Particle Filter based Localization using NCLT and KITTI Dataset

Shreshtha Basu, Kai Cheng, Atishay Singh, Hang Zhang

***Abstract*—Localization is essential for autonomous systems as it serves as a precursor to determining the system's future actions. This paper aims to solve localization in urban environments by using pole-like features from 3D LiDar data to build a map and employing a particle filter to estimate the pose of the autonomous system. Our paper explores the performance of a standard particle filter and the UKF Distribution based particle filter on the NCLT as well as the KITTI Dataset.**

***Index Terms*—Localization, Particle Filter, Poles Landmark, Occupancy Map**

I. Introduction

When designing and implementing autonomous vehicle systems, one major issue that often comes up is path planning and localization within densely populated, urban environments. Due to the increased traffic, pedestrian presence on the roads and, a crowded environment in general, attempting to navigate through urban environments often presents a series of unique challenges in the development of autonomous vehicles. In the past, GNSS systems have proved to be an efficient solution for localization, but can be unreliable in areas with tall buildings and foliage where the satellite signal may not be received. Localization based on grid maps and point clouds will be more dependable in such settings but is computationally heavy and requires high memory usage. Condensing these maps into salient features can make this problem more tractable. Once the map is represented by these salient features, a particle filter can be used to estimate the state of the autonomous vehicle.

In our project, we use a pole detector to extract landmarks from 3D LiDar scans and register them on a global map. For localization, we use a particle filter to estimate the vehicle pose by aligning the poles detected in the live sensor data with that of the global map. We have implemented a standard particle filter and a particle filter based on UKF Distribution and compared their performance.

This paper is organized as follows: Related work is covered in Section II. The Algorithmic Architecture is described in Section III - where we cover Pole Extractor, Global Mapping, Standard Particle Filter, and UKF Distribution based Particle Filter. Section IV comprises Experiments and Results, and finally, Section V concludes the paper with suggestions of future directions.

II. Related Work

Generally, the location of a vehicle or autonomous system can be estimated by a Bayesian estimator such as the Kalman Filter, the Extended Kalman Filter, or a Particle Filter. The Kalman Filter relies on the assumption that the system is linear and that the posterior probability is Gaussian [1]. Thus, it can achieve high accuracy only in such environments. The extended Kalman Filter is linearized over a small instance of time and assumes the properties of the Kalman Filter during that instant, making the extended Kalman Filter only an approximation of the actual model. Additionally, both will fail in non-Gaussian posterior probability environments. Particle filters can overcome these disadvantages by representing the posterior probability using samples [2].

Throughout the years, many variants of particle filters have emerged. The Rao Blackwellised Particle Filter (RBPF) wherein the posterior probability is expressed as a product of one estimator over the autonomous system's path and N independent estimators over the landmark positions conditioned on the path estimate is shown to be more accurate than the standard particle filter [3]. [4] proposed an improved RBPF algorithm where the UKF is utilized to estimate landmark features and an adaptive method to resample particles to maintain diversity. Our paper compares the performance between the standard particle filter and the proposed improved RBPF algorithm.

Another field of research lies in determining what kind of features be used to calculate the likelihood of the particles. This can vary from problem to problem. For instance, [1] uses 2D LiDar data to detect features on the road like curbs and road markings and generate a grid-based feature map for autonomous vehicles. ORBSLAM is another common visual (monocular)SLAM technique where ORB features are used for place recognition [5]. This algorithm is robust enough to work in any environment. [6] uses pole-like features from 3D LiDar scans to serve as vehicle landmarks for localization. In our project, we use their polex library [7] for feature detection.

III. Algorithm Architecture

As shown in Fig. 1, the high-level idea of our algorithm is to first build a landmarks map by extracting pole landmarks from the point cloud generating by 3D Lidar scans and then feed this landmarks map to a particle filter. The particle filter will then search and get the position measurements of the nearby landmarks via a KDtree, and use them to correct the previous pose prediction, which is derived based on IMU data. Besides, the measurement of a precise RTK-GPS is treated as the ground truth of pose according to [9].

A. Pole Extractor

For the landmarks extraction, we use the same method as the one proposed in [6] and implemented the poles extraction

Project Resources
YouTube Video, Git Repository

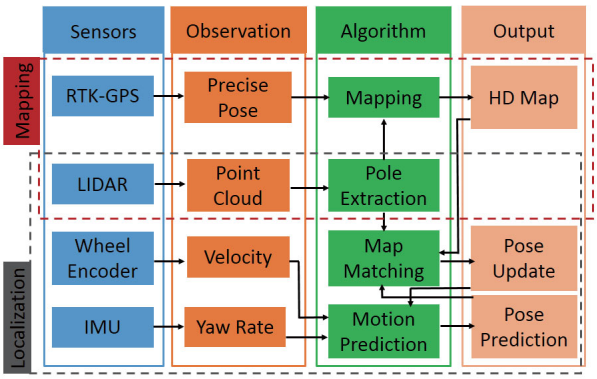

Fig. 1. Algorithm Architecture [8]

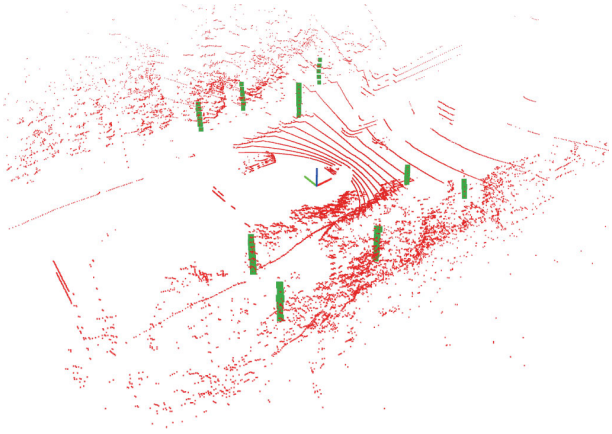

Fig. 2. Sample result for poles extraction [8]

module based on the provided Polex [7] library via Python 3 environment. The poles extraction module takes a set of 3D Lidar scans and returns the 2-D coordinates of the centers of the detected landmarks with respect to the ground plane, along with the estimated landmark widths. The basic idea behind this is to build an occupancy map from the laser ray reflection rate, and extract poles based on occupancy probability, which means that the greater the probability that the corresponding partition of space is part of a pole. To derive the specific calculation process, we denote a single laser measurement as $z = \{u, v\}$, where u is its Cartesian starting point and v is the endpoint. Then we start the process by tessellating the map space, trace the laser rays, and model the posterior probability that the j-th voxel reflects an incident laser ray based on the probability given in [10], which is shown as below:

$$p(\mu_j|Z) = Beta(h_j + \alpha, m_j + \beta) \tag{1}$$

Where h_j is the numbers of laser reflections, m_j is the transmissions in the j_th cell, α and β are the parameters of the prior reflection probability $p(\mu_j) = Beta(\alpha, \beta)$. Denote the maximum-likelihood reflection map as $M := h_j(h_j + m_j)^{-1}$, $\gamma = E[M]$ as its mean and $\sigma = var[M]$ as its variance, then α and β can be given as the following according to [6]:

$$\alpha = -\frac{\gamma(\gamma^2 - \gamma + \sigma)}{\sigma} \tag{2}$$

$$\beta = \frac{\gamma - \sigma + \gamma\sigma - 2\gamma^2 + \gamma^3}{\sigma} \tag{3}$$

For the next step, we covert the laser reflection rate distribution model $p(\mu_j|Z)$ we describe above to an occupancy map $O := \{o_j\}$, where the occupancy probability is given as below according to [6]

$$o_j = \int_{\mu_0}^{1} p(\mu_j|Z)\, d\mu_j \tag{4}$$

which means that we assume a cell is occupied if its reflection rate exceeds a threshold μ_0 defined by us. Next, we transform O to a 2-D map of pole scores in the ground plane. To achieve that, we first create a set of intermediate 3-D score maps $Q_a := \{q_{a,j}\}$ of the same size as O, then every cell $q_{a,j}$ tells how likely it is that this portion of space is part of a pole with edge length $a \in \mathbb{N}^+$ measured in units of grid spacing. Notice that we define a pole landmark as a vertical stack of occupied voxels with quadratic footprint, laterally surrounded by a hull of free voxels, if we calculate the difference between the mean occupancy value inside the pole and the maximum occupancy value of the volume of free space around the pole, we can get score lies in $[-1, 1]$, representing the probability that the corresponding partition of space is part of a pole, as shown in the Fig. 2. By merging the resulting 3-D maps $Q_a := \{q_{a,j}\}$ into a single map $Q = q_j = \{max_a\{q_{a,j}\}\}$, determining the contiguous vertical stack of voxels that all surpass a given score threshold q_{min} for each horizontal position in Q, and discarding all stacks that fall below a certain height threshold h_{min} and computing the mean score for each of the remaining stacks, we can transform the map into a 2-D score map. Finally, we convert this discrete score map to a set of continuous pole position along with width estimates which are computed as the weighted average over all pole widths a, where for every a, the weight is the mean of all cells in Q_a that touch the pole [6].

B. Global Mapping

In practice, simply applying a pole extractor described in the previous section to a set of registered laser scans is infeasible since the grid maps and laser scans have high memory complexity. Hence, in order to create such a landmarks map with limited memory resources, we partition the mapping trajectory into shorter segments of equal length and feed the Lidar measurements taken along each segment to the pole extractor one by one. To make sure these small segments are consistent with the global map, these local grid maps are aligned with the axes of the global map and that all of them have the same raster spacing. However, it can still happen that the length of a trajectory segment is smaller than the size of a local map, which will lead to multiple landmarks overlap. To solve this, we first project all poles onto the ground plane, which returns a set of axis-aligned squares, then we reduce them to a single pole estimate by computing a weighted average

Project Resources
YouTube Video, Git Repository

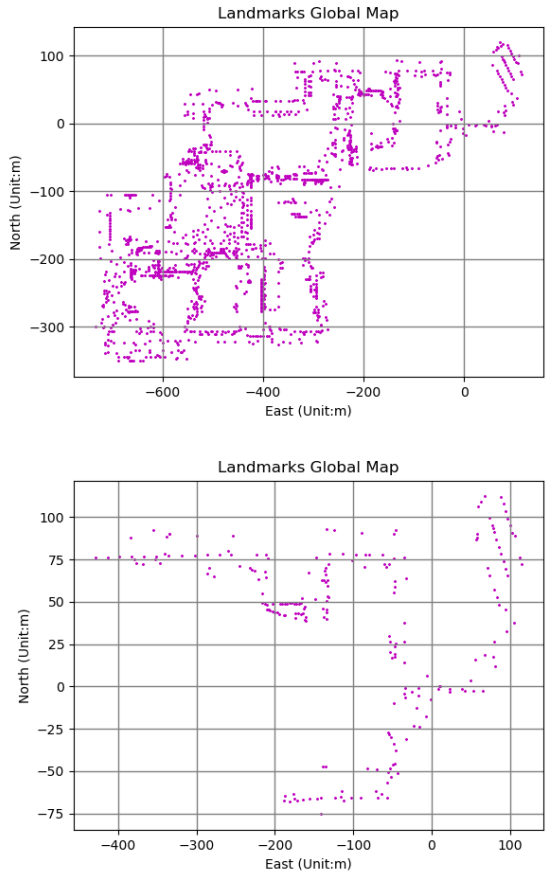

Fig. 3. Global Landmarks Map generated on an NCLT session

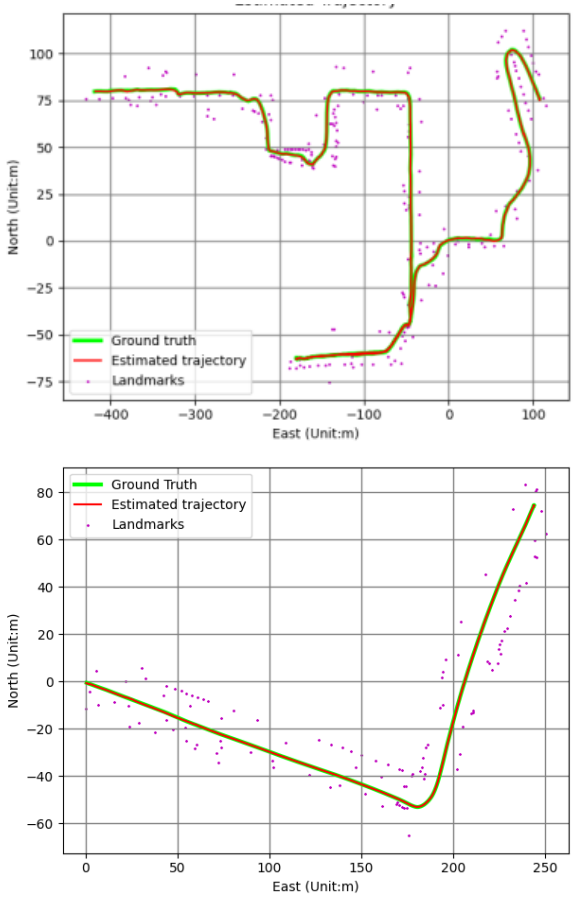

Fig. 4. Results Obtained by Particle Filter on an NCLT session (Up), Results Obtained by Particle Filter on a KITTI sequence (Down)

over their center coordinates and widths, where each weight is given by the mean pole score determined by averaging over the scores of all voxels that touch the pole in all score maps Q_a. As shown in Fig. 3, we can finally generate such a global landmarks map, which is used to derive measurements for correcting estimated pose by a particle filter later.

C. Standard Particle Filter

A particle filter is well suited for the localization task because it can not only maintain multiple pose hypotheses in parallel but also handle global localization. In the project, we first use the particle filter algorithm to implement the localization task. Concretely, as [7] proposed, the motion model is:

$$X_t = transform(\xi) X_{t-1} \quad | \quad \xi \sim \mathcal{N}(\chi, \Sigma) \tag{5}$$

where X_t are particles represented by 3 by 3 transformation matrices; ξ is subject to normal distribution with the mean of χ and the covariance of Σ, where $\chi = [x, y, \phi]^T$ denotes the latest relative odometry measurement, with x, y and ϕ representing the translation and the heading of the vehicle, respectively. The $transform(\xi)$ function is to change a vector ξ to the corresponding transformation matrix. For measurement update, we assume that the data associations between online landmarks λ_k and the reference landmarks r_n via KDTree search algorithm. Since the independence of each landmark, we update the particle weights according to the following measurement probability

$$\begin{aligned} w^{(i)} &= w^{(i)} \prod_k p(\lambda_k | X, r_{n(k)}) \\ &= w^{(i)} \prod_k \mathcal{N}(\|X\lambda_k - r_{n(k)}\|, \sigma) \end{aligned} \tag{6}$$

with the reference and online landmarks represented by homogeneous 2-D position vectors $[x, y, 1]^T$.

For pose estimation, we implemented three modes. First, MEAN mode: we used the weighted mean of all particles to obtain the estimation. Second, MAX mode: the particle which has the highest weight was considered as the estimation. Third, BEST mode: we used the weighted mean of top 10% particles to obtain the estimation. It turns out that the BEST performs best. The results are shown in Fig. 4.

D. UKF-Distribution Particle Filter

Since the traditional particle filter has particle degradation issue, which means that for very accurate sensors, it is possible for the sampling algorithm to select particles that are largely

Project Resources
YouTube Video, Git Repository

similar, cutting down on the diversity of data, the UKF-distribution particle filter is designed to compensate for issues in particle filter sampling. There are two main advantages of UKF-distribution particle filter. Firstly, UKF-distribution particle filter makes use of observations to guide the sampling of particles, which keeps both accuracy and diversity of data. Secondly, the UKF distribution does not depend on linearization. It only needs a few points to accurately obtain the mean and covariance of updated particles. In this project, we roughly implemented UKF-distribution particle filter algorithm to implement localization task. Concretely, first we used the same motion dynamics model in 5. Then instead of updating particles via motion dynamics model directly, we used UKF algorithm to obtain the mean and covariance of new particles and took the mean added by zero-mean Gaussian noise as updated particles:

$$\begin{aligned} X_t^{(i)}, \bar{\sigma}_t^{(i)} &= UKF(\bar{X}_{t-1}^{(i)}, \bar{\sigma}_{t-1}^{(i)}, r_{n(k)}) \\ \bar{X}_t^{(i)} &= transform(\mathcal{N}(0, \bar{\sigma}_t^{(i)}))X_t^{(i)} \end{aligned} \tag{7}$$

where i denotes the index of each particle; $\bar{X}_t^{(i)}$ denotes the ith particle at time horizon t. For updating weights, we used the important sampling formula to obtain new weights:

$$w^{(i)} = w^{(i)} * \frac{likelihood * prior}{proposal} \tag{8}$$

where

$$\begin{aligned} likelihood &= \prod_k \mathcal{N}(\|X\lambda_k - r_{n(k)}\|, \sigma) \\ prior &= \mathcal{N}\left((\bar{x}_t^{(i)} - \bar{x}_{t-1}^{(i)}), \bar{\sigma}_{t-1}^{(i)}\right) \\ proposal &= \mathcal{N}\left((\bar{x}_t^{(i)} - x_t^{(i)}), \bar{\sigma}_t^{(i)}\right) \end{aligned} \tag{9}$$

where $x_t^{(i)}$ means the vector corresponding to $X_t^{(i)}$. For estimating pose, we followed the same mode used in original particle filter algorithm. The UKF-distribution algorithm is shown in Algorithm 1.

Algorithm 1 UKF-distribution particle filter algorithm

Input: Particles $\bar{X}_{t-1}, w_{t-1}$, reference landmarks $r_{n(k)}$, resampling threshold n_t, number of particles N
Output: New particles$\bar{X}_t, w_t$, estimated pose p_t
for $i = 1, 2, \cdots, N$ **do**
  $X_t^{(i)}, \bar{\sigma}_t^{(i)} = UKF(\bar{X}_{t-1}^{(i)}, \bar{\sigma}_{t-1}^{(i)}, r_{n(k)})$ **(Update motion)**
  $\bar{X}_t^{(i)} = transform(\mathcal{N}(0, \bar{\sigma}_t^{(i)}))X_t^{(i)}$ **(Update particles)**
end
for $i = 1, 2, \cdots, N$ **do**
  $likelihood = \prod_k \mathcal{N}(\|X\lambda_k^{(i)} - r_{n(k)}\|, \sigma)$
  $prior = \mathcal{N}\left((\bar{x}_t^{(i)} - \bar{x}_{t-1}^{(i)}), \bar{\sigma}_{t-1}^{(i)}\right)$
  $proposal = \mathcal{N}\left((\bar{x}_t^{(i)} - x_t^{(i)}), \bar{\sigma}_t^{(i)}\right)$
  $w_t^{(i)} = w_{t-1}^{(i)} * \frac{likelihood * prior}{proposal}$ **(Update weights)**
  $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^N w_t^{(i)}}$**(Normalization)**
end
if $n_{eff} = \sum_{i=1}^N (w_t^{(i)})^2 < n_t$ **then**
  $\bar{X}_t, w_t, \bar{\sigma}_t^{(i)}$ = Resample($\bar{X}_t, w_t, \bar{\sigma}_t^{(i)}$) **(Resample)**
end
$\bar{x}_t = \text{transform}^{-1}(\bar{X}_t)$
$p_t = \sum_{i=1}^N w_t^{(i)} \bar{x}_t^{(i)}$ **(Estimate pose)**
return $\bar{X}_t$,w_t, p_t;

The results of UKF-distribution particle filter were shown in Fig. 5.

IV. Experiment and Results

The University of Michigan North Campus Long-Term Vision and Lidar (NCLT) dataset [9] and the KITTI dataset [11] are used in this project to evaluate the different particle filter algorithms. We generated a visualization of the particles, highlighting the nearest landmarks at every position as the robot moves. In Fig:6, the particles are represented as red dots, the poles in the map are represented by blue pixels, the nearest landmarks for calculating the likelihood are represented by the beige cones, the trajectory is represented by the green line and finally the direction of movement is represented by the black arrow.

A. Localization on the NCLT Dataset

This dataset consists of omnidirectional imagery, 3D LiDar, 2D LiDar, GPS, and proprioceptive sensors for odometry [9]. It is comprised of 27 sessions spaced biweekly over 15 months, and each session covers a trajectory of about 5 km. The sessions explore both indoor and outdoor environments, different trajectories, at varying times of the day and weather. The dataset has captured challenging environments like moving obstacles, changing lighting, weather changes, and different lighting.

For localization, the posterior is initialized with 5000 particles sampled from a circle within a 2.5 m radius around the earlier ground-truth pose. We use odometry data generated by fusing wheel encoder readings, gyroscope, and IMU data for

Project Resources
YouTube Video, Git Repository

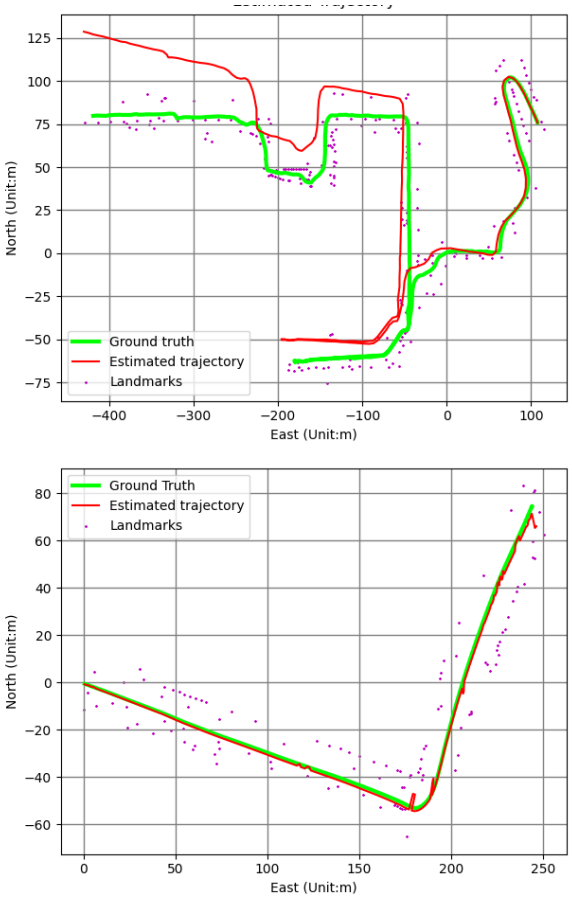

Fig. 5. Results Obtained by UKF-Distribution Particle Filter on an NCLT session (Up), Results Obtained by UKF-Distribution Particle Filter on a KITTI sequence (Down)

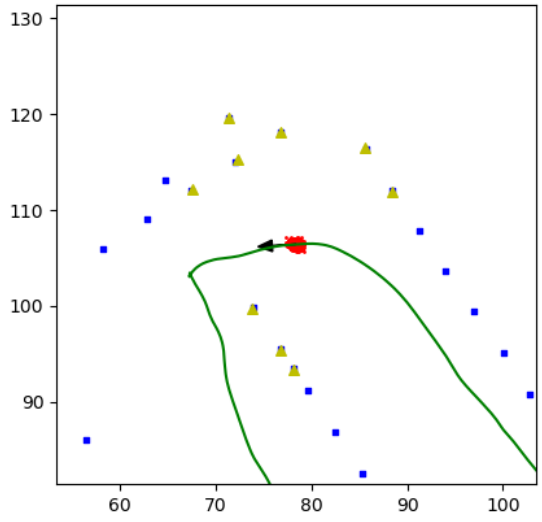

Fig. 6. Simulation of the robot particles following a trajectory

the motion model. The correction is done using landmarks seen by the 3D LiDar scanner and matching them to the landmarks in the map. Table I shows the results of running localization using the standard particle filter (PF) and the UKF Distribution Particle Filter (UKF - PF).

Date	Algorithm	Absolute Error (m)	RMS Error (m) Error(m)	Time Taken
2012-01-15	PF	0.1915	0.2242	2:02:49
	UKF - PF	21.3764	25.4162	2:16:05
2013-01-10	PF	0.1407	0.1954	18:10
	UKF - PF	19.2499	23.1936	19:46

TABLE I
RESULTS EVALUATED ON THE NCLT DATASET

As seen in Fig:4 (Up), the trajectory generated using the standard particle filter is very close to the ground truth trajectory. However, the errors generated using the UKF-PF are extremely large. Additionally, the UKF-PF generally also takes a longer amount of time to run. The trajectory is maintained close to the ground at the beginning, but the error accrues as the robot moves farther away from the starting position. The drift in the trajectory can be seen in Fig:5 (Up).

B. Localization on the KITTI Dataset

This dataset consists of data from high-resolution color and grayscale stereo cameras, 3D Velodyne LiDar, and a high-precision GPS/IMU inertial navigation system. It is comprised of sequences that capture diverse environments like inner-city scenes, rural areas, freeways, and includes many static as well as dynamic objects. The sequences in the KITTI dataset are usually shorter than the NCLT sessions, ranging between 10 seconds to about 2 minutes. The

Date	Algorithm	Absolute Error (m)	RMS Error (m) Error(m)	Time Taken
09-26-2011 Drive 5	PF	0.0876	0.112	39:00
	UKF - PF	2.534	2.873	1:07:00
09-26-2011 Drive 9	PF	0.0628	0.0737	1:24:56
	UKF - PF	1.209	1.476	1:06:16
09-26-2011 Drive 11	PF	0.0679	0.0781	30:32
	UKF - PF	0.1872	0.2004	33:04
09-26-2011 Drive 13	PF	0.1080	0.1196	29:29
	UKF - PF	0.3702	0.7003	32:00
09-26-2011 Drive 18	PF	0.0544	0.0607	37:00
	UKF - PF	0.1889	0.196	1:05:00

TABLE II
RESULTS EVALUATED ON THE KITTI DATASET

Table II shows the results of running localization using the standard particle filter (PF) and the UKF Distribution Particle Filter (UKF - PF) on the KITTI Dataset. In general it is seen that the performance of the UKF-PF is worst than that of the standard Particle Filter. However, the performance of the UKF PF is better on the KITTI Dataset than it is on the NCLT Dataset as seen when comparing the trajectory from Fig:4 (Down) and Fig:5 (Down). This could be attributed to the fact that the KITTI Dataset is comprised of shorter sequences, and so the errors are not able to accrue over time as much.

Project Resources
YouTube Video, Git Repository

V. Conclusion and Future Scope

Ultimately, based on our implementations of the standard and UKF-distribution particle filter, we conclude that the standard particle filter outperforms the UKF-distribution filter in terms of computation time and accuracy and is, therefore, better suited for localization and navigation applications within urban environments. We found that faulty or inaccurate data association significantly affects the performance of the UKF-distribution particle filter, making it unsuitable for applications in which landmarks are unlabelled, difficult to distinguish from each other, or otherwise inaccurately matched to observation data.

Going forward, the best course of action would be to expand on our standard particle filter implementation to deal with environments in which pole-like objects are sparse or difficult to detect, such as highways or rural environments. In these types of environments, we would likely have to significantly modify our data collection scheme to account for features that would be easier to detect in such environments or switch to a detection scheme capable of handling environments without pole features, such as ORB-SLAM or FAST-SLAM.

References

[1] I. Ullah, X. Su, X. Zhang, and D. Choi, "Simultaneous localization and mapping based on kalman filter and extended kalman filter," *Wireless Communications and Mobile Computing*, vol. 2020, 2020.

[2] K. Ahn and Y. Kang, "A particle filter localization method using 2d laser sensor measurements and road features for autonomous vehicle," *Journal of Advanced Transportation*, vol. 2019, 2019.

[3] K. Murphy and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 499–515.

[4] C. Ji, H. Wang, and Q. Sun, "Improved particle filter algorithm for robot localization," in *2010 2nd International Conference on Education Technology and Computer*, vol. 4. IEEE, 2010, pp. V4–171.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, "Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans," in *2019 European Conference on Mobile Robots (ECMR)*, 2019, pp. 1–7.

[7] ——, "Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans," https://github.com/acschaefer/polex, 2019.

[8] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang, "Pole-based real-time localization for autonomous driving in congested urban scenarios," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2018, pp. 96–101.

[9] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

[10] L. Luft, A. Schaefer, T. Schubert, and W. Burgard, "Closed-form full map posteriors for robot localization with lidar sensors," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6678–6684.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

Project Resources
YouTube Video, Git Repository