# A Comparison of Particle Filter and Graph-based Optimization for Localization with Landmarks in Automated Vehicles

Daniel Wilbers[1] Christian Merfels[1] Cyrill Stachniss[2]

***Abstract*— It is an essential capability of mobile robots and automated vehicles to localize themselves in a map. Multiple state estimation techniques exist to this end, and it is often unclear which one to employ. In this paper we compare two prominent techniques in the context of automated vehicles: particle filters and graph-based localization. The latter is the state-of-the-art approach to simultaneous localization and mapping, and is also superseding the particle filter as the state-of-the-art for pure localization. We compare both algorithms to show why. For both state estimation algorithms, we detect pole-like objects in laser scanner data and compare them against a prebuilt landmark map. The main novelty of this work is the experimental comparison on a real prototype vehicle. Furthermore, we discuss the different advantages of both algorithms in the context of automated driving and additionally show how both approaches can adapt their computational demand to the available resources at runtime.**

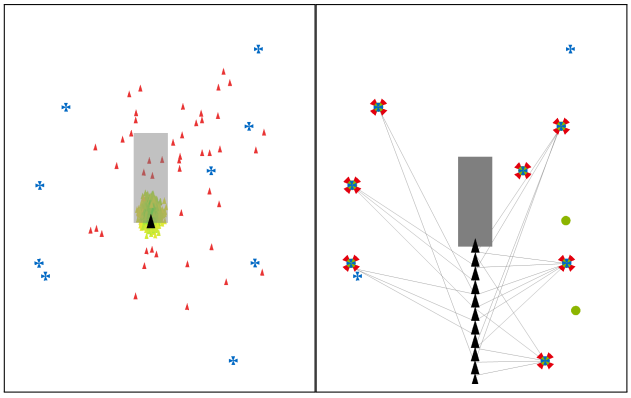
## I. Introduction

Localization is a fundamental task for mobile robots. Particle filter localization [1], [2] (often called Monte Carlo localization) is a highly popular approach in the mobile robotics community. For simultaneous localization and mapping (SLAM) approaches, the focus has clearly shifted in the last years from using Rao-Blackwellized particle filters [3] towards graph-based approaches [4], [5]. However, this cannot be said for pure localization approaches. The majority of these approaches seem to be implemented with either a Kalman filter variant or a particle filter. Only recently a few graph-based localization approaches were presented [6], [7]. The question is therefore: is it beneficial for pure localization approaches to prefer the graph or particle filter paradigm?

In this paper we provide insights and experiments to compare these two. Fig. 1 illustrates both concepts. Our investigation focuses on the application in automated vehicles. Nevertheless our findings are in principle transferable to other domains.

Localization is a prerequisite for interpreting information stored in maps. This information is beneficial for navigation, path planning, and perception. If we want to be able to make use of this map information, then we need to be able to localize with respect to the map. Thus, in this paper we focus on the problem of map-based self-localization in the context of automated vehicles.

Nowadays, Global Navigation Satellite Systems (*GNSS*) are commonly used for the localization in road vehicles. The most accurate of these systems combine Inertial Measurements Units and Real-time kinematics to achieve centimeter-level results. Besides their great costs, satellite based systems suffer from strong multipath and blocked line of sight effects especially in urban scenarios. A common approach for highly available localization in scenarios without or limited satellite reception is to use landmark-based localization techniques which we investigate in this paper. We rely on pole-like vertical objects as landmarks which are for example lampposts, round pillars, vertical trees, and similar structures. Compared to using raw sensor data maps (e.g., point clouds), semantic landmark maps are easier to inspect for correctness, require smaller storage and thus are faster to process. It is possible to choose a different kind of landmark type for other domains without contradicting the general findings of this paper.



(a) Particle filter (b) Graph-based localization

Fig. 1: Localization problem in the same situation solved with two different approaches. (a) Particle filter. The color of a particle represents its current weight. If a weight is high the color ranges from green to yellow. Recovery particles from GNSS receive a low weight (red). (b) Sliding window graph. Finding matches (red crosses) between landmark observations (green dots) and map landmarks (blue crosses) allows us to constrain the vehicle trajectory (black triangles). The connections (gray lines) illustrate from which poses a landmark was observed. The current vehicle pose is depicted as a gray rectangle.

The main contribution of this paper is a comparison of particle filter and graph-based localization in the context of automated vehicles. We discuss the characteristics of both algorithms and investigate three key aspects in the context of automated driving: (i) accuracy, (ii) adaptive behavior in terms of computational resources, and (iii) benefit of estimating old poses.

We experimentally evaluate these questions on data gathered on a real prototype vehicle.

[1]Daniel Wilbers and Christian Merfels are with Volkswagen Group Research, Wolfsburg, and Institute of Geodesy and Geoinformation, University of Bonn, Germany.

[2]Cyrill Stachniss is with Institute of Geodesy and Geoinformation, University of Bonn, Germany.

## II. Related Work

Map-based localization is typically either carried out as a standalone approach by comparing sensor against map data or as part of a SLAM approach. The usual state estimation machinery for this encompasses extended Kalman filters (EKFs) and their variants, particle filters, and graph-based approaches, which nowadays can be considered state-of-the-art. Stachniss et al. [5] review these three methodologies. We also refer to the work of Cadena et al. [8] for a recent and to Durrant-Whyte and Bailey [9] for a general overview.

Pure localization has also started off by being approached with Kalman filters, and continues so successfully for certain applications. Similarly to SLAM, particle filters have been introduced and used in the last two decades. However, few approaches propose map-based localization with graph-based systems.

Lundgren et al. [10] use camera and radar detections of landmarks for map-based self-localization with a particle filter. One of their main findings is that both sensing modalities are needed for a robust and precise result. Deusch et al. [11] focus on detecting lane markings and road paintings in grayscale camera images and laser scanner data. Their particle filter compares the detections to a self-built map and computes a global pose estimate. Similarly to Lundgren et al., they find that using both sensing modalities increases robustness and reliability at the expense of a higher computational cost. Spangenberg et al. [12] detect poles in stereo camera images. These are used in a coupled particle and Kalman filter setup to estimate the current vehicle pose by comparing them against a prebuilt map. They favor poles as type of landmark because they are distinct and long-term stable. Moreover, they can be reliably detected and efficiently stored. In a similar line of thinking, Brenner [13] proposes to use pole landmarks for global localization. Schlichting and Brenner [14] apply this system on a vehicle with an automotive-grade lidar. Schindler [15] approaches the localization problem by fusing camera-based lane marking and lidar-based pole detections with a particle filter. The measurement model employs prototype fitting to estimate the likelihood of the detections given the map. Interestingly, the approach by Levinson et al. [16] generates maps with a graph-based SLAM system, while a particle filter is employed for localization. Wu et al. [7] presented a localization approach based on graph optimization using camera and lidar data. In comparison to their work, our graph-based approach benefits from a different association strategy as well as grid independent landmark detections. Recently, Harr et al. [6] presented a pose graph approach using gray-scale cameras. Our graph-based approach uses a different association strategy and relies on different sensors resulting in a more accurate system.

An initial version of our particle filter is described by Stess [17]. Our graph-based localization approach is built on the foundation of a pose fusion system [18], [19] and our related work et al. [20]. We briefly summarize both approaches in Sec. III.

## III. Localization

Map-based localization aims to estimate the current pose of the robot with respect to the map. The pose is defined as the two-dimensional location and orientation $\boldsymbol{x}_t = [x, y, \theta]^\top$ within the coordinate frame of the map. To this end, we take the measurements $\boldsymbol{z}_t = [\boldsymbol{z}^{\mathrm{o}}, \boldsymbol{z}^{\mathrm{g}}, \boldsymbol{z}^{\mathrm{l}}]^\top$ into account. Here, $\boldsymbol{z}^{\mathrm{o}}$ denotes the measured odometry. We assume that the velocity and yaw rate measurements have already been preprocessed with an appropriate motion model into a relative movement vector $\boldsymbol{z}^{\mathrm{o}} = [\Delta x, \Delta y, \Delta\theta]^\top$. The vector $\boldsymbol{z}^{\mathrm{g}} = [x^{\mathrm{g}}, y^{\mathrm{g}}, \theta^{\mathrm{g}}]^\top$ denotes the global pose estimate from the GNSS receiver. The landmark detections are stored in the vector $\boldsymbol{z}^{\mathrm{l}} = \{[x^{\mathrm{l}}_k, y^{\mathrm{l}}_k]^\top\}_{k=1}^{\mathcal{K}}$. We compare them against the landmark map $\boldsymbol{m} = \{[x^{\mathrm{m}}_i, y^{\mathrm{m}}_i]^\top\}$, which is in world coordinates. As each landmark is reduced to two coordinates, the map can be stored efficiently. Its size remains small even for large environments compared to typical feature or point cloud methods.

The goal in vehicle localization is to estimate the most likely vehicle pose $\boldsymbol{x}^*_t$ given the measurements and the map. Formally, we seek to compute

$$\boldsymbol{x}^*_t = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}_t | \boldsymbol{z}_t, \boldsymbol{m}). \tag{1}$$

While both estimation algorithms, the particle filter and the graph-based localization, contribute to this goal, they employ different methods to achieve it.

### A. Particle filter

The particle filter is a nonparametric Bayesian filter that represents its posterior distribution via a set of hypotheses. Each of these hypotheses is called a particle and consists of a weighted estimate of the vehicle's location and orientation. Therefore, the state is represented as a set of $\mathcal{M}$ particles $\mathcal{S} = \{\langle \boldsymbol{x}^i, \omega^i \rangle\}_{i=1}^{\mathcal{M}}$, where each particle has a weight $\omega^i$. The sum of all weights is equal to one. This set of weighted samples can represent arbitrary probability distributions given enough particles. We sample $\mathcal{S}_0$ initially from a large Gaussian distribution around the pose estimate $\boldsymbol{z}^{\mathrm{g}}$ computed by low-cost GNSS receiver. Subsequently, the particle filter algorithm estimates $\mathcal{S}_t$ at time $t$ based on its previous estimate $\mathcal{S}_{t-1}$. The main steps of a particle filter are motion update, importance weighting, and resampling.

We refer to Thrun et al. [21] for a detailed derivation of these steps. Important for our real-world application is that we use $1\,\%$ of the available particles to add recovery particles sampled around the GNSS pose estimate. This is, for example, useful when we drive for some time in unmapped areas and reenter mapped areas.

An important design question is when to resample. The naive approach to resample in every step can lead to issues regarding state diversity and accidentally dropping good hypotheses. One common idea to counteract this is to decide when to resample based on the criterion of *effective number of particles* [22]. While this yields reasonable results, we have observed even better results with taking into account

the history of the particle weights. For this, we compute with

$$\omega_t^i = \alpha \cdot p(\boldsymbol{z}_t^{\mathrm{l}} | \boldsymbol{x}_t^i, \boldsymbol{m}) + (1 - \alpha) \cdot \omega_{t-1}^i \tag{2}$$

the exponential moving average of the particle weight. The idea is to prevent low weights for good particles at the current time step due to false positive detections, if they have obtained high weights in the past.

To obtain a single vehicle pose as output of our particle filter, we fit a Gaussian distribution to the particle cloud, i.e., we take the weighted mean of the particle poses. Other choices, such as taking the particle with the maximum weight or clustering the particle set, are also possible.

### B. Sliding window graph-based localization

The optimization-based localization can be represented as a factor graph. While measurements $z_i$ and their corresponding covariance matrices $\boldsymbol{\Sigma}_i$ can be seen as factors, state variables are denoted as nodes similar to e.g. [23]. In our approach the state vector $\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{p}} & \boldsymbol{x}^{\mathrm{l}} \end{bmatrix}$ is a sliding window with length $n$ of vehicle poses $\boldsymbol{x}_{t-n+1:t}^{\mathrm{p}}$. Additionally it contains all landmark estimates $\boldsymbol{x}^{\mathrm{l}}$. Although including landmarks into the estimation is unusual in common pose graph localization approaches, the benefits of our approach are easy visual inspection and predominantly the ability to refine landmark maps. As updating a map in the context of automated driving may affect safety requirements, we store the updates separately for later inspection. The update process itself is not part of this paper as we focus on pure online localization.

Assuming Gaussian distributions and i.i.d. measurements, we represent (1) as a weighted least squares problem similar to [24]. We omit the detailed description of our error functions here and refer to our related work [20], [19]. We include knowledge about our map by matching detected landmarks against the map. The state of every matched landmark is globally constrained by the error function

$$e^{\mathrm{m}}(\boldsymbol{x}^{\mathrm{l}}, m_k) = \boldsymbol{x}^{\mathrm{l}} - m_k, \tag{3}$$

with map landmark position $m_k$. By globally constraining the landmarks we implicitly constrain the vehicle poses inside the sliding window. This allows us to globally localize the vehicle even if no GNSS is available.

The weighted least squares problem is nonlinear and is solved iteratively. A common technique is to use the Gauss-Newton or Levenberg-Marquardt algorithms, as explained by Grisetti et al. [24]. In contrast to the particle filter, this optimization approach searches for a single optimized estimate rather than maintaining a set of weighted hypotheses.

### C. Resource-adaptive localization

We turn our attention to the question what size of state vector we can allow ourselves to choose. In case of the particle filter, this is the number of particles that we choose to represent the posterior. For the graph-based localization, it is the length of the sliding window. We cannot set these values arbitrarily large as we have finite computational resources and need to deliver pose estimates at a fixed frequency $f$. If the available CPU time changes over time we still want to fully exploit the remaining CPU time to achieve the best localization accuracy. Our solution is to use a proportional-integral-derivative (PID) controller from linear control theory as e.g., applied by Li and Nahrstedt [25] in a more general domain. The specific controller used in this paper is similar to the one described by Merfels and Stachnis [19]. For simplification we assume linear runtime complexity in the size of the state vector for both our algorithms. For the particle filter, this is exactly true, and for the graph-based localization, this is approximately true which we show empirically in our experiments.

### D. Deterministic vs. stochastic algorithm

Determinism is the property of an algorithm to always produce the same output given a fixed input. In the context of localization, this means that deterministic algorithms compute the same trajectories if they are fed with the same stream of input data. Stochastic algorithms are non-deterministic as they involve some form of randomness.

Particle filter localization is a stochastic algorithm as it is fundamentally based on randomly drawing samples from a proposal distribution. The two steps motion update and importance weighting are usually deterministic, but the resampling step introduces randomness.

Graph-based localization relies on optimizing a set of constraints that are derived from the input data. The conventional probabilistic interpretation of this step is that the maximum likelihood of the set of robot poses is determined. However, this probabilistic notion does not imply that it is a stochastic algorithm. In fact, it is deterministic and produces the same results if it is run repeatedly on the same data.

In the context of automated driving, localization plays a safety-critical role as driving maneuvers are based on the pose of the vehicle relative to the map. Therefore, functional safety requirements need to be fulfilled. This is usually significantly harder—if not impossible—for stochastic than for deterministic algorithms which is why the graph-based approach is preferable.

### E. Unimodal vs. multimodal

Particle filter and graph-based localization differ in how they represent their belief about their estimated poses. The particle filter tracks a nonparametric distribution over time. It can represent multimodal beliefs that are of non-Gaussian nature. This is useful in situations of high ambiguity due to perceptual aliasing or self-similar environments, for example. The algorithm relies on having sufficient and well-distributed samples to represent the belief about the current pose.

In graph-based localization under the conventional probabilistic interpretation, the graph infers the set of most likely poses given the measurements. As a graph is usually built up by multiple poses, this is technically a multimodal Gaussian distribution. However, each pose corresponds to a single Gaussian that is being estimated. Therefore, the expressiveness about the belief of the pose at a given point in

time is higher for particle filters, as they are not restricted to parametric or Gaussian distributions. Depending on the use case and subsequent software modules for which localization is required both approaches are feasible.

### F. Integration of outdated measurements

The integration of out-of-sequence or delayed measurements is of practical importance for most state estimation techniques. Out-of-sequence measurements describe data that comes in the wrong order, i.e., newer data from that input source has already been processed. Delayed measurements refers to data that arrives later than usual and after the state estimation has processed the data for the corresponding time frame. Both types of behavior can be caused by sensors, faulty network transmission, or sensor preprocessing and have a similar effect on the state estimation process.

Particle filters struggle to integrate data from timestamps that have already been processed. They would have to propagate their state back in time and reperform all other steps up the current point in time. As this is not straightforward and computational expensive, delayed and out-of-sequence measurements are usually simply discarded.

Graph-based approaches intrinsically keep some outdated poses as nodes within their graph. As long as the delayed or out-of-sequence measurements fall within the sliding window of the graph, it is therefore straightforward how to integrate this information. It is simply a matter of adding new nodes and edges that reflect these measurements, and the remaining graph structure stays untouched. Therefore graph-based methods are preferable over particle filters when integrating out-of-sequence or delayed measurements.

### G. Estimating old poses

Graph-based approaches maintain a recent history of poses within the state estimation problem while particle filters only maintain the current pose. This allows graph-based approaches to benefit from delayed data association and frequent relinearization. Meaning that the past and present states are simultaneously improved, when e.g., past associations are revised. Additionally, this allows us to output a *lagged pose*, i.e., a pose with a certain delay compared to the current timestamp, that represents the current best estimate of the past. Outputting a lagged pose can therefore be beneficial for applications that can cope with older pose estimates in favor of even higher estimation accuracy.

In comparison, Particle filters never change their past beliefs about the robot poses. They only track the current set of particles and thus lack the option of outputting a lagged pose.

### H. Ease of implementation

For the practictioner topics like ease of implementation, software maintainability, ease of debugging and inspection etc. play an important role. Though, these points are to a certain degree subjective which makes it hard to judge them objectively. Still, these are important topics and we think it is safe to say that it is generally less complex to implement a particle filter than a graph-based system. However, note that many libraries ease the implementation of graph-based systems substantially, e.g., g2o [26] or GTSAM [23].

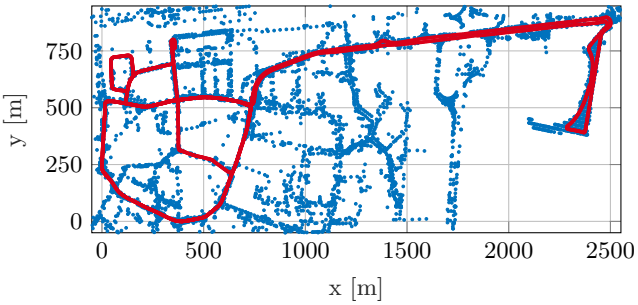



Fig. 2: Map and trajectory of our real-world dataset. All pole-like landmarks in the test area are denoted in blue. The 16 km trajectory is denoted in red. It contains velocities between 0 km/h and 70 km/h and covers typical urban scenarios like heavy and light traffic, different street types, and different junction sizes.

## IV. Experimental Evaluation

Our experiments are designed to serve as a comparison between the particle filter and graph-based localization approaches. Apart from our theoretical discussion in Sec. III, we investigate three key aspects in the context of automated driving: (i) accuracy, (ii) adaptive behavior in terms of computational resources, and (iii) benefit of estimating old poses.

We conduct a set of experiments in which we vary the size of the state vector. In case of the particle filter we directly control the state vector size by setting the number of particles, whereas for the graph-based localization the state vector size is only partly influenced by controlling the number of poses. In the following, we study the effects and focus on accuracy and computation time. Both approaches are required to deliver a pose estimate at a fixed frequency of 20 Hz. The experiments are based on a 16 km long dataset from a real prototype vehicle. The dataset of this evaluation is based on Velodyne VLP-16 to detect pole-like landmarks, wheel-tick and IMU based odometry, and a low-cost GNSS receiver. The trajectory of our dataset is depicted in Fig. 2. Our reference system is a high-frequency and post-processed Real Time Kinematic (RTK) system with a location accuracy specification of 2 cm. We interpolate the reference trajectory to the timestamps of the resulting trajectories to compute the errors.

### A. Accuracy

Our first set of experiments compares the accuracy of both localization approaches. For clarity, Fig. 3 shows the empirical cumulative error distributions (CDF) only for a selected number of experiments. The chosen experiments represent the lower and upper bound of the CDFs for each method. Our experiments show that in terms of accuracy the graph-based localization (GBL) is superior to the particle filter (PF) as the CDF accumulates faster. In both localization approaches it is necessary to maintain at least a minimum number of the states. This can be seen in the experiment

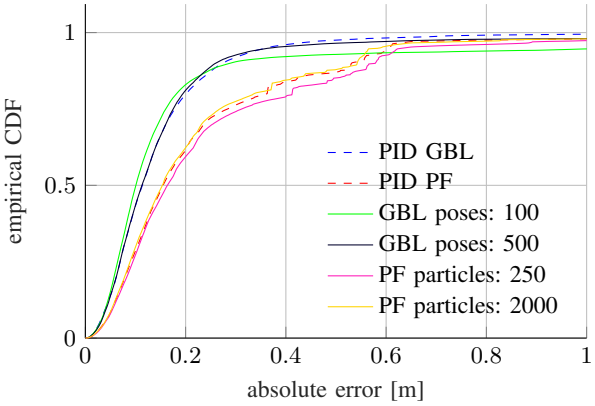



Fig. 3: Empirical cumulative distribution function (CDF) for a set of experiments, comparing particle filter (PF) vs. graph-based localization (GBL).

| | Experiment | abs. | long. | lat. | heading |
|---|---|---|---|---|---|
| GBL | 100 poses | 0.221 m | 0.136 m | 0.143 m | 0.49° |
| | 500 poses | 0.168 m | 0.116 m | 0.096 m | 0.39° |
| | PID (1104) | 0.150 m | 0.111 m | 0.078 m | 0.34° |
| PF | 250 particles | 0.270 m | 0.196 m | 0.139 m | 1.46° |
| | 2000 particles | 0.237 m | 0.183 m | 0.105 m | 1.38° |
| | PID (2160) | 0.240 m | 0.176 m | 0.119 m | 1.42° |

TABLE I: Absolute, longitudinal, lateral, and heading errors for different configurations during a 16 km urban drive. For the PID experiments we denote the average number of particles and poses in the graph in brackets.

with 250 particles and respectively for the graph-based approach with a sliding-window of 100 poses. Both are less accurate than the experiments with the greater state vector size. Additionally, Fig. 3 shows that our resource-adaptive PID controller with a variable state vector size delivers comparable accuracy results.

Tab. I shows the accuracy in terms of average trajectory errors. The errors reflect that the PID controller works in terms of accuracy for both approaches and achieves comparable accuracy.

### *B. Runtime behavior and PID controller*

Our second experiment is designed to show the relation between computation time and the size of the state vectors. Fig. 4 compares the runtime behavior between experiments with a fixed number of particles and poses. The average computation time for each experiment is denoted as a red cross, with the single standard deviation in black lines. Both approaches show a linear relation between computation time and control variable. For the graph-based approach we rely on the highly tuned g2o [26] framework for optimization and apply sparse block matrices together with Cholesky decomposition, which improves the runtime performance.

Additionally, Fig. 4 shows that our PID controller manages to satisfy timing requirements by adjusting the number of particles and number of poses in the graph. In both cases our PID approach successfully provides pose estimates around the set target frequency (setpoint). It exploits the complexity of the problem, which depends on the number of visible landmarks in each timestamp. When only a small number

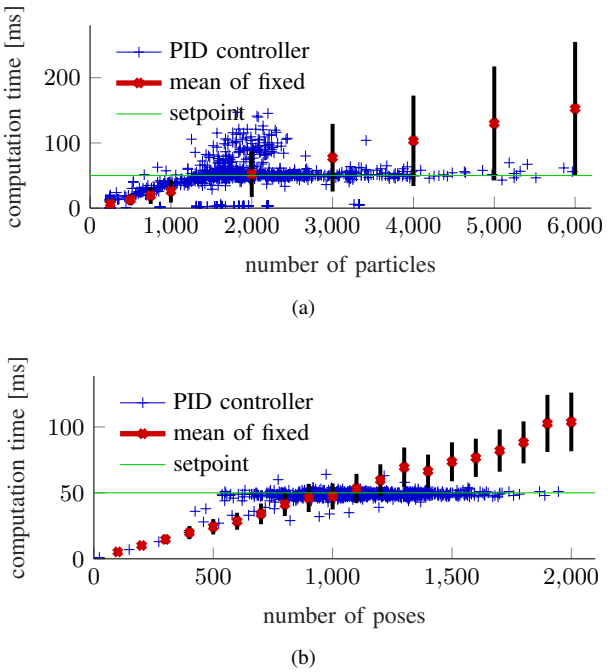



Fig. 4: Relation between computation time and control variable. (a) For the particle filter, we control the number of particles. (b) For graph-based localization, we control the number of poses.

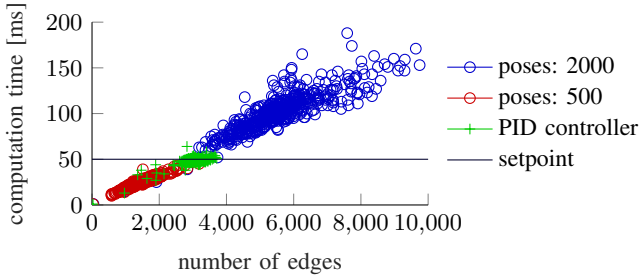



Fig. 5: Relation between computation time and the number of edges in the graph-based localization. The number of edges represents the number of measurements used in the graph.

of landmarks is visible, the complexity decreases and the size of the state vector is increased. We show this behavior in Fig. 5 for the graph-based localization. It shows that the computation time for the experiments with a fixed number of poses depends roughly linearly on the number of edges in the graph. As the number of measurements itself is not constrained, the number of measurements in the graph depends on the environment structure. Due to our sliding window approach, in which all poses have the same temporal distance, the number of poses implicitly controls the number of measurements in the graph. Our PID controller exploits this fact and successfully provides pose estimates with the set target frequency.

### *C. Estimating old poses*

Our third experiment demonstrates the benefit of optimizing past poses in our graph-based approach. In contrast to the particle filter, estimating the trajectory within the sliding window is part of the graph-based optimization. Fig. 6 shows

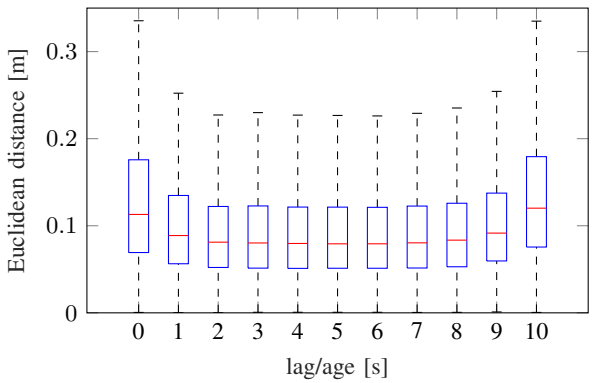



Fig. 6: Accuracy within the trajectory of a graph. A lag of 0 s represents the most recent pose at the head of the graph, whereas a lag of 10 s denotes the oldest pose at the tail of the graph. This plot corresponds to a graph with 500 poses, which are 20 ms apart. The figure shows that the most accurate poses are in the middle of the graph.

box plots for the Euclidean errors within the trajectory of the sliding-window graph. The accuracy in the middle of the graph is better than at the head and tail. The reason for this effect is that the poses in the middle of the graph are more constrained than the poses at the ends of the graph. The effect at the tail of the graph is caused by using truncation instead of marginalization. Using lagged poses is especially beneficial for non-timing-critical applications, for which a more accurate but lagged pose is helpful.

## V. Conclusion

In this paper we provided an argumentative and experimental comparison between particle filter and graph-based localization for automated vehicles. On the one hand, we argued that particle filters have the advantage of being able to represent multimodal distributions and are generally easier to implement. On the other hand, graph-based approaches as non-stochastic algorithms can fulfill functional safety requirements more easily. Moreover, their sliding window of the past allows them to easily integrate outdated measurements, perform delayed data association, and optionally output a lagged pose. We have experimentally shown that the graph-based localization achieves a higher accuracy than the particle filter, that outputting a lagged pose allows us to trade estimation age versus accuracy, and that a PID controller is able for both approaches to regulate the computation time to not exceed a predefined setpoint.

Depending on the importance of these criteria, practitioners can choose the best algorithm for their application. In general we recommend using graph-based approaches as they have proven to be of higher accuracy, which is usually the most important property for localization. Most SLAM approaches have switched from adopting the particle filter to the optimization paradigm, and we believe that this is similarly beneficial for many localization systems.

## References


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

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *J. Artif. Intell.*, vol. 128, no. 1-2, pp. 99–141, 2001.

[3] C. Stachniss and W. Burgard, "Particle filters for robot navigation," *Foundations and Trends in Robotics*, vol. 3, no. 4, pp. 211–282, 2014.

[4] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[5] C. Stachniss, J. J. Leonard, and S. Thrun, *Springer Handbook of Robotics*. Springer, 2016, ch. Simultaneous Localization and Mapping, pp. 1153–1175.

[6] M. Harr, J. Janosovits, C. Stiller, and S. Wirges, "Fast and robust vehicle pose estimation by optimizing multiple pose graphs," in *fusion*, 2018.

[7] C. Wu, T. A. Huang, M. Muffert, T. Schwarz, and J. Graeter, "Precise pose graph localization with sparse point and lane features," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2017.

[8] C. Cadena, L. Carlone, H. Carillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robot. & Automation Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[10] M. Lundgren, E. Stenborg, L. Svensson, and L. Hammarstrand, "Vehicle self-localization using off-the-shelf sensors and a detailed map," in *IEEE Intell. Vehicles Symp.*, 2014, pp. 522–528.

[11] H. Deusch, J. Wiest, S. Reuter, D. Nuss, M. Fritzsche, and K. Dietmayer, "Multi-sensor self-localization based on maximally stable extremal regions," in *IEEE Intell. Vehicles Symp.*, 2014, pp. 555–560.

[12] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016, pp. 2161–2166.

[13] C. Brenner, "Global localization of vehicles using local pole patterns," in *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, 2009, vol. 5748.

[14] A. Schlichting and C. Brenner, "Genauigkeitsuntersuchung zur Lokalisierung von Fahrzeugen mittels Automotive-Laserscannern," *DGPF Tagungsband*, no. 23, 2014.

[15] A. Schindler, "Vehicle self-localization with high-precision digital maps," Ph.D. dissertation, University of Passau, 2013.

[16] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Proc. Robotics: Science and Syst. Conf. (RSS)*, 2007.

[17] M. Stess, "Ein Verfahren zur Kartierung und präzisen Lokalisierung mit klassifizierten Umgebungscharakteristiken der Straßeninfrastruktur für selbstfahrende Kraftfahrzeuge," Ph.D. dissertation, Gottfried Wilhelm Leibniz Universität Hannover, 2017.

[18] C. Merfels and C. Stachniss, "Pose fusion with chain pose graphs for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016, pp. 3116–3123.

[19] ——, "Sensor fusion for self-localization of automated vehicles," *J. Photogr. Remote Sensing & Geoinf. Sci.*, vol. 85, no. 2, pp. 113–126, 2017.

[20] D. Wilbers, L. Rumberg, and C. Stachniss, "Approximating marginalization with sparse global priors for sliding window SLAM-graphs," in *Proc. IEEE Int. Conf. Robotic Computing (IRC)*, 2019.

[21] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 1st ed. MIT Press, 2005.

[22] J. S. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Stat. & Comp.*, vol. 6, no. 2, pp. 113–119, 1996.

[23] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[24] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, pp. 31–43, 2010.

[25] B. Li and K. Nahrstedt, "A control theoretical model for quality of service adaptations," in *Proc. Int. Workshop on Quality of Service*, 1998.

[26] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.