



# Fast and accurate localization and mapping method for self-driving vehicles based on a modified clustering particle filter

Anas Charroud<sup>1</sup> · Karim El Moutaouakil<sup>2</sup> · Ali Yahyaouy<sup>3</sup>

Received: 5 May 2022 / Revised: 16 August 2022 / Accepted: 25 October 2022 /  
Published online: 14 November 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

Self-driving systems require the creation of perception, which means that they must learn to interact with their environment, gather information and perform tasks while driving. Localization and mapping are essential concepts that any autonomous vehicle must perceive, as they provide information about the location and distance between objects. The absence of GPS, for example, in tunnels and urban canyons, makes it necessary to develop robust methods based on other vehicle equipment, such as Lidar, cameras, IMU, or to combine them all. Based on Lidar measurements, our work presents an architecture consisting of two main phases of mapping and localization; The mapping phase creates a global map of the environment based on non-semantic features using a fuzzy c-means algorithm instead of a semantic algorithm as it was done in some state-of-the-art work. In addition, the remaining clusters were filtered using the DBSCAN algorithm. The localization phase adopted here followed the particle filter architecture; motion update, measurement update, and resampling to estimate the positions. The main contribution of this work is the novel extension of selecting particles to reduce computational time and maintain long-term localization reliability. We have exhibited a method to select relevant particles after motion updates, based on two approaches: clustering with the k-means algorithm and the sigma points algorithm, which were thoroughly examined on short sequences of the Kitti dataset to discover the best one. The selected approach thoroughly tested on the Pandaset dataset. In addition, we tested our method on a long sequence dataset and compared it with the most recent methods. The analysis performed demonstrated the speed of our method and its ability to capture the features needed for real-time localization. Furthermore, it outperformed the well-known localization methods.

**Keywords** Localization · Mapping · Particle filter · SLAM · Features extraction · Clusteringz

---

✉ Anas Charroud  
anas.charroud@usmba.ac.ma

## 1 Introduction

With a progression of successful testing and developing speculation. Autonomous vehicles (AVs) are a step toward revolutionary technology that promotes road safety. Many scientists and specialists agree that autonomous vehicles are ready to move beyond the experimental phase [42]. Global organizations such as Tesla and Waymo have invested in autonomous vehicle research, with some preliminary large-scale projects around the world. AVs assist in controlling and managing the traffics by the Traffic Flow Prediction (TFP) which exerts a favourable impact on solving those issues [5]. AVs aim to reduce injuries and fatalities on the road by eliminating human error, but this is difficult to achieve because the AVs must know the area in which they are working [42].

AVs require robust localization and mapping to ensure the consistency and reliability of the tasks. A centimeter error of localization can perturb several AVs tasks including path planning, avoiding obstacles, lane following, and time of overtaking [38].

A Global Navigation Satellite System (GNSS) is used to locate the vehicle among three or more satellite signals. However, Global Positioning System (GPS) can have reliability problems due to visibility in tunnels, under bridges, in urban canyons and dense tree canopies, multipath reflections and signal latency. As a result, on most roads, we will not be able to obtain sufficient information to locate the AVs vehicle. This also means that more sophisticated techniques are required to achieve sufficient localisation accuracy in difficult terrains [13, 19, 27].

The Inertial Navigation System (INS) is a key component of most guidance, navigation and control systems, particularly those which must operate in the absence of GNSS signals. In this context, an Inertial Measurement Unit (IMU) is the most important component of the system. The IMU consists of accelerometers and heading gyros whose outputs are integrated by a suitable algorithm to provide information on the position and orientation of the vehicle. However, this system suffers from cumulative errors caused by the double integration of accelerometers and heading gyroscopes. Like INS, wheel-odometry has been a reliable method for measuring the speed of production vehicles for many years due to its easy integration, high reliability and low cost. However, wheel-odometry is prone to errors, especially when the sensors are biased, i.e. when the average is not zero (causing a drift effect) or when the variance varies over time (causing a diffusion effect). [17, 34]. Several machine learning algorithms were proposed to correct the measurement information provided by the sensors with the GPS measurements [28].

Light Detection and Ranging (Lidar) and camera sensors can separately find the transformation (translation and rotation) that took place in each segment of the trajectory based on the so-called registration. Registration involves two main steps, namely feature extraction and feature matching. Features are typically detected from laser scans as 3D points, or as 2D points from images. Feature matching then determines the similarities between the extracted features using similarity measures (e.g. Euclidean distances for points, chi-square test for histograms). After determining the transformation, the vehicle position can be deduced easily by this formulation (1).

$$X_t = R * X_{t-1} + T \quad (1)$$

$X_t$  is the position of the vehicle at time  $t$ ,  $T$  is the translation vector, and  $R$  is the rotation matrix.

Iterative Closest Point (ICP) registration is one of the most popular and successful approaches to local pose estimation [14]. It is the process of registering two point clouds

by finding the transformations that best align them. The ICP algorithm proposes an optimization that determines the degree of alignment of the last/current scan and calculates the rigid registration that minimizes the sum of squared Euclidean distances between two sets of corresponding 3D points; see (2).

$$\arg \min_{R,T} \left\{ \frac{1}{M} \sum_{j=1}^M \|a_j - (Rb_j + T)\|_2 \right\} \quad (2)$$

$\{a_j\}_j, \{b_j\}_j$  are the 3D coordinates points of two scans  $A$  and  $B$ .

Despite its advantages, the ICP is sensitive to outliers and only works well when the two point clouds overlap [6]. It can be slow on large 3D scans because the transformation has to be calculated by iterating over each pair of points [42]. Due to measurement errors, the ICP algorithm also fails to achieve good long-term matching accuracy, especially for non-linear trajectories [6]. To solve this problem, the authors of [2] proposed the Normal Distribution Transform (NDT), which encodes each voxel with a probability distribution instead of a single value. The proposed approach addresses two problems in the traditional 3D registration framework. First, it discretised the set of points cloud into voxels using the NDT feature representation. Second, the authors match the statistical (i.e mean and variance) properties between two distributions (voxels) as a matching feature and find the correspondence based on the standard distance metrics of probability theory.

The surveyed methods either cost a lot of time, or they lack reliability, robustness, or accuracy. In contrast to all these methods that are based on a single sensor. In this work, we are mainly interested in sensors fusion, i.e., examining more than one source of information to localize the vehicle. The particle filter (PF) [16] is one of the methods that offer the possibility of processing multiple sources, i.e., measurement sensors such as cameras, radar or lidar, and motion sensors such as IMU or wheel odometry. Measurement sensors provide a description image of the environment, including the shape of the object and the distance between the reference vehicle and the objects. Motion sensors provide the translation and rotation performed between a time segment. Therefore, the particle filter, and more generally the Bayesian filter, offers the advantage of using the translation and rotation provided by the IMU and correcting the error by examining the difference between the real distance of the vehicle to objects and the one we have obtained from the measurement sensors. A particle filter is a form of Bayesian filtering. It has found applications in areas such as robotics, computer vision, and signal processing [36]. The particle filter extends the Kalman filter by including additional states for particles that are not associated with a particular measurement, which provides the flexibility to work with nonlinear trajectories and also to work without assumptions about the probability distribution, such as Gaussianity in the Kalman filter family [36]. The PF algorithm iteratively performs two steps: updating the motion and updating the measurements. After uniformly generating a number of particles, the motion update calculates the translation and rotation found from the motion sensors for all the particles. Next, the distance between the global map features and the particles is calculated in order to be compared with the information from the local features. The higher the similarity, the more likely the position state. The position is calculated by taking the average of the closest particles or by directly taking their maximum.

In this paper, we propose a modified clustering particle filter, an extension that speeds up the position computation process by making the matching process fast in the update measurement phase, i.e., instead of matching all particles with the feature map, we use only a few prototype particles that can be efficiently distinguished. We addressed the particle selection problem with two approaches: clustering-based and sigma point-based methods,

each of which is examined in detail. To facilitate understanding of the workflow of our method, we have exposed an architecture consisting of mapping and localization. The mapping phase is responsible for extracting features from the Lidar measurements (especially fuzzy c-means clustering and noise filtering with DBSCAN) and creating a global map of the environment. The localization phase is responsible for the input of the IMU and the results of the mapping phase to determine the corrected position.

The remainder of this paper is organized as follows: In the second part, we present related work on localization methods. In the third section, we provide a clear description of the architecture of our framework and the methodologies used to solve the localization task. The fourth section analyzes the performance of the method on two different datasets: Kitti [8] and Pandaset [40]. The last section is a conclusion.

## 2 Related work

In our state of the art, we have found different methods for locating AVs, mainly divided into three approaches: Registration (scan matching), optimization, and probability-based. Registration methods are used to find the best transformation (translation, rotation) between the last and current position of the vehicle. The most interesting work that adopted this approach was the LiDAR odometry and mapping (LOAM) [43] which used scan-to-scan and scan-to-map matching with edges and flat plane feature points. However, scan-to-scan matching consumes more time and energy. LOAM\_Velodyne [18] is a modified version of the LOAM [43] characterized by real-time speed. Another extension of the LOAM [43] method was proposed to reduce the cost of time in scan-to-scan matching. Lightweight and ground-optimized LOAM (LeGO-LOAM) [31] improved the features extraction process by removing noisy points and optimizing ground and non-ground points. A downsampling technique was proposed in A-LOAM [9] in order to reduce the complexity of the scan-to-scan matching. However, neither LeGO-LOAM nor A-LOAM arrived to reduce efficiently the high computational time. Deep learning-based methods take place in LO-Net [20] to reduce the error of the matching process by using the end-to-learning. An extension of LO-Net [20] was given in the same article [20] named LO-Net-M which is a LO-Net underpinned by a mapping stage which gives better results compared to LO-Net. SGLO [32] have registered good results in term of accuracy and fastness using only sparse geometric line and plane features with line-to-line and plane-to-plane data associations in the scan-to-map matching. However, the accuracy of the method depends mainly on the initialization. Recently, there have been many improvements in the scan matching approach. One way to overcome the problems of ICP is to use end-to-end learning as in DeepICP [22], which is an interesting improvement that provides a good registration rate. SDRSAC [1] and DS++ [26] have proposed a method to transform the optimization problem (2) into a semidefinite optimization problem by introducing a symmetric matrix  $A$  describing the matching potential between the points of point pair  $X$  and  $Y = |X|^2$ . The methods transform the classical optimization problem into  $\max_{XY} AY$  with respect to some constraints that depend mainly on the normalization of the data. This technique offers the possibility to estimate the NP-hard global solution in (2). The transformation can be derived according to the correspondence in [11]. Gaussian mixture models (GMM) also have their place here to solve the log problem in (2) by modeling the input data as a maximum likelihood. The advantages of the GMM-based method are its robustness to noise and outliers [10]. The main idea in it is to develop an optimization strategy to optimize the transformation matrix by maximizing

the likelihood [41]. DeepGMR [21] is one of the methods that applied this concept by teaching a deep learning model the similarity between GMM components and points. However, this technique suffers from a number of problems including noisy data and outliers captured from 3D lidar measurement, also some problems may appear due to unbalanced density, partial overlap or scale variation of points between scans that deeply affect the matching process [41].

The second approach to address the localization and mapping problem are least squares (or optimization) techniques that aim to minimize (maximize) a cost function while respecting certain constraints. In this area, bundle adjustment is one of the first methods that allows to simultaneously find the camera position parameter (intrinsic or extrinsic)  $\{C_j\}$  and the corresponding 3D points  $\{X_i\}$ . The method solves this problem :

$$\min \sum_{i=1}^n \sum_{j=1}^m (u_{ij} - \pi(C_j, X_i))^2 \quad (3)$$

$u_{ij}$  is the observation of pixels at node  $i_{th}$  and  $j_{th}$ ,  $\pi(C_j, X_i)$  is the projection operator that projects the 3D coordinate points into 2D coordinates. In general, bundle adjustment (BA) searches for attachments to find the best parameters  $(C_j, X_i)$  that yield the 2D point (pixel) closest to the actual pixel  $u_{ij}$ . The position can be derived from the extrinsic matrix. BALM [21] from the recent methods that used bundle adjustment adapted for the 3D Lidar measurement. The results were concatenated with the LOAM [43] method to deduce the map of the environment. Moreover, BA was used to solve the SLAM problem like in the ORB-SLAM family [24, 25, 33] which used ORB features to locate cameras and create a map of the environment. BA method act as a helper technique for locale matching. A tracking phase was proposed to aid the system in the process of relocation, site recognition, or even frame matching. These feature points are stored in a robust database (DB) architecture called DBoW2 [7]. ORB-SLAM frameworks have been tested on real-world data and have achieved high accuracy in terms of tracking points, loop closure, and frame localization. However, changes in weather and lack of brightness are the main problems for camera-based methods.

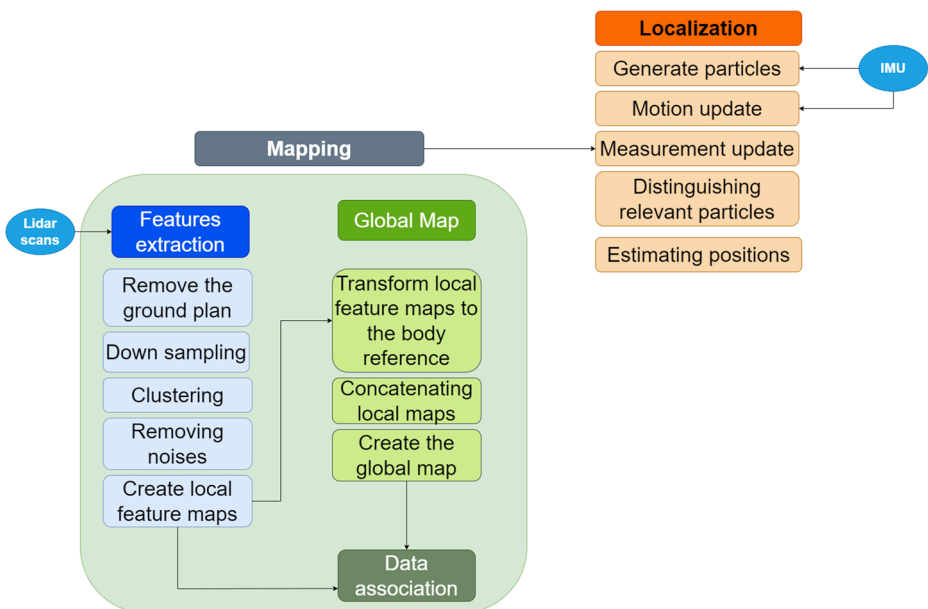
The third option for locating AVs in environments is with the probabilistic perspective. The authors [30] of have used particle filtering to examine the translation and rotation provided by the IMU and find higher candidate poles by looking at the difference in between the features extraction. The authors voxelized the fit point cloud and vertically connected the cells that exceeded a fixed threshold. In addition, a cylinder was applied to all poles to eliminate noise and avoid object overlap. The method achieved good accuracy on the Kitti [8] data set. Kummerle et al. [15] detected poles and walls according to their characteristics and distinctiveness in the environment. Localization was performed using a particle filter, and they obtained a more interesting result. The particle filter was also used in [39] with an interesting algorithm for feature extraction. The authors have investigated the isolation and distinction of pole landmarks in the detection process from the point cloud. The method obtained good results on the Kitti dataset [8]. Schaefer et al. [29] is an interesting work that used probabilistic map features to speed up the process of location determination and meanwhile construct a map of the environment. the location determination was performed based on a modified particle filter. The modification is a small perturbation (multiplication with a generated noise) applied to the predicted state which gives a higher generalization. The method has been tested on the Kitti [8] and NCLT dataset and has shown excellent accuracy, especially on the long-term trajectory. A real-time Monte Carlo localization (RT\_MCL) was proposed in [38] which a trade-off between handling pose estimation accuracy and real-time

performance was addressed. RT\_MCL extracted features as landmarks after fusing Lidar and radar measurements using the unscented Kalman filter (UKF) and used a tailored particle filter to derive the vehicle pose. Data association between the global map (generated offline from the 3D lidar) and the local feature map was performed using ICP matching. The method showed good results in the CARLA driving simulator [35]. However, the method may have some distortion in real scenarios, such as an environment without poles and landmarks. In this paper, we extend the work done in [3] on non-semantic feature map reduction using k-means, global map generation using GMM and localization using particle filters by adding a DBSCAN filter after performing fuzzy c-means feature extraction to remove noisy points and striking lines from the environment. Moreover, we present the modified clustering particle filter which consists of choosing some relevant particles for position computation. We propose two methods to choose those particles and we analyze their potential using several tests in real scenarios.

### 3 Methodology

#### 3.1 System architecture

This section describes the architecture followed to solve the problem of localization and mapping in self-driving vehicles by exploring and manipulating Lidar and IMU measurements. The overall system architecture is illustrated in Fig. 1, which includes two main components: mapping and localization. The mapping phase is responsible for creating a global map of the environment by processing the Lidar scans (or scenes, or measurements).



**Fig. 1** Presentation of the framework used in this work, which consists of two main components, mapping and localization. Mapping is responsible for extracting the features and creating the global map. Localization is responsible for calculating the positions

It consists of two main steps: feature extraction and global mapping. Feature extraction is the process of manipulating the Lidar scans to extract relevant features, remove noise, and create a local feature map for each scan. The global mapping step is set up to create a global map of the environment by converting all local feature maps into a body reference coordinate system and then merging them all. The localization phase then calculates the position of the vehicle based on the vehicle motion provided by the IMU and based on the results of the mapping step. Each of the steps (and stages, components) is explained in more detail below.

### 3.2 Mapping

To create a map of the environment, we use Lidar scans to represent objects in the environment with 3D coordinates, where the accuracy and robustness of these scans to changes in the environment are important. However, Lidar scans contain a huge amount of data points that contain noise and irrelevant information that needs to be removed to create a lightweight map and speed up the localization process. Below, we demonstrate our relevant feature extraction techniques to help create the global map.

- **Feature extraction:** Acts as a pre-processing step, which receives data from the 3D lidar point cloud and transforms it into usable inputs, as the huge number of point data can especially affect performance and consume huge time for execution, which increases the need to reduce it without losing relevant information. We processed each lidar scan data with a series of filters; ground plane removal, downsampling of points, clustering, and removing noises. Let us define a scan of 3D Lidar points by:

$$P = [p_0, p_1, \dots, p_N] \quad (4)$$

Such that

$$p_i = [x_i, y_i, z_i] \quad (5)$$

Each scan  $P$  is processed as follows. First, the 3D Lidar point scans provide an overview of the environment which certainly contains noise and information that we will not examine in our architecture, such as the ground plane. Since the ground plane did not show any hard distinguishable features that could help the matching process we **remove the ground plane** to reduce time. Our algorithm for removing the ground plane is take into consideration the points whose z-axis is above 0.1 of all points within the scan  $P$ . A **down sampling** method was applied to reduce the cost of feature extraction. The points are grouped into voxels (3D cube which is like pixels in images). Then, each occupied voxel generates exactly one point by averaging all the points in it. Finally, we applied a **clustering** phase to the remaining points. We tested several clustering methods to find the best one that meets the criteria of speed and representativeness. According to Table 1, fuzzy c-means [4] has recorded the lowest log time, which means that it takes less time to generate cluster functions. Given its ability to handle stochastic systems, it will be our choice in this work. To remove the noisy points and preserve the local feature map, we used the Density-Based Spatial Clustering of Applications with Noise (DbSCAN) algorithm. We tracked features that only formed lines, where each line must contain a minimum number of points and the distance between them must be small. (An example of the process was illustrated in Fig. 2)

- **Create global Map:** It is a very important step before the main task of localization. It is the process of creating a map of the environment based on the data collected by the sensors (lidar scans). In our case, we transform the local feature maps resulting

**Table 1** Time consumption (mm: ss) to execute the features extraction workflow with the specified clustering methods tested in the sequences 0001 from the Kitti data set [8]

| Clustering method                   | Time cost    |
|-------------------------------------|--------------|
| Growing Neural Gas                  | 01:24        |
| KMeans                              | 01:38        |
| fuzzy c-means                       | <b>00:43</b> |
| Hierarchical clustering             | 01:07        |
| Gaussian mixture model              | 00:49        |
| Self Organizing maps                | 02:06        |
| Agglomerative Clustering            | 01:18        |
| Particle Swarm Optimized Clustering | 00:52        |

from the feature extraction workflow into a **reference coordinate** system that unifies the viewing angle and places each scan on its corresponding timestamp (see Fig. 3a). furthermore, We concatenate the transformed scan in order to create the global map of the trajectory and the environment (see Fig. 3b).

Let we note:

$$k \in [0, M_f] \quad M = \{m_k\} \quad (6)$$

$M_f$  number of feature points in the map,  $m_k$  the feature point at position  $k$ ,  $M$  contain features of the global map. and let also define:

$$k \in [0, N_f] \quad F^t = \{f_k^t\} \quad (7)$$

$N_f$  number of feature points in the lidar scan  $t$ ,  $f_k^t$  the feature point at position  $k$  in the scan  $t$ ,  $F^t$  list of features at scan  $t$  which contain features of the local map at scan  $t$ .

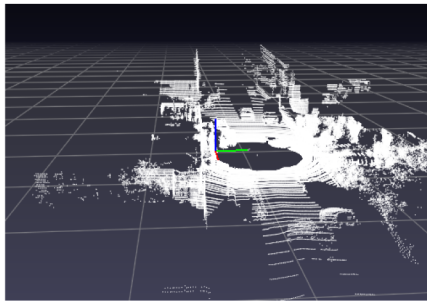
- **Data association:** It is the task of finding a part of the global map that matches the local map at scan  $t$ ; that is, we search for similar points of the local map  $F^t$  (local map at scan  $t$ ) with the corresponding part of the global map  $M_{n(t)}$ . Such that  $n(t)$  is the index of the matching global feature map. Here, a Kd-tree algorithm was used to find the closest points to a local future map. An example of the process is given in Fig. 3c

### 3.3 Localization:

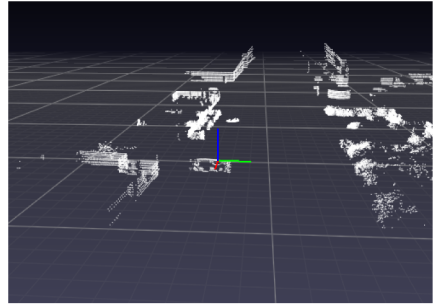
After executing the mapping stage. The map is well prepared for the localization phase. Here present our contribution of the modified clustering particle filter that takes the IMU inputs (translation and rotation) and the results of the mapping stage (global and local maps). For simplicity, it is important to understand what the particle filter is and why we use it, which makes it easier to explain paper contributions in depth. Particle filters are used to correct the IMU sensor error (translation and rotation) by **generating** a series of particles  $X = \{X_{Par_i}\}$  around an initial state of the GPS system representing the candidate positions (Fig. 4a blue dots). Each candidate is associated with a weight  $w_{Par_i}$  which is uniformly initialized. After the AV moves to the next state, the IMU computes the transformation inside it, which is usually error-prone. Each particle followed this transformation (**motion updates** see Fig. 4b)

$$X^t = \{X_{Par_i}^t\}, \quad X_{Par_i}^t = R^t * X_{Par_i}^{t-1} + T^t \quad (8)$$

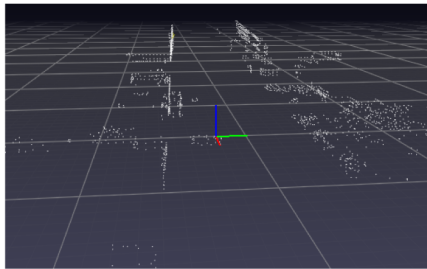




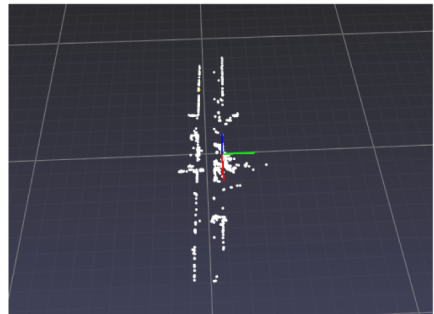
(a) Lidar 3D points scan



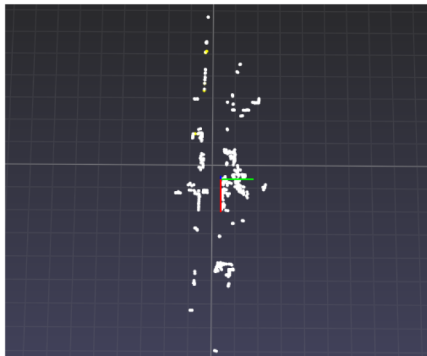
(b) Removing the ground plan Scene



(c) Downsampling process



(d) Features extraction

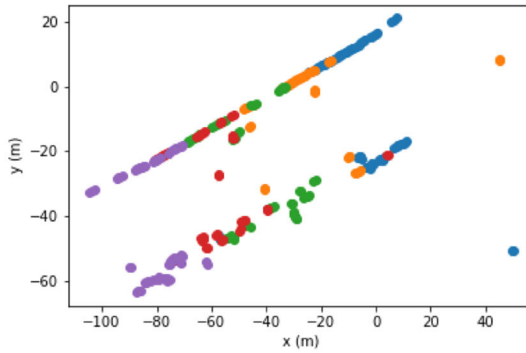


(e) Removing noises

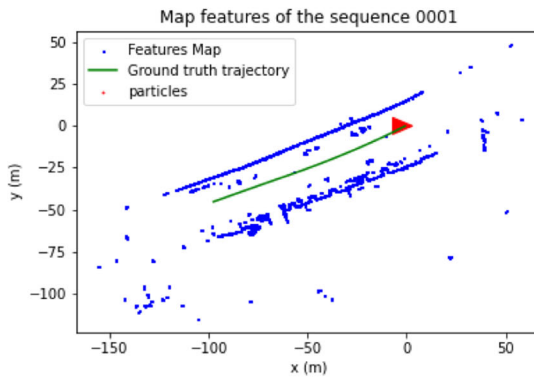
**Fig. 2** Example of the features extraction workflow. Here we took K-means as a clustering algorithm (for image clarity it is recommended to zoom in the image (d))

$X_{Par_i}^t$  is 4 by 4 homogeneous matrix created after the generation of the x,y coordinates and the  $\theta$  orientation uniformly,  $i \in [0, N]$  Such that  $N$  is the number of particles,  $R^t$  and  $T^t$  are, respectively, the rotation and translation matrices at scan  $t$ .

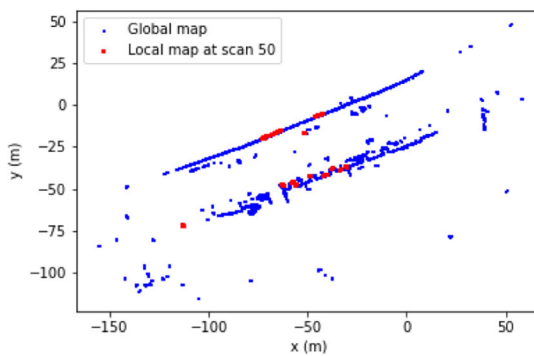
Moreover, the second stage in the particle filtering algorithm is the **measurement update**, which aims to calculate the weights  $\{w_{Par_i}\}$ . These weights play the role of a judge,



(a) Example of transforming and concatenating 5 scans from the Kitti dataset, sequence 0001. Each scan (red, blue, green points) occupied its suitable place that corresponds to each timestamp.



(b) The results of the mapping phase after concatenation and transformation. The blue dots are the feature map, the red triangle is the particle initialization and the green line is the ground truth.



(c) Example of data association of the scan 50 of the sequence 0001 of the Kitti dataset with global map

**Fig. 3** Illustration of the process of creating the global map of the environment

such that a higher weight value means that the associated particle is close to the actual position of the state. According to [29], these weights can be updated by:

$$w_{Par_i}^t = \prod_{p \in [0,t]} p(F^p | X^p, M_{n(p)}) \tag{9}$$

Such that

$$p(F^p | X^p, M_{n(p)}) := \mathcal{N}(\|X^p F^p - M_{n(p)}\|, \sigma) \tag{10}$$

The product in  $X^p F^p$  gives the distances between the particles and features in scan  $p$ ,  $\sigma$  is an isotropic position uncertainty depending on the reference landmarks.  $n(k)$  is an index of the global features associated with the  $k_{rh}$  local feature (provided by kd-tree), and  $\mathcal{N}$  is the normal distribution. weights are like a probability of contribution of corresponding the particle to calculate the real position. The idea is to compare the normal curve of particle distances to features on a global map with what we have on the corresponding local map in each scan. The distances to features in the local map are the actual distances. Consequently, we compare them with those of the particles. The closer the distances are, the greater the contribution to the position calculation. The correspondence between the local and global maps is established by the data association step (see. Fig. 4b for an illustration)

The state estimation at scanning  $t$

$$pos^t \approx \sum_{i=1}^N w_{Par_i}^t X_{Par_i}^t \tag{11}$$

Such that

$$\sum_{i=1}^N w_{Par_i}^t = 1 \tag{12}$$

$N$  is the number of particles.

The accuracy of the method depends mainly on the number of particles, the higher the number of particles, the higher the probability of convergence to the real state. However, increasing the number of particles causes additional energy costs and increases time consumption, which is not recommended for self-driving vehicles [37].

To speed up this process while maintaining accuracy, we proposed to choose efficiently some relevant particles. This step should be done directly after the **motion update** step in order to reduce the computational time of calculating weights and calculating positions also. We suggest two approaches to choose those points which will be thoroughly analysed in the next section.

- **Using clustering approach:** Clustering is a well-known approach to store information in a smaller number of clustering centers. This reduces the time cost and preserves the particle information. Let us define.

$$C^t = \{C_i^t\} \tag{13}$$

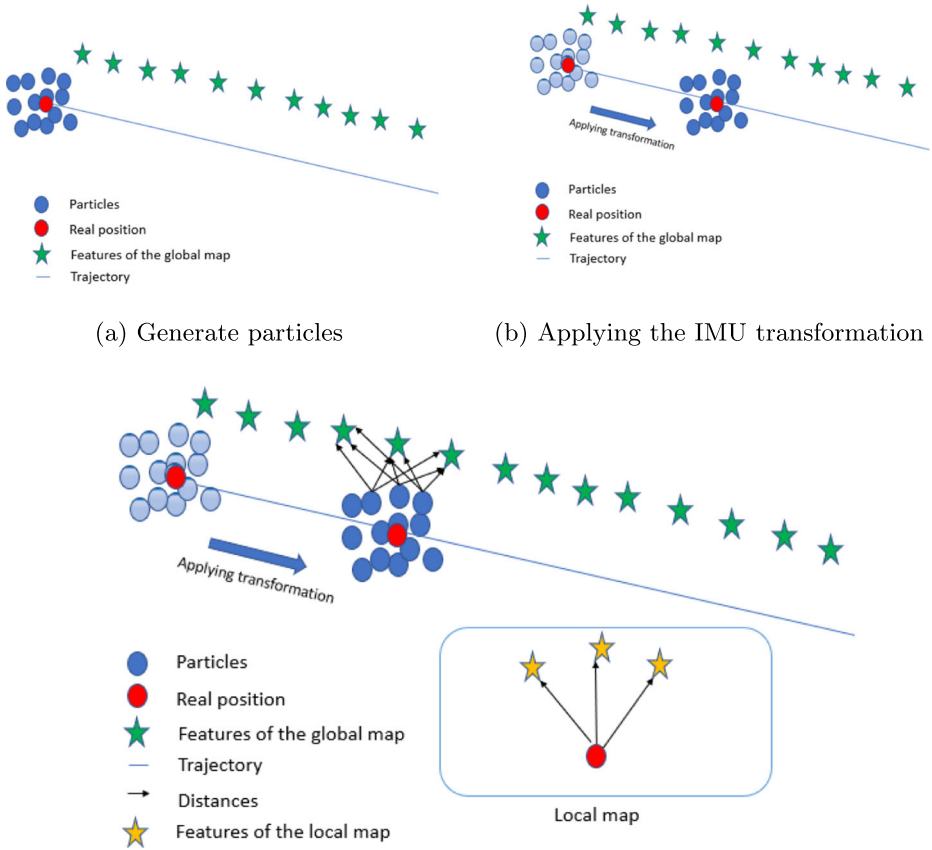
$i \in [0, N_c]$ ,  $N_c$  clusters number,  $C_i^t$  is the center cluster  $i$  at scan  $t$

Consequently,

$$w_{C_i}^t = \prod_{p \in [0,t]} p(F^p | C^p, M_{n(p)}) \tag{14}$$

Such that

$$p(F^p | C^p, M_{n(p)}) := \mathcal{N}(\|C^p F^p - M_{n(p)}\|, \sigma) + \epsilon \tag{15}$$



**Fig. 4** Explication of the workflow of the particle filter (the images are not real. They are generated to explain in depth the process.)

The state estimation at scan  $t$ ,  $\mathcal{N}$  is the normal distribution

$$pos_C^t \approx \sum_{i=1}^{N_c} w_{C_i}^t C_i^t \tag{16}$$

Such that

$$\sum_{i=1}^{N_c} w_{C_i}^t = 1 \tag{17}$$

The K-means algorithm was chosen to perform this task regarding its convergence capability and its ability to incorporate clusters of different shapes.

- **Using sigma points approach:** After the **motion update** step, we calculate the mean and covariance of the particles,  $x$  and  $P$ , respectively.

In order to get those sigma points, we need to find the minimum of these formulas:

$$\min_{(\mathcal{X}, w)} c((\mathcal{X}, w), r, P(\mathbf{x})) \quad \text{subject to } \xi((\mathcal{X}, \mathbf{w}), \mathbf{r}, p(\mathbf{x})) = 0 \tag{18}$$

$\chi$  are the sigma points,  $\omega$  are the weights,  $r$  is the number of sigma points which is equal to  $2L + 1$  and  $L$  is the dimension of  $x$ . In this paper, we chose three different methods: Van der Merwe 2004 dissertation [37], Julier and Jeffery K. Uhlmanns [12], and the simplex method of Phillippe M., and Dominique C. [23] and we merge their results to investigate the benefit of all these methods.

Without diving into the theory of each of them, we present their method to distinguish the sigma points. According to Julier et al. the points can be generated by using these formulas:

$$\begin{aligned} \chi_J[0] &= x \\ \chi_J[i] &= x + \left(\sqrt{(L + \kappa)P}\right)_i \quad i = 1, \dots, L \\ \chi_J[i] &= x - \left(\sqrt{(L + \kappa)P}\right)_i \quad i = L + 1, \dots, 2L \end{aligned} \tag{19}$$

$\chi_J$  is the sigma points generated, and  $\kappa$  is a scaling factor that can reduce high order errors. Note that we need to find the matricial root square weather using the Cholesky method or other. However, regarding its fastness, we choose the Cholesky decomposition.

According to Van der Merwes the points can be generated by using these formulas:

$$\begin{aligned} \chi_M[0] &= x \\ \chi_M[i] &= x + \left(\sqrt{(L + \lambda)P}\right)_i \quad i = 1, \dots, L \\ \chi_M[i] &= x - \left(\sqrt{(L + \lambda)P}\right)_i \quad i = L + 1, \dots, 2L \end{aligned} \tag{20}$$

$\lambda = \alpha^2(L + \kappa) - L$  is a scaling parameter.  $\alpha$  determines the spread of the sigma points around  $x$ , and  $\kappa$  is a scaling factor that can reduce high order errors.

According to Phillippe M. et al the points can be generated by using these formulas:

$$\begin{aligned} \chi_P[i] &= x - I^{(i)} \quad i = 1, \dots, L + 1 \\ I^{(i)} &= \sqrt{P} \tilde{I}_r^{(i)} \end{aligned} \tag{21}$$

Such that the vectors  $I_r^{(i)}$  are the columns of the matrix noted  $[\tilde{I}_r^*]$  recursively defined by

$$\begin{cases} [\tilde{I}_1^*] = \left(-\frac{1}{\sqrt{2\alpha}} \frac{1}{\sqrt{2\alpha}}\right), \quad \alpha = \frac{p}{p+1} \\ [\tilde{I}_d^*] = \begin{pmatrix} & & 0 \\ & [\tilde{I}_{d-1}^{(i)}] & \vdots \\ \frac{1}{\sqrt{\alpha d(d+1)}} & \cdots & \frac{1}{\sqrt{\alpha d(d+1)}} & \frac{-d}{\sqrt{\alpha d(d+1)}} \end{pmatrix}, \quad 2 \leq d \leq p \end{cases} \tag{22}$$

Finally, we concatenate our sigma points.

$$\chi^t = \{[\chi_J, \chi_M, \chi_P]^t\} = \{\chi_i^t\}$$

$i \in [0, N_\chi]$ ,  $N_\chi$  sigma points number,  $\chi_i^t$  sigma point of the element  $i$  at scan  $t$   
Consequently,

$$w_{\chi_i}^t = \prod_{p \in [0, t]} p(F^p \mid \chi^p, M_{n(p)}) \tag{23}$$

Such that

$$p(F^P | \chi^P, M_{n(p)}) := \mathcal{N}(\|\chi^P F^P - M_{n(p)}\|, \sigma) + \epsilon \tag{24}$$

The state estimation at scan  $t$ ,  $\mathcal{N}$  is the normal distribution

$$pos_{\chi}^t \approx \sum_{i=1}^{N_{\chi}} w_{\chi_i}^t \chi_i^t \tag{25}$$

Such that

$$\sum_{i=1}^{N_{\chi}} w_{\chi_i}^t = 1 \tag{26}$$

### 3.4 Evaluation

Our metrics included six error measurement  $\Delta_{pos}$ ,  $\Delta_{lat}$ ,  $\Delta_{lon}$  and  $\Delta_{ang}$  denote the mean absolute positional, latitudinal, longitudinal, and heading errors, while  $RMSE_{pos}$ ,  $RMSE_{ang}$  represents the corresponding root mean squared errors.

$$\Delta_{lat} = \frac{\sum_{i=1}^n |X_i - X'_i|}{n} \quad \Delta_{lon} = \frac{\sum_{i=1}^n |Y_i - Y'_i|}{n}$$

$$\Delta_{pos} = \frac{\sum_{i=1}^n \|pos_i - pos'_i\|_2}{n} \quad \Delta_{ang} = \frac{\sum_{i=1}^n |\theta_i - \theta'_i|}{n}$$

$$RMSE_{pos} = \sqrt{\frac{\sum_{i=1}^n \|pos_i - pos'_i\|_2^2}{n}} \quad RMSE_{ang} = \sqrt{\frac{\sum_{i=1}^n |\theta_i - \theta'_i|^2}{n}}$$

Such that  $n$  is the key frames number,  $pos_i = [X_i, Y_i]$  is the predicted state,  $pos'_i = [X'_i, Y'_i]$  is the actual state,  $\theta$  is the predicted angle,  $\theta'$  actual state

## 4 Results

In this section, we analyze the performance of our method for locating the vehicle on the trajectory in the short, medium, and long term, and investigate which method is more accurate and efficient among the two presented in our methodology; i.e., clustering-based and sigma-point based methods.

### 4.1 Experiments

We have tested our method in two benchmark data sets; Kitti [8], and Pandaset [40]. Kitti data contain several sequences driven around the mid-size city of Karlsruhe, The vehicle was equipped with two high-resolution colour and grey-scale video cameras. Accurate ground truth is provided by a Velodyne laser scanner and a GPS location system. We have chosen to test our method in different categories provided in the data set, including city, residential, Road, Campus, and Person. This advantageous data set enables the test of various environmental changes and investigate the performance of the vehicles in short and long-term localization trajectories. In addition, we examined the performance of our method on a second data set that included a complex showcase of urban driving scenarios, including steep

**Table 2** Testing the performance of the method with different numbers of cluster particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs

| Number of cluster | 10    | 50     | 100   |
|-------------------|-------|--------|-------|
| RMSE_{pos} (m)    | 0.127 | 0.1479 | 0.552 |
| Time cost (s)     | 01:25 | 03:53  | 06:42 |

hills, construction, heavy traffic and pedestrians, and a variety of weather and light conditions in the morning, and afternoon, dusk, and evening. Pandaset vehicle was equipped also with IMU/GPS information and Lidar, cameras measurement. The tests were conducted in an i7 8th generation pc with 20 GB Ram. However, a GPU is recommended for this task. We have fixed some configuration in the features extraction process, including the voxel size ( $0.5 * 0.5 * 0.5$ ) per voxel, the number of cluster of 100 clusters (This choice depend mainly on the execution cost and the representation of features in the environment) and we keep as defaults other fuzzy c-means parameters. For the DBSCAN algorithm, we have fixed 2 points as a minimum sample to fit the lines and 1 m unit as the maximum distance between two samples for one to be considered as in the neighbourhood of the other which is very important parameter to choose accurately. In the localization stage, we have fixed a 10 cluster for the localization based on k-means clustering and we generate 100 particles to run the localization tests for both approaches (clustering or sigma points). The justification of this choice will be discussed in the next subsection. In the sigma localization method, we set  $\kappa=0$  which gives the standard unscented filter,  $L = 3$ ,  $\alpha = .00001$ . We have run the Mapping stage for one time to create the local and global maps.

## 4.2 Parameters discussion

We have analyzed the appropriate configuration to obtain better results in terms of accuracy and time cost. Table 2 shows that, for a fixed number of 2000 particles, 10 cluster centers recorded better results and represent a practical parameter that serves for accuracy, energy and time cost. Furthermore, Table 3 shows that, for a cluster of 10 fixed centers, we obtained that 100 particles are sufficient to obtain good accuracy in a reasonable time These results justify the choice of our configuration (10-center cluster and 100 particles) and constitute our main contribution to reduce time consumption. However, some perturbation of the accuracy occurs due to the initialization of the particle filter or the reduced number of center clusters.

**Table 3** Testing the performance of the method with different numbers of particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs

| Number of Particles | 100   | 1000   | 2000   |
|---------------------|-------|--------|--------|
| RMSE_{pos} (m)      | 0.206 | 0.3704 | 0.1802 |
| Time cost (s)       | 00:23 | 00:55  | 02:38  |

**Table 4** Accuracy and time consumption of the Localization based on the clustering in the Kittı dataset from different categories sequences

| Category   | Seq  | frames no | $t_{Feat}$ (s) | $t_{loc}$ (s) | $\Delta_{pos}$ (m) | $\Delta_{lat}$ (m) | $\Delta_{lon}$ (m) | $\Delta_{ang}$ (°) | $RMSE_{pos}$ (m) | $RMSE_{ang}$ (°) |
|------------|------|-----------|----------------|---------------|--------------------|--------------------|--------------------|--------------------|------------------|------------------|
| City       | 0001 | 108       | 00:02          | 00:06         | 0.08               | 0.01               | 0.07               | 0.009              | 0.11             | 0.01             |
| Residence  | 0035 | 131       | 00:04          | 00:10         | 0.06               | 0.01               | 0.05               | 0.04               | 0.08             | 0.06             |
| Road       | 0027 | 188       | 00:07          | 00:16         | 0.10               | 0.031              | 0.09               | 0.005              | 0.005            | 0.0037           |
| Campus     | 0034 | 49        | 00:03          | 00:04         | 0.0063             | 0.003              | 0.004              | 0.005              | 0.008            | 0.005            |
| Person     | 0053 | 68        | 00:02          | 00:05         | 0.0009             | 0.0007             | 0.0005             | 0.002              | 0.001            | 0.002            |
| <b>SUM</b> |      | 544       | 00:18          | 00:41         | 0.2472             | 0.0547             | 0.2145             | <b>0.061</b>       | 0.204            | <b>0.0807</b>    |

### 4.3 Short-term localization discussion

The first type of experiment was conducted to investigate the performance of our two models (i.e., based on clustering techniques, or based on sigma points) and to test the extent to which each is able to accurately locate the vehicle without wasting too much time and energy. Therefore, we have distinguished a number of short Kittı and Pandaset dataset sequences that contain several environmental changes and different trajectory shapes. The results in Tables 4 and 5 show the accuracy of the two models in locating the vehicle within the chosen sequences and the time consumed in each. It was shown that our models are able to accurately determine the positions in a short computation time; **41 s** to locate the vehicle in 544 frames (105 s across all sequences) with **0.24 m** average absolute positioning error, demonstrating the effectiveness of using clustering techniques to reduce the time required to calculate the positions. Meanwhile, the sigma point based localization used **5 s** to locate the vehicle in 544 frames (105 s across all sequences) with **0.16 m** in the average absolute positioning error which is super fast and accurate, especially for real-time applications. Based on the mean and covariance of the particles after the motion update, the sigma points method manages to perfectly choose the “winning” points that efficiently contribute to the calculation of the positions. According to the analysis performed above, the localization based on sigma points outperforms the localization based on clustering both in terms of accuracy and speed. Therefore, it will be our choice in the remainder of this review.

Table 6 confirm our statement; a mean absolute positioning error of 0,18 m was obtained across all sequences in the pandaset [40] reference data set over 400 frames in a 5(s) of

**Table 5** Accuracy and time consumption of the Localization based on the sigma points in the Kittı dataset from different categories sequences

| Category   | Seq  | frames no | $t_{Feat}$ (s) | $t_{loc}$ (s) | $\Delta_{pos}$ (m) | $\Delta_{lat}$ (m) | $\Delta_{lon}$ (m) | $\Delta_{ang}$ (°) | $RMSE_{pos}$ (m) | $RMSE_{ang}$ (°) |
|------------|------|-----------|----------------|---------------|--------------------|--------------------|--------------------|--------------------|------------------|------------------|
| City       | 0001 | 108       | 00:02          | 00:01         | 0.054              | 0.013              | 0.05               | 0.01               | 0.067            | 0.011            |
| Residence  | 0035 | 131       | 00:04          | 00:01         | 0.01               | 0.004              | 0.015              | 0.048              | 0.025            | 0.088            |
| Road       | 0027 | 188       | 00:07          | 00:01         | 0.084              | 0.019              | 0.07               | 0.005              | 0.098            | 0.006            |
| Campus     | 0034 | 49        | 00:03          | 00:01         | 0.0064             | 0.004              | 0.005              | 0.005              | 0.0097           | 0.005            |
| Person     | 0053 | 68        | 00:02          | 00:01         | 0.00087            | 0.0007             | 0.0004             | 0.0012             | 0.001            | 0.0015           |
| <b>SUM</b> |      | 544       | 00:18          | 00:05         | <b>0.15527</b>     | 0.0407             | <b>0.1404</b>      | 0.0692             | <b>0.2007</b>    | 0.1115           |



**Table 6** Accuracy and time consumption of the Localization based on the sigma points in the Pandaset dataset from various sequences

| Seq | frames no | $t_{Feat}$ (s) | $t_{Loc}$ (s) | $\Delta_{pos}(m)$ | $\Delta_{lat}(m)$ | $\Delta_{lon}(m)$ | $RMSE_{pos}(m)$ |
|-----|-----------|----------------|---------------|-------------------|-------------------|-------------------|-----------------|
| 100 | 80        | 00:03          | 00:01         | 0.269             | 0.056             | 0.252             | 0.331           |
| 109 | 80        | 00:03          | 00:01         | 0.207             | 0.041             | 0.195             | 0.267           |
| 117 | 80        | 00:04          | 00:01         | 0.237             | 0.043             | 0.227             | 0.295           |
| 139 | 80        | 00:02          | 00:01         | 0.0825            | 0.048             | 0.063             | 0.107           |
| 158 | 80        | 00:03          | 00:01         | 0.149             | 0.05              | 0.134             | 0.193           |

$t_{Feat}$ : Is the cost of time to calculate the features extraction of all the sequences

$t_{Loc}$ : Is the cost of time to execute the Localization process

execution time which demonstrates again the robustness and efficiency of our method to localize the vehicle in different environmental scenarios.

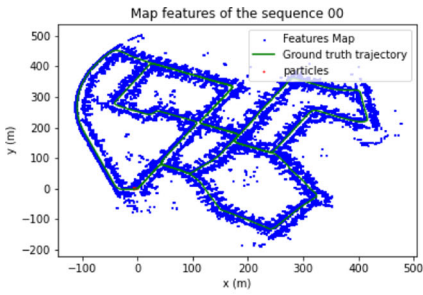
We manually tested Scheafer's method on sequence 0001 (city category) of the Kitti data set and obtained 0.06 m absolute pose error and 0.1 *degree* absolute rotation error. The run took 19 s, which was more time and energy than what we obtained in Table 7. 0.054 m absolute pose error, 0.01 *degree* rotation error in 1 s of execution. We also see in Table 7 that we have such excellent rotation and translation errors that we outperform all competing methods, which is due to the robust technique that distinguishes the sigma points from the particles based on the mean and covariance. Scheafer et al. also record a small absolute pose error with respect to the data association followed in that work. Extracting pole features, as done in Scheafer et al. also significantly helps to achieve these results. However, poles, such as trees, or even the use of walls, do not exist in all environmental settings, which especially affects the localization process. Non-semantic feature extraction guarantees the existence of a certain number of features that will represent the data, which ensures the continuity of the data association workflow. Moreover, this extended work outperforms the one of Charroud. A et al. [3] in term of accuracy and robustness where they have used Gaussian Mixture Models (GMM) to find the global map which affects the data association process. Even the fastness of the process, GMM with a fixed number of clusters misleads the matching process. In contrast, we have concatenated all the features received from the local map to ensure the reliability of the matching process.

**Table 7** Evaluation of the accuracy of the strategies of Sefati et al. and A. Schaefer and of our methodology. The results of Weng et al. and Kümmerle et al. are not directly comparable and are presented for qualitative analysis only

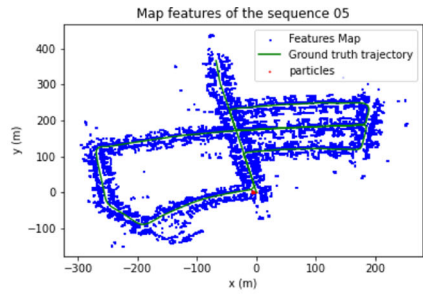
| Methods                 | $\Delta_{pos}(m)$ | $RMSE_{pos}$ | $\Delta_{lat}(m)$ | $\sigma_{lat}(m)$ | $\Delta_{lon}(m)$ | $\sigma_{lon}(m)$ | $\Delta_{ang}(^\circ)$ | $\sigma_{ang}(^\circ)$ | $RMSE_{ang}(^\circ)$ |
|-------------------------|-------------------|--------------|-------------------|-------------------|-------------------|-------------------|------------------------|------------------------|----------------------|
| Kümmerle et al. [15]    | 0.12              | —            | 0.07              | —                 | 0.08              | —                 | 0.33                   | —                      | —                    |
| Weng et al. [39]        | —                 | —            | —                 | 0.082             | —                 | 0.164             | —                      | 0.329                  | —                    |
| Sefati et al. [30]      | —                 | 0.24         | —                 | —                 | —                 | —                 | —                      | —                      | 0.68                 |
| A. Schaefer et al. [29] | 0.096             | 0.111        | 0.061             | 0.075             | <b>0.060</b>      | 0.067             | 0.133                  | 0.188                  | 0.214                |
| Charroud. A et al. [3]  | 0.12              | 0.141        | <b>0.059</b>      | 0.09              | 0.08              | 0.05              | <b>0.043</b>           | 0.078                  | <b>0.057</b>         |
| Ours                    | <b>0.077</b>      | <b>0.085</b> | <b>0.0195</b>     | <b>0.025</b>      | 0.071             | <b>0.059</b>      | <b>0.04</b>            | <b>0.04</b>            | <b>0.05</b>          |

**Table 8** Accuracy evaluation of our method with state-of-art registration methods

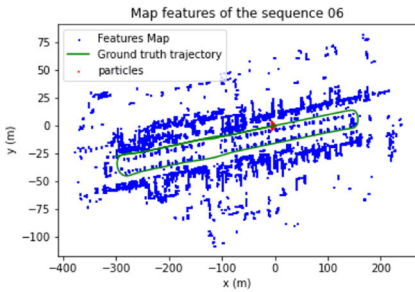
| Seq | LOAM [43] | LOAM Velodyne [18] | LeGO- LOAM [31] | A- LOAM [9]  | SGLO [32]    | LO-Net [20] | LO-Net-M [20] | OURS         |
|-----|-----------|--------------------|-----------------|--------------|--------------|-------------|---------------|--------------|
| 00  | 0.78      | 3.410              | 1.347           | 0.791        | <b>0.775</b> | 1.47        | 0.78          | 0.95         |
| 05  | 0.57      | 1.318              | 0.887           | <b>0.504</b> | 0.596        | 1.04        | 0.62          | 0.73         |
| 06  | 0.65      | 1.007              | 0.87            | 0.615        | 0.483        | 0.71        | 0.55          | <b>0.297</b> |
| 07  | 0.63      | 1.256              | 0.77            | 0.453        | 0.515        | 1.70        | 0.56          | <b>0.391</b> |
| 08  | 1.12      | 2.159              | 1.337           | 1.107        | <b>1.059</b> | 2.12        | 1.08          | 1.16         |
| 09  | 0.77      | 1.437              | 1.282           | 0.739        | 0.733        | 1.37        | 0.77          | <b>0.71</b>  |
| 10  | 0.79      | 1.912              | 1.675           | 1.006        | 0.977        | 1.80        | 0.92          | <b>0.65</b>  |



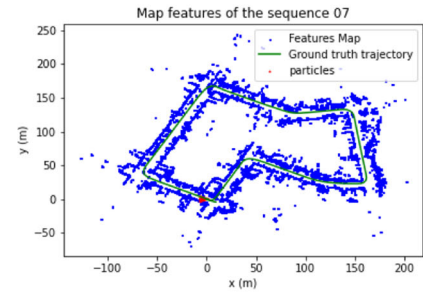
(a) Seq00



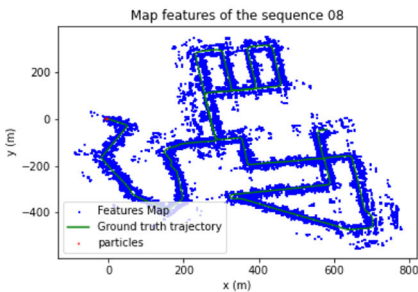
(b) Seq05



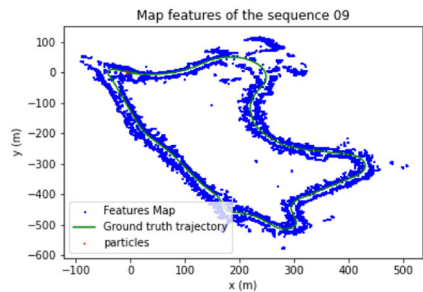
(c) Seq06



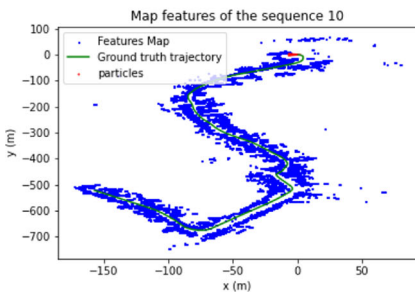
(d) Seq07



(e) Seq08



(f) Seq09



(g) Seq10

Fig. 5 Global maps

## 4.4 Long-term localization discussion

We tested our method on medium and long duration sequences from the Kitti odometry dataset (different from the short sequences in Tables 5 and 7). The method was evaluated on sequence 00 and sequences 05-10 over 17554 frames and 29 minutes of running. According to Table 8, our method produced excellent results; 0.69 m in average absolute positioning error across all sequences. We compared our method with more advanced registration methods, including LOAM\_Velodyne [43] LOAM\_Velodyne [18], LeGO- LOAM [31], A-LOAM [9], SGLO [32], LO-Net [20], LO-Net-M [20], and ours. The metric used by these methods is called  $t_{rel}$ ; the relative translation drift averaged over trajectories from 100 to 800 m. In our case, however, we took the average absolute error over all trajectories. Since the approach used to locate the vehicle differs between us and the registration methods. The analysis was performed approximately to provide a general inspection of the performance of the methods. Table 8 shows that our method outperforms the other methods in 4 sequences scenario, including trajectories of long duration. Moreover, we obtained competitive results in other sequences. Our method records a positioning error of 1.16 m in sequence 08, a long trajectory of 08 min :38 s driving. Again, our method proves its speed by performing the localization task in only 22s. On another side, our method was capable to generate accurately the global map of the environment for these long term trajectories which showed the performance of our features extraction workflow (Fig. 5).

## 5 Conclusion

In this paper, we proposed a multi-source localization and mapping method for self-driving vehicles based on non-semantic feature extraction. The fuzzy c-means method was used to find patterns in LiDAR scans. The cluster centers were filtered by a DBSCAN algorithm to remove the noises and were used to create the local map features. In addition, the global map was created by merging local map features and transforming them into a reference coordinate system which facilitates data association. The localization process was carried out using a particle filter with modifications which is a technique to distinguish some relevant features from the generated particles in order to speed up the calculation process and track the trajectory positions efficiently. We have proposed two approaches to distinguish those points; clustering-based and sigma points based approaches. sigma points has registered the best accuracy in a small calculation time which was our choice for the next experiments and comparisons.

Our proposed approach (sigma points) achieved competitive results in terms of accuracy but does so in significantly less time compared to the state-of-the-art methods evaluated on the Kitti and pandaset benchmark dataset; 0.16 m in the average absolute positioning error of all the short-term sequences of the Kitti dataset in 5 s of execution. Also, we obtained 0.18 m in the average absolute positioning error of all the short-term sequences of the Pandaset dataset. We have tested extensively our method in long-term sequences and we obtained 0.69 m of the average absolute positioning error of all sequences, which demonstrates the robustness, reliability, and efficiency of our method. Meanwhile, we have compared our method with recent state-of-the-art works. Indeed, our method outperforms all of them in short-term sequences, especially, the Schaefer et al. method [29] where we registered 0.077 m, while 0.096 m is registered by Schaefer's method. Furthermore, we have obtained competent results in the long-term sequences.

However, some accuracy deviations occurred due to poor initialization of the particle filter or poor choice of cluster and number of particles. Therefore, our future work will mainly focus on finding the best initialization and investigating the parameters to find the best configuration based on optimal control or using machine learning.

**Acknowledgements** The authors thank all those who contributed to this article.

**Data availability statement** The datasets generated during and analysed during the current study are available in the Kitti repository and Pandaset repository: <http://www.cvlibs.net/datasets/kitti/>, <https://pandaset.org/>

## Declarations

**Conflict of interests and Funding** All authors certify that they have no affiliation or involvement with any organization or entity with financial or non-financial interests in the subject matter or material covered by this manuscript.

## References

1. Biber P (2003) The normal distributions transform: a new approach to laser scan matching. *IEEE Int Conf Intell Robot Syst* 3:2743–2748. <https://doi.org/10.1109/iroso.2003.1249285>
2. Biber P (2003) The normal distributions transform: a new approach to laser scan matching. *IEEE Int Conf Intell Robot Syst* 3:2743–2748. <https://doi.org/10.1109/iroso.2003.1249285>
3. Charroud A, Yahyaouy A, El Moutaouakil K, Onyekpe U (2022) Localisation and mapping of self-driving vehicles based on fuzzy K-Means clustering: a Non-Semantic approach 2022 international conference on intelligent systems and computer vision (ISCV). <https://doi.org/10.1109/iscv54655.2022.9806102>
4. Daszykowski M, Walczak B (2009) Density-Based Clustering methods. *Compr Chemom* 2:635–654. <https://doi.org/10.1016/B978-044452701-1.00067-3>
5. Dawen X, Zhang M, Yan X, Bai Y, Zheng Y, Li Y, Li H (2020) A distributed WND-LSTM model on mapreduce for Short-Term traffic flow prediction. *Neural Comput Applic* 33(7):2393–2410. <https://doi.org/10.1007/s00521-020-05076-2>
6. Du S, Xu Y, Wan T, Hu H, Zhang S, Xu G, Zhang X (2017) Robust iterative closest point algorithm based on global reference point for rotation invariant registration. *PLoS One* 12:1–14. <https://doi.org/10.1371/journal.pone.0188039>
7. Gálvez-López D, Tardós JD (2012) Bags of binary words for fast place recognition in image sequences. *IEEE Trans Robot* 28:1188–1197. <https://doi.org/10.1109/TRO.2012.2197158>
8. Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: the KITTI dataset. *Int J Robotics Res Int J Rob Res*, pp 1–6
9. HKUST-Aerial-Robotics (2022) HKUST-Aerial-Robotics/A-LOAM: advanced implementation of loam. Github, Accessed 8 August, <https://github.com/HKUST-Aerial-Robotics/A-LOAM>
10. Huang X, Mei G, Zhang J, Abbas R (2021) A comprehensive survey on point cloud registration, pp 1–17
11. Huang X, Zhang J, Wu Q, Fan L, Yuan C (2018) A coarse-to-fine algorithm for matching and registration in 3d cross-source point clouds. *IEEE Trans Circuits Syst Video Technol* 28:2965–2977. <https://doi.org/10.1109/TCSVT.2017.2730232>
12. Julier SJ (1997) Uhlmann, Jeffrey A New Extension of the Kalman Filter to Nonlinear Systems. *Proc. SPIE* 3068, signal processing, sensor fusion, and target recognition VI, 182 (28 July 1997)
13. Karaim M, Elsheikh M, Noureldin A (2018) GNSS Error sources. *Multifunct Oper Appl GPS*. <https://doi.org/10.5772/intechopen.75493>
14. Kim D, Chung T, Yi K (2015) Lane map building and localization for automated driving using 2D laser rangefinder. 2015 *IEEE Intell Veh Symp*, pp 680–685. <https://doi.org/10.1109/IVS.2015.7225763>
15. Kummerle J, Sons M, Poggenhans F, Kuhner T, Lauer M, Stiller C (2019) Accurate and efficient self-localization on roads using basic geometric primitives. *Proc - IEEE Int Conf Robot Autom* 2019-May, pp 5965–5971. <https://doi.org/10.1109/ICRA.2019.8793497>
16. Künsch HR (2013) Particle filters. *Bernoulli*, pp 19. <https://doi.org/10.3150/12-BEJSP07>

17. Kuutti S, Fallah S, Katsaros K, Dianati M, Mccullough F, Mouzakitis A (2018) A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet Things J* 5:829–846. <https://doi.org/10.1109/JIOT.2018.2812300>
18. Laboshinl (2022) Laboshinl/loam-velodyne. GitHub. Accessed 8, August. <https://github.com/laboshinl/loam-velodyne>
19. Levinson J, Montemerlo M, Thrun S (2007) Map-based precision vehicle localization in urban environments. *Robotics: Sci Syst III*, <https://doi.org/10.15607/rss.2007.iii.016>
20. Li Q, Chen S, Wang C, Li X, Wen C, Cheng M (2019) LO-Net: deep realtime Lidar odometry. In: *Proc IEEE Conf Comput Vis Pattern Recognit*, pp 8473–8482
21. Liu Z, Zhang F (2021) BALM: bundle adjustment for lidar mapping. *IEEE Robot Autom Lett* 6:3184–3191. <https://doi.org/10.1109/LRA.2021.3062815>
22. Lu W, Wan G, Zhou Y, Fu X, Yuan P, Song S (2019) DeepICP: an end-to-end deep neural network for point cloud registration. *Proc IEEE Int Conf Comput Vis 2019-October*, pp 12–21. <https://doi.org/10.1109/ICCV.2019.00010>
23. Moireau P, Dominique C (2010) Reduced-order unscented kalman filtering with application to parameter identification in large-dimensional systems. *ESAIM Cont, Optimisation Calculus of Variations* 17(2):380–405. <https://doi.org/10.1051/cocv/2010006>
24. Montiel JMM, Mur-Arta R, Tardos JD (2015) ORB-SLAM: a versatile and accurate monocular. *IEEE Trans Robot* 31:1147–1163
25. Mur-Artal R, Tardos JD (2017) ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans Robot* 33:1255–1262. <https://doi.org/10.1109/TRO.2017.2705103>
26. Nadav DYM, Maron H, Lipman Y (2017) DS++: a flexible, scalable and provably tight relaxation for matching problems. *ACM Trans Graph*, pp 36. <https://doi.org/10.1145/3130800.3130826>
27. Nerem RS, Larson KM (2001) *Global positioning system, theory and practice* 5th edition. <https://doi.org/10.1029/01eo00224>
28. Onyekpe U, Palade V, Herath A, Kanarachos S, Fitzpatrick ME (2021) WhONet: wheel odometry neural network for vehicular localisation in GNSS-deprived environments. *Eng Appl Artif Intell*, pp 105. <https://doi.org/10.1016/j.engappai.2021.104421>
29. Schaefer A, Büscher D., Vertens J, Luft L, Burgard W (2021) Long-term vehicle localization in urban environments based on pole landmarks extracted from 3-D lidar scans. *Rob Auton Syst* 136:103709. <https://doi.org/10.1016/j.robot.2020.103709>
30. Sefati M, Daum M, Sondermann B, Kreiskother KD, Kampker A (2017) Improving vehicle localization using semantic and pole-like landmarks. *IEEE Intell Veh Symp Proc*, pp 13–19, <https://doi.org/10.1109/IVS.2017.7995692>
31. Shan T, Brendan E (2018) Lego-loam, : lightweight and ground-optimized lidar odometry and mapping on variable terrain 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), <https://doi.org/10.1109/iros.2018.8594299>
32. Shuang L, Cao Z, Wang C, Yu J, Wang S (2021) A novel sparse geometric 3-d lidar odometry approach. *IEEE Syst J* 15(1):1390–1400. <https://doi.org/10.1109/jsyst.2020.2995727>
33. Sjafrie H (2013) Introduction to self-driving vehicle technology. *Bernoulli* 19:1391–1403. <https://doi.org/10.3150/12-BEJSP07>
34. Sjafrie H (2019) Introduction to self-driving vehicle technology (1st edn.) Chapman and hall/CRC. <https://doi.org/10.1201/9780429316777>
35. Team CARLA (2022) Carla CARLA simulator. 9 Accessed August, <https://carla.org/>
36. Thrun S (2002) Probabilistic robotics. *Commun ACM* 45:52–57. <https://doi.org/10.1145/504729.504754>
37. Van Der Merwe R, Wan R (2004) Sigma point Kalman filters for probabilistic inference in dynamic state-space models. PhD thesis, OGI school of science & engineering, oregon health & science university USA
38. Wael F (2021) Real-Time Autonomous vehicle localization based on particle and unscented kalman filters. *J Cont, Autom Electric Syst* 32(2):309–25. <https://doi.org/10.1007/s40313-020-00666-w>
39. Weng L, Yang M, Guo L, Wang B, Wang C (2019) Pole-based real-time localization for autonomous driving in congested urban scenarios. 2018. *IEEE Int Conf Real-Time Comput Robot RCAR 2018:96–101*. <https://doi.org/10.1109/RCAR.2018.8621688>
40. Xiao P, Shao Z, Hao S, Zhang Z, Chai X, Jiao J, Li Z, Wu J, Sun K, Jiang K, Wang Y, Yang D (2021) Pandaset: advanced sensor suite dataset for autonomous driving. *IEEE Conf Intell Transp Syst Proc, ITSC*. 2021-September, pp 3095–3101, <https://doi.org/10.1109/ITSC48978.2021.9565009>
41. Yuan W, Eckart B, Kim K, Jampani V, Fox D, Kautz J (2020) DeepGMR: learning latent gaussian mixture models for registration. *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)*, 12350 LNCS, pp 733–750. [https://doi.org/10.1007/978-3-030-58558-7\\_43](https://doi.org/10.1007/978-3-030-58558-7_43)

42. Yurtsever E, Lambert J, Carballo A, Takeda K (2020) A survey of autonomous driving: common practices and emerging technologies. *IEEE Access* 8:58443–58469. <https://doi.org/10.1109/ACCESS.2020.2983149>
43. Zhang J, Singh S (2017) Low-drift and real-time lidar odometry and mapping. *Auton Robots* 41:401–416. <https://doi.org/10.1007/s10514-016-9548-2>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

## Affiliations

Anas Charroud<sup>1</sup>  · Karim El Moutaouakil<sup>2</sup> · Ali Yahyaouy<sup>3</sup>

Karim El Moutaouakil  
karim.elmoutaouakil@usmba.ac.ma

Ali Yahyaouy  
ayahyaouy@yahoo.fr

- <sup>1</sup> Engineering Science Laboratory (ESL), FST Fes, University Sidi Mohamed Ben Abdellah Morocco, Fes, Morocco
- <sup>2</sup> Engineering Science Laboratory (ESL), FPT Taza, University Sidi Mohamed Ben Abdellah Morocco, Taza, Morocco
- <sup>3</sup> Computer Science, Signals, Automatics and Cognitivism Laboratory, FSDM, USMBA, Fes, 30050, Morocco