index $k = 7$ after the resampling process. Therefore, they had the same position as particle 7 before being transferred to the next state. As the toy example shows, there is a tree with the root in particle 7 in time $t = 0$ connected with all vertices in time $t = 9$. Whereas case (b) in Fig. 3 represents the same filtering distribution with the same relocation of particles 4, 5, 6, 7, 8 to the position

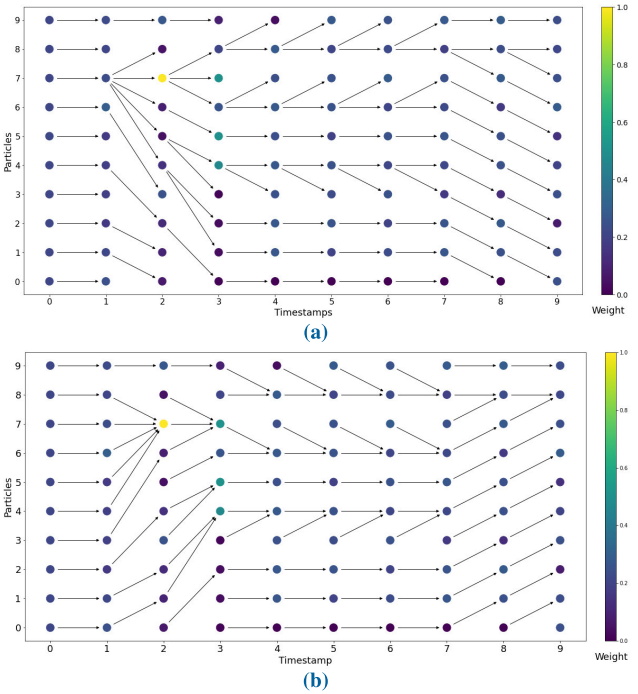



**FIGURE 3.** Example of Particle Transition directed graphs based on particle transitions during PF resamplings. (a): ancestor particle creating children in its position; (b): children particles relocating to the ancestor particle position. The nodes (circles) represent particles at different time steps, and the edges (arrows) illustrate the resampling: which particle was sampled. Note that in PF, the same particle can be sampled several times. The color of nodes illustrates the importance (weight) of a particle (the brighter - the higher).

of particle 7 after the resampling but the edges of vertices are different. The roots of trees are at the last timestamp $t = 9$ but not at the beginning as in (a). The example clearly illustrates the difference between the two paradigms: there are 3 trees with roots in particles 6, 7, 8 in time $t = 9$ connecting all the particles in time $t = 0$.

Despite the fact that scheme (a) for defining the edges guarantees the existence of at least 1 root connecting all the vertices of $t = T$ with one or more vertices of $t = 0$, in our experiments we analyze PTG constructed with approach (b) which allows to find far more unique particles (we call them "ancestors") in time $t = 0$. However, one can not guarantee the existence of the root between $t = [0 : T]$ for this case, especially for a long sequence of states. In practise, building PTG with scheme (a) leads to an insufficient number of ancestor particles: 1 or 2 in 99% cases for a 10-minute sequence of measurements with 1000 particles. Moreover, these 1-2 ancestors can be far from real robot's position. Our goal is to find such a priori that most plausibly describes the initial robot's state; thus, a large number of ancestors is preferable for the future analysis, as few particles unlikely provide a comprehensive belief about an initial robot's state.

### D. PRIOR DISTRIBUTION REFINEMENT

Depending on available information about the robot's initial position, a prior distribution $P(X_0)$ is modeled differently. In case of complete uncertainty about the robot's initial

state, it is reasonable to spread the hypothesis (particles) uniformly across the map. Note that the number of particles needs to be sufficient for representing the likely robot's state and filter convergence. By filter convergence, we assume a kinematically attainable estimation of the robot's trajectory. However,if the particles are uniformly distributed across the map, there may be kinematically unattainable jumps in an expectation $\mathbb{E}_{[X_t|Z_{1:t}]}$ due to resampling. If the robot's pose is approximately known, a common practice is to assume that $X_0 \sim \mathcal{N}(\mathbb{E}_0, \Sigma_0)$, where $\mathbb{E}_0$ is an assumption about the robot's initial pose and confidence level $\Sigma_0$ of the Gaussian distribution. Obviously, a good estimation of the robot's initial state leads to faster convergence of PF.

Unfortunately, a reasonable confidence level in the initial position is elusive, as it depends on many factors. An incorrect assumption of the initial covariance matrix $\Sigma_0$ may lead to an incorrect or inefficient convergence of PF. If $\Sigma_0$ overestimates a priori, the filter needs more time to converge and this process will be accompanied by kinematically unattainable jumps of the robot's pose. The worst situation happens when an estimation converges to an incorrect location. For example, if there are two similar featureless parallel roads for PF estimation with a measurement model based on road markup, this could localize a robot on the neighboring side. Conversely, if $\Sigma_0$ underestimates a priori, a PF estimation for the beginning part of the trajectory might be tied up to some local features (e.g. a road border) because the spread of particles is too narrow.

To fix this, we propose a Prior Distribution Refinement (PDR) algorithm which iteratively runs the Particle Filter, analyzes the Particles' Transition Graph, removes edges that represent unlikely resampling relocations (''jumps''), and tracks the initial ancestors of the particles that survived to the last timestamp $T$ modifying the prior distribution.

The input is the observations needed for PF, and the output – a refined filtering distribution represented as PTG. The resulting trajectory can be directly reconstructed from PTG. Afterwards, we smooth the refined distribution with the Viterbi technique to obtain the most precise results.

The proposed algorithm is described in listing 1. The initialization step of the algorithm is defined in line 1 where the filtering distribution $P(X_t|Z_{1:t})$ is initialized with normal distribution $X_0 \sim \mathcal{N}(\mathbb{E}_0, \Sigma_0)$ or randomly $X_0 \sim \mathcal{U}_{[a,b]}$ inside $[a, b]$ rectangular area. In line 2, the loop starts with calculating the filtering distribution $P(X_t|Z_{1:t})$ and saving it as PTG. In lines 3-4, we analyze the current state $X_t$ by comparing the maximum Euclidean distance between each particle and expectation $\mathbb{E}(X_t)$ (separately for position and rotation) with the precomputed threshold $\epsilon$. If either position or rotation difference is above the corresponding $\epsilon$ (line 5), it starts checking the resamplings of each particle $i$ with its ancestor $k$ for the current state (line 6-7). The relocation analysis of the particles' current state showed competitive results in our experiments but, of course, is not optimal and can be improved in further research. Thus, it detects the resampling which leads to significant relocation (''jumping'')

**Algorithm 1** Prior Distribution Refinement

**Input:** $Z_{1:t}$-measurements.
**Output:** $P(X_t|Z_{1:t})$ as *PTG*.
*Initialisation*:
1: initialize $X_0 \sim \mathcal{N}(\mathbb{E}_0, \Sigma_0)$;
2: **while** stopping criterion **do**
calculate $P(X_t|Z_{1:t})$ - forward PF run;
*Main part*:
3: **for** $X_t \leftarrow 1$ to $T$ **do**
4: compute $\Delta = max(||X_t^i - E(X_t)||_2)$;
5: **if** $\Delta \geq \epsilon$ **then**
6: **for** $X_t^i \leftarrow 1$ to $N$ **do**
7: compute $\delta = ||X_t^{i_{[x,y]}} - X_t^{k_{[x,y]}}||_2$,
8: **if** $\delta \geq \epsilon$ **then**
remove edge in *PTG*,
9: **end if**
10: **end for**
11: **end if**
12: **end for**
13: start Depth-First Search at $X_T$ till $X_0$
14: make set $S = [s_1, s_2, .., s_K]$ of unique particles at $X_0$;
15: recalculate initial covariance matrix $\tilde{\Sigma}_0$ and threshold $\epsilon$,
16: initialize new $X_0 \sim [\mathcal{N}_0(\mathbb{E}_{s_0}, \tilde{\Sigma}_0), \ldots, \mathcal{N}_k(\mathbb{E}_{s_k}, \tilde{\Sigma}_0)]$;
17: **end while**
18: **return** $P(X_t|Z_{1:t})$ as *PTG*.

of a particle and removes the corresponding edge it PTG (line 8). The unique particles which have the connection of state $X_T$ with $X_0$ are tracked (line 13) with Depth-First Search (DFS) [29]. It is worth mentioning, there exist cases when no initial particles can be tracked due to the edges in PTG having been removed. This is the drawback of the proposed method. When the particle set $S = [s_1, s_2, .., s_K]$ is formed, the prior distribution should be initialized for the next PF forward run. We consider ancestor particles to be perspective candidates, thus their positions are used as means of future Gaussian distributions $\tilde{X_0} \sim [\mathcal{N}_0(\mathbb{E}_{s_0}, \tilde{\Sigma}_0), \ldots, \mathcal{N}_k(\mathbb{E}_{s_k}, \tilde{\Sigma}_0)]$ (line 15) with equal covariance matrices $\tilde{\Sigma}_0 = \tilde{\Sigma}_k$, $\forall k = [1, .., K]$. We randomly sample $N$ particles out of $K$ Gaussians. The covariance matrix $\tilde{\Sigma}_0 = \begin{bmatrix} \sigma_{x_0}^2 & 0 & 0 \\ 0 & \sigma_{y_0}^2 & 0 \\ 0 & 0 & \sigma_{\theta_0}^2 \end{bmatrix}$ is calculated as $\tilde{\Sigma}_0 = \begin{cases} \frac{(N-K)^2}{N^2}\tilde{\Sigma}_0, K \neq N \\ \frac{1}{N^2}\tilde{\Sigma}_0, K = N \end{cases}$ (line 14). Unlike the multivariate Gaussian mixture model [30] we do not sample initial particles from $\tilde{X_0} \sim \sum_{i=1}^{K} \mathcal{N}(\mathbb{E}_{s_i}, \tilde{\Sigma}_0)$, as it may spread out the density between several remote regions of the map. The decrease of initial variances is needed to prevent a situation of unguided growth of the particles' spread. For instance, there are 1000 particles in the filter and only 100 of them are perspective ancestors spread widely across the map. If one uses unmodified $\Sigma_0$, the complete set of 1000 particles will

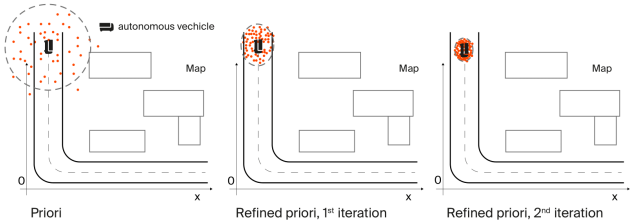



**FIGURE 4.** Scheme of prior distribution refinement after 2 iterations of the proposed algorithm. Initially, the particles (red dots) are distributed on the map with high variance around an autonomous vehicle.

**FIGURE 5.** Simulated outdoor environment from Carla [11].

**FIGURE 6.** Outdoor testing environment.

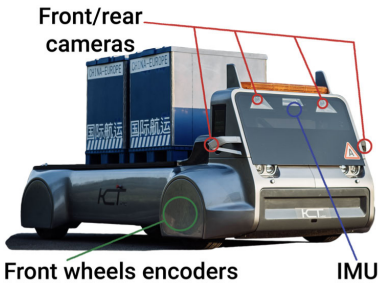



**FIGURE 7.** Logistic autonomous vehicle used to collect experimental data. Note: image is reprinted from [31].

fill in even more space of the map. Thus, the algorithm increases the uncertainty that contradicts the original goal of the method. A new threshold $\epsilon$ (line 14) is being recalculated as follows: $\epsilon = 3 \cdot std||S_k - E(S)||_2$ separately for rotation and translation, where *std* is a standard deviation. However, if there is a lack of measurements or unstable income of observations during some part of the trajectory, the uncertainty grows as well as the number of resamplings with unlikely particles' relocation. This leads to additional Particles' Transition Graph check (line 4, in algorithm 1) and edges liquidation (line 7). Thus, there might be no connection between state $X_T$ and $X_0$. This leads to an inability to find ancestor particles and the PDR algorithm stops - this is one of the stopping criteria for the "while" loop in line 2. Another stopping criterion is a similarity between ancestors sets $S$ for two sequential iterations of PDR. During initialization procedure, the algorithm samples new particles out of $\tilde{X_0} \sim [\mathcal{N}_0(\mathbb{E}_{s_0}, \tilde{\Sigma}_0), \ldots, \mathcal{N}_k(\mathbb{E}_{s_k}, \tilde{\Sigma}_0)]$. If set $S_i$ of the PDR iteration $i$ is similar to the set $S_{i-1}$ of iteration $i-1$, due to the proposed initial covariance recalculation method based on ancestors particles from set $S$, the new $P(X_0)_i$ is

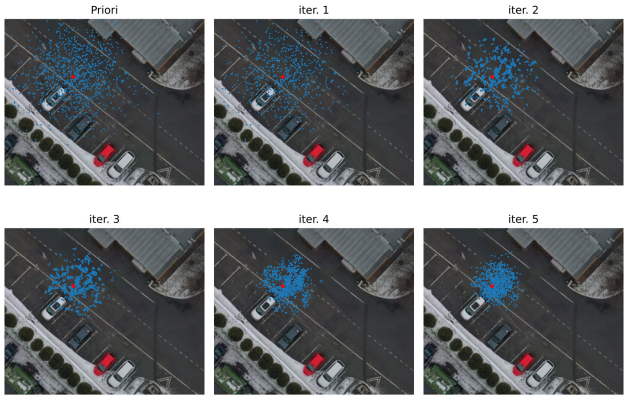



**FIGURE 8.** The refined prior distribution after 5 iterations of PDR algorithm. Blue dots – particles (hypotheses), red dot – ground truth position of a vehicle.

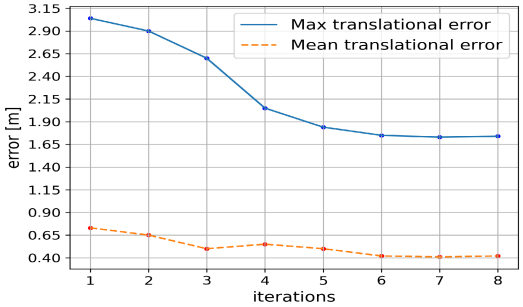



**FIGURE 9.** Maximum and mean translation error in meters for 8 PDR+Viterbi iterations.

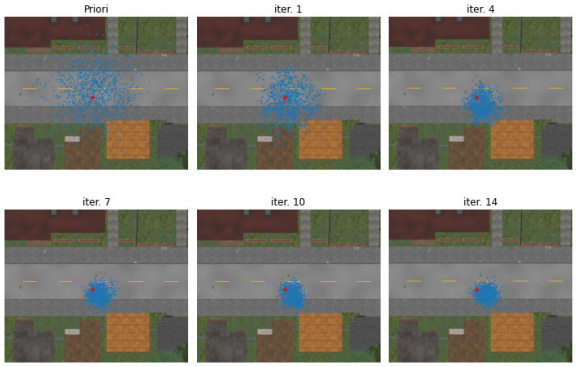
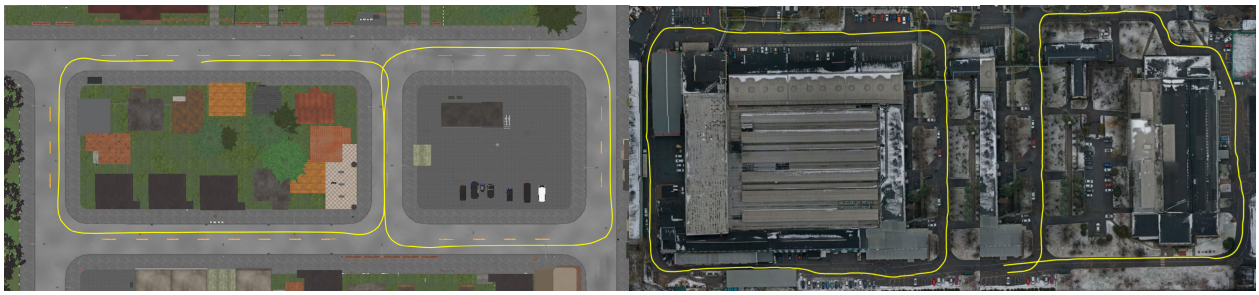



**FIGURE 10.** The refined prior distribution after 14 iterations of the PDR algorithm. Blue dots – particles (hypotheses), red dot – ground truth position of the vehicle.

similar to $P(X_0)_{i-1}$. Thus, no sense to run PF many times with the same priori in a "while" loop. The proposed method of particles relocation analysis in PDR and initial covariance recalculation based on ancestor might be improved in further research.

## IV. EXPERIMENTAL RESULTS

To validate the proposed algorithm, we conducted tests both on synthetic (Fig. 5) and real-world data (Fig. 6). Two real-world routes and one simulated one were used



(a) Route in Carla virtual environment. (b) Route 1 in real environment. (c) Route 2 in real environment.

**FIGURE 11. Test routes used for algorithms evaluation.**

for the evaluation. The routes are presented in Fig. 11. An autonomous logistic vehicle (Fig. 7) is equipped with the Particle Filter-based localization system described in [31]. The sensor measurements $Z_{1:t}$, control signals $U_{1:t}$, and Particles' Transition Graph from the test runs were recorded and used as input for the experiments.

In all test runs, the Particle Filter was initialized with particles normally distributed around the ground truth initial vehicle pose $X_0 \sim \mathcal{N}(\mathbb{E}_0, \Sigma_0)$ with $3\sigma_{x,y} = \pm 7.5$ meters for position and $3\sigma_\theta = \pm 0.5$ radians for rotation (Fig. 10, Fig. 8). The parameters of the initial distribution were chosen to model the lack of knowledge about the initial state of the vehicle.

A particle filter with the lidar-based NDT localization algorithm [32] and GPS RTK data was used to generate a ground truth reference trajectory. The output of the NDT based localization algorithm is the vehicle's pose on a lidar map. This approach allows a precise localization for non-rapid driving to be attained. We set a narrow variance for the Gaussian noise model in order to tighten the resulting point cloud of particles. Thus, all particles are tightly coupled with the output of NDT-based localization. This method of filtering provides precise ground truth information and is used as a reference trajectory for error calculation in the outdoor real-world environment, whereas the Carla simulator provides its accurate reference.

Fig. 12 illustrates the result of reference trajectory estimation with the proposed algorithm. The figure includes the ground truth trajectory, estimated with lidar-based NDT localization, trajectory estimated with particle filter, PF-based localization smoothed with the Viterbi algorithm (named as "Viterbi"), and the result of the proposed algorithm, combining PDR and Viterbi (named as "PDR+Viterbi").

As demonstrated, the first several meters PF-based trajectory (green curve) has a significant kinematically unattainable "jump" of vehicle position as well as a starting point is far from reality, which demonstrates an unlikely PF solution. Meanwhile, the Viterbi smoother (red curve) allows obtaining a kinematically attainable trajectory but retires the initial position still shifted from the ground truth starting point (bottom right picture in Fig. 12). Finally, the combination of the PDR and Viterbi methods (purple curve) managed to overcome both problems: it found the closest initial point and decreased the number of objectionable rapid changes in the vehicle's position. Furthermore, a plausible priori entails not only the first meters of the path but improves the trajectory for the whole travel distance, which is indicated with orange ellipsoids in Fig. 12.

**TABLE 2. Numerical results of absolute pose error for different methods in the simulated environment.**

| | Translation(m) | | | | Rotation(deg) | | | |
|---|---|---|---|---|---|---|---|---|
| | max | rms | mean | std | max | rms | mean | std |
| Particle Filter | 3.04 | 0.92 | 0.73 | 0.55 | 10.62 | 1.59 | 0.92 | 1.29 |
| Viterbi | 2.47 | 0.87 | 0.68 | 0.54 | 8.24 | 1.44 | 0.70 | 1.26 |
| PDR+Viterbi | **1.70** | **0.53** | **0.42** | **0.32** | **4.58** | **1.01** | **0.55** | **0.85** |

**TABLE 3. Numerical results of absolute pose error for data sequence 1 in a real outdoor environment.**

| | Translation(m) | | | | Rotation(deg) | | | |
|---|---|---|---|---|---|---|---|---|
| | max | rms | mean | std | max | rms | mean | std |
| Particle Filter | **3.21** | **1.65** | **1.51** | **0.66** | 7.87 | **2.41** | **1.98** | **1.36** |
| Viterbi | 2.79 | 1.56 | **1.41** | 0.66 | **7.96** | **2.43** | **1.92** | **1.48** |
| PDR + Viterbi | **2.68** | **1.54** | **1.41** | **0.62** | **8.08** | **2.41** | 1.95 | 1.42 |

**TABLE 4. Numerical results of absolute pose error for data sequence 2 in a real outdoor environment.**

| | Translation(m) | | | | Rotation(deg) | | | |
|---|---|---|---|---|---|---|---|---|
| | max | rms | mean | std | max | rms | mean | std |
| Particle Filter | **4.77** | **1.78** | **1.67** | **0.62** | **21.39** | 4.49 | **3.15** | 3.19 |
| Viterbi | 3.64 | 1.58 | 1.47 | 0.59 | 22.01 | **4.79** | 3.04 | **3.70** |
| PDR + Viterbi | **2.79** | **1.45** | **1.37** | **0.49** | **11.53** | **2.80** | **2.33** | **1.56** |

Tables 2-4 present the measured quality metrics for estimated trajectories for two real-world routes and a simulated one. We evaluate absolute pose error (APE) which consists of translation, rotation parts; and inherent statistics: maximum, mean, square root mean, and standard deviation for all errors. For each $i$-th pose of the trajectory we compute separate errors for position $X_i^{trans} = [x_i, y_i]$ and rotation $X_i^{rot} = \theta_i$ as follows:

$$\begin{aligned} APE_i^{trans} &= ||X_{ref}^{trans} - X^{trans}||_2, \\ APE_i^{rot} &= ||X_{ref}^{rot} - X^{rot}||_1. \end{aligned} \tag{7}$$

"*ref*" subscript indicates an element of the reference trajectory.

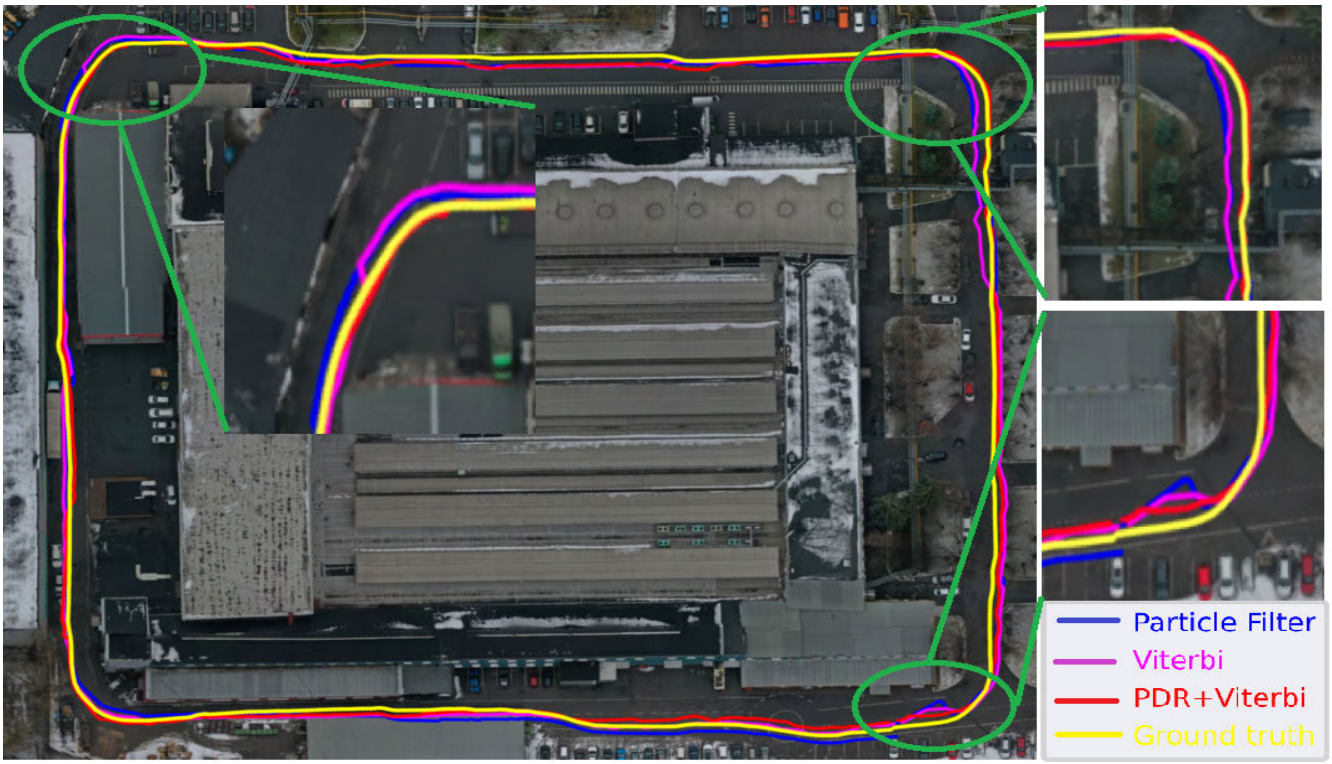



**FIGURE 12.** Trajectories of PF-based localization with initialization by area and different smoothing techniques. Green circles indicate refined parts which illustrate the benefits of PDR+Viterbi algorithm.

Fig. 9 demonstrates the decreasing mean and maximum translation error in meters with 8 iterations of the PDR+Viterbi combination for the same data set used for table 2. High maximum and mean translation errors ($\approx$ 3 m) could be explained as due to the imperfect quality of the localization system and obviously might be improved. But the main goal of the proposed approach is to show that it can significantly refine the quality and precision of the trajectory obtained with the existing localization system and allows us to use this trajectory as a reference one.

## V. CONCLUSION

As demonstrated by experiments on both simulated and real-world data, the proposed Prior Distribution Refinement algorithm combined with Viterbi smoothing allows to significantly improve the accuracy of the reference trajectory relying on data provided by Particle Filter. PDR refines an estimation of the initial state (prior distribution) which leads to faster convergence of the Particle Filter to a more precise solution as $P(X_0)$ becomes clearly known. The refined output of the filter stored as a Particles' Transition Graph becomes an input for the Viterbi smoother to obtain the most likely sequence of states. This provides a smooth trajectory with a likely initial pose, which can be used as a reference for evaluating different Particle Filter-based localization algorithms and parameters fine-tuning. The proposed algorithm allows reconstructing most likely robot's path before the PF-based localization system stabilizes and obtain meaningful estimation of the robot's location in case of uncertain initial pose: widespread particles around some assumption on the map. However, the performance of the proposed method depends on many factors. As Particle Filter-based localization system does not guarantee a globally optimal solution, before using PDR + Viterbi, one should check if the robot's trajectory is globally consistent and if the filter has obtained a meaningful solution for a major part of the path. Worth mentioning, the proposed method is applicable not only for reference trajectory estimation in robot's localization problem but any kind of challenge where Monte Carlo-based filtering is applicable. If a state of any process is described by a multivariate random variable and represented by many particles (Monte Carlo paradigm) the information about a posteriori probability estimate of the state with the given observations could be stored as Particles' Transition Graph. Many graph analysis techniques might be applied to the PTG; one of them is a Viterbi algorithm, which gives the maximum a posteriori probability estimate of the most likely sequence of hidden states.

## REFERENCES

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, May 1999, pp. 1322–1328.

[2] P. Heinemann, J. Haase, and A. Zell, "A combined Monte-Carlo localization and tracking algorithm for RoboCup," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 1535–1540.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.
[4] F. Sauerbeck, L. Baierlein, J. Betz, and M. Lienkamp, "A combined LiDAR-camera localization for autonomous race cars," *SAE Int. J. Connected Automated Vehicles*, vol. 5, no. 1, pp. 61–71, Jan. 2022.
[5] A. Kornilova and G. Ferrer, "Be your own benchmark: No-reference trajectory metric on registered point clouds," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Aug. 2021, pp. 1–8.
[6] J. Baltes, "A benchmark suite for mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 2, Oct./Nov. 2000, pp. 1101–1106.
[7] H. Kikkeri, G. Parent, M. Jalobeanu, and S. Birchfield, "An inexpensive method for evaluating the localization performance of a mobile robot navigation system," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 4100–4107.
[8] C. Cherciu, A. N. Croitoru, P. Mugur Svasta, and D. I. Nastac, "Analysis of RTK corrections for GNSS ground stations," in *Proc. IEEE 25th Int. Symp. Design Technol. Electron. Packag. (SIITME)*, Oct. 2019, pp. 304–307.
[9] Q. Sun, J. C. Xia, J. Foster, T. Falkmer, and H. C. Lee, "Pursuing precise vehicle movement trajectory in urban residential area using multi-GNSS RTK tracking," *Transp. Res. Proc.*, vol. 25, pp. 2361–2376, Jan. 2017.
[10] R. B. Ong, M. G. Petovello, and G. Lachapelle, "Assessment of GPS/GLONASS RTK under various operational conditions," in *Proc. 22nd Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS)*, 2009, pp. 3297–3308.
[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.
[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.
[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "$G^2o$: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 3607–3613.
[15] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *Int. J. Robot. Res.*, vol. 38, no. 6, pp. 642–657, May 2019.
[16] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan north campus long-term vision and lidar dataset," *Int. J. Robot. Res.*, vol. 35, no. 9, pp. 1023–1035, Aug. 2016.
[17] C. Nalini Iyer, A. Kulkarni, R. Shet, and U. Keerthan, "Localization of self-driving car using particle filter," in *Advances in Computing and Network Communications*, S. M. Thampi, E. Gelenbe, M. Atiquzzaman, V. Chaudhary, and K.-C. Li, Eds. Singapore: Springer, 2021, pp. 147–155.
[18] P. Slowak and P. Kaniewski, "LIDAR-based SLAM implementation using Kalman filter," *Proc. SPIE*, vol. 11442, Feb. 2020, Art. no. 114420N.
[19] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, 2012.
[20] D. Wilbers, C. Merfels, and C. Stachniss, "A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Feb. 2019, pp. 220–225.
[21] M. Klaas, M. Briers, N. D. Freitas, A. Doucet, S. Maskell, and D. Lang, "Fast particle smoothing: If I had a million particles," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 481–488.
[22] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, vol. 12. New York, NY, USA: Oxford Univ. Press, 2009, pp. 656–704.
[23] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state–space models," *Ann. Inst. Stat. Math.*, vol. 62, no. 1, pp. 61–89, 2010.
[24] S. Godsill, A. Doucet, and M. West, "Maximum a posteriori sequence estimation using Monte Carlo particle filters," *Ann. Inst. Stat. Math.*, vol. 53, pp. 82–96, Mar. 2001.
[25] K. Zoubert-Ousseni, C. Villien, and F. Le Gland, "Comparison of post-processing algorithms for indoor navigation trajectories," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2016, pp. 1–6.
[26] M. Klaas, D. Lang, and N. D. Freitas, "Fast maximum a-posteriori inference on Monte Carlo state spaces," in *Proc. Int. Workshop Artif. Intell. Statist.*, 2005, pp. 158–165.
[27] S. Lee, S. Lee, and S. Baek, "Vision-based kidnap recovery with SLAM for home cleaning robots," *J. Intell. Robot. Syst.*, vol. 67, no. 1, pp. 7–24, 2012.
[28] D. Maier and A. Kleiner, "Improved GPS sensor model for mobile robots in urban terrain," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 4385–4390.
[29] N. R. Malik, "Graph theory with applications to engineering and computer science," *Proc. IEEE*, vol. 63, no. 10, pp. 1533–1534, Oct. 1975.
[30] D. J. B. S. Hand Everitt, *Finite Mixture Distributions*. Dordrecht, The Netherlands: Springer, 1981.
[31] O. Shipitko, V. Kibalov, and M. Abramov, "Linear features observation model for autonomous vehicle localization," in *Proc. 16th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Dec. 2020, pp. 1360–1365.
[32] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.