Received 10 May 2024; accepted 28 May 2024. Date of publication 11 June 2024; date of current version 11 July 2024.
The review of this paper was arranged by Associate Editor Subhas Chandra Mukhopadhay.

Digital Object Identifier 10.1109/OJIM.2024.3412219

LiDAR-Based Optimized Normal Distribution Transform Localization on 3-D Map for Autonomous Navigation

ABHISHEK THAKUR, (Graduate Student Member, IEEE), AND P. RAJALAKSHMI, (Senior Member, IEEE)

Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Hyderabad 502285, India.

CORRESPONDING AUTHORS: A. THAKUR AND P. RAJALAKSHMI (e-mail: ee20resch11014@iith.ac.in; raji@ee.iith.ac.in)

This work was supported in part by the Technology Innovation Hub on Autonomous Navigation (TiHAN) at the Indian Institute of Technology, Hyderabad, India, and in part by the Prime Minister Research Fellowship (PMRF).

ABSTRACT Autonomous navigation has become a topic of immense interest in robotics in recent years. Light detection and ranging (LiDAR) can perceive the environment in 3-D by creating the point cloud data that can be used in constructing a 3-D or high-definition (HD) map. Localization can be performed on the 3-D map created using a LiDAR sensor in real-time by matching the current point cloud data on the prebuilt map, which is useful in the GPS-denied areas. GPS data is inaccurate in indoor or obstructed environments, and achieving centimeter-level accuracy requires a costly real-time kinematic (RTK) connection in GPS. However, LiDAR produces bulky data with hundreds of thousands of points in a frame, making it computationally expensive to process. The localization algorithm must be very fast to ensure the smooth driving of autonomous vehicles. To make the localization faster, the point cloud is downsampled and filtered before matching, and subsequently, the Newton optimization is applied using the normal distribution transform to accelerate the convergence of the point cloud data on the map, achieving localization at 6 ms per frame, which is 16 times less than the data acquisition rate of LiDAR at 10 Hz (100ms per frame). The performance of optimized localization is also evaluated on the Kitti odometry benchmark dataset. With the same localization accuracy, the localization process is made five times faster. LiDAR map-based autonomous driving on an electric vehicle is tested in the TiHAN testbed at the IIT Hyderabad campus in real-time. The complete system runs on the robot operating system (ROS). The code will be released at https://github.com/abhishekt711/Localization-Nav.

INDEX TERMS Autonomous navigation, light detection and ranging (LiDAR), localization, mapping, point cloud, robot operating system (ROS).

I. INTRODUCTION

AUTONOMOUS vehicles are the pivotal aspects of the intelligent transportation systems (ITSs) and future mobility. Autonomous vehicles are one of the emerging research topics revolutionizing the future of ground vehicles. An autonomous driving assistance system provides the necessary safety features to the vehicle, which can reduce fatal accidents. Perception, localization and mapping, planning, and control constitute the four basic building blocks of the autonomous vehicles, as shown in Fig. 1. Perception is the process by which multiple sensors, such as light detection and ranging (LiDAR), cameras, and radars are used to perceive the environment by acquiring the image and point cloud data [1]. Maps are created using the data obtained from the sensor. On the built map, localization is a process of determining vehicle position and orientation.

LiDAR is one of the most widely used sensors in autonomous vehicles and it is used for environment perception, mapping, localization, and navigation [2], [3], [4]. The LiDAR emits laser pulses, which are reflected by various objects and then absorbed by LiDAR. It creates the 3-D point cloud data with x, y, and z coordinates and intensity values of each point. The depth of each point is calculated using the time difference between the emitted and received

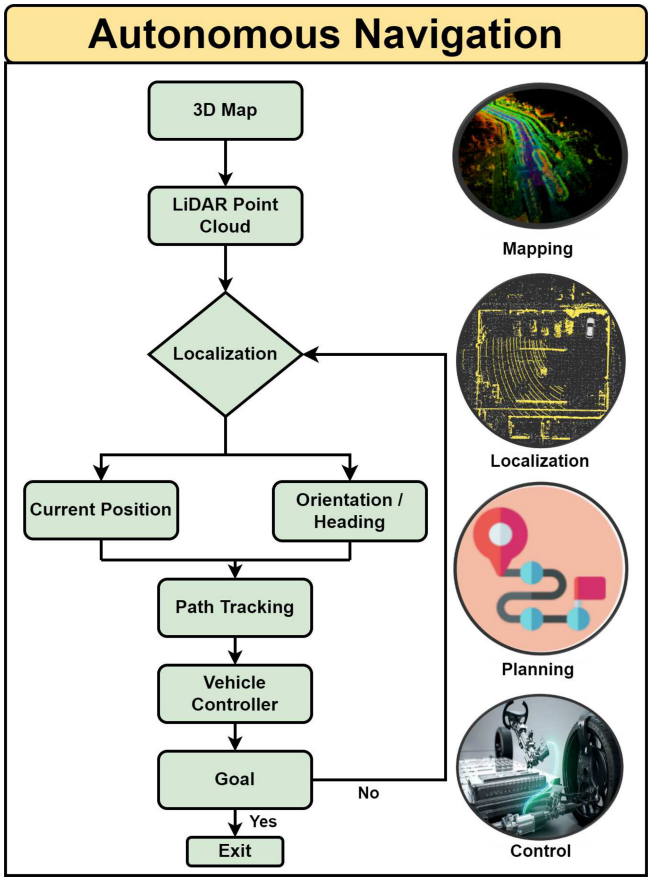

FIGURE 1. Block diagram of the autonomous navigation system.

pulse [5], [6]. Point cloud data creates a highly detailed and accurate 3-D map of an environment [7], [8]. Vision-based loop closure detection presents significant challenges due to changes in appearance and field of view. Additionally, camera sensors are susceptible to variations in seasons, weather, and lighting conditions. In contrast, LiDAR is less affected by lighting conditions (day and night) [9]. Localization of an autonomous vehicle is a crucial aspect of navigation, as it provides accurate and reliable information about the vehicle's position in 3-D space. The normal distribution transform (NDT) algorithm has been widely used for estimating the position and orientation of an autonomous vehicle in 3-D space by comparing the LiDAR scan data with a prebuilt 3-D map [10], [11]. Matching time is one of the essential parameters in the localization algorithm, and it means the time taken to match the current input scan frame on the map to determine the current position and orientation of the ego-vehicle. In this article, we adopted a probabilistic approach to match the LiDAR points on a 3-D point cloud map, which is independent of color or intensity information. This probabilistic approach significantly improved localization speed. However, integrating the camera and LiDAR fusion for high-definition (HD) map creation could greatly enhance autonomous vehicle capabilities, enabling the detection of lanes, road signs, and traffic signs, tasks that LiDAR alone cannot accomplish. LiDAR map-based navigation on autonomous vehicles is suitable for controlled environments and GPS-denied scenarios. Path-tracking algorithms, such as the pure pursuit algorithm, are commonly used in autonomous navigation. It allows a vehicle or robot to follow a desired path by continuously adjusting its steering angle to track a target point on the path ahead [12]. The LiDAR map-based localization and autonomous navigation are useful in the GPS-denied areas or areas where the GPS data is inaccurate. The significant contributions of this article are as follows.

1) A lightweight map of a long stretch is created using the key-frames and increasing the edge and plane resolution parameters to ensure smooth functionality within the robot operating system (ROS) platform. An optimized NDT localization algorithm for the LiDAR map is proposed. Point cloud filtering and downsampling reduce computational load, while the Newton nonlinear optimization method accelerates convergence on the map. This reduces matching time per frame to 6 ms, 16 times faster than the LiDAR's data acquisition rate at 10 Hz (100 ms per frame), enhancing the autonomous vehicle navigation.
2) The integration of localization and navigation algorithms onto a 3-D map to demonstrate the reliable autonomous navigation of a shuttle vehicle in real-time using only a LiDAR sensor. The localization performance is compared and evaluated on the Kitti odometry benchmark dataset. Our approach achieved faster localization with the same localization accuracy.

In Section II, the related work in this domain is discussed. The methodology of the proposed navigation system is discussed in Section III. The results and their analysis are done in Section IV.

II. RELATED WORK

Although autonomous navigation is one of the trending research topics, most of the research related to navigation in this area is limited to robots or based on GPS. Integrating mapping, localization, and navigation algorithms to demonstrate autonomous navigation on a large vehicle using the LiDAR is an area to be explored extensively. In [13], a method related to 3-D scan matching using an improved 3-D NDT for the mobile robotic mapping is discussed. The matching time per frame is 0.09 s (90 ms), nearly equal to the frame rate of the LiDAR data at 10 Hz. Ort et al. [14], [15] addressed the challenge of autonomous vehicle navigation in rural environments where detailed prior maps may not be available. It presents a navigation system that relies on the onboard sensors and perception algorithms to enable autonomous driving. This article used an open street map (OSM) for localization. Muñoz-Bañón et al. [16] proposed a navigation framework for autonomous vehicles based on the OSM data and LiDAR-based naive-valley-path obstacle avoidance. The framework leverages OSM

road information and LiDAR point cloud data to enable safe and efficient navigation in complex urban environments. Still, in many countries in rural areas, a detailed OSM map is not available, and getting a centimeter-level GPS position is a challenging task without a real-time kinematic (RTK) connection. Thus, the localization error will affect navigation accuracy. Zhang and Singh [17] proposed the LiDAR odometry and mapping algorithm (LOAM). The algorithm focuses on robustly estimating the motion of the sensor platform and concurrently building a 3-D map of the environment. There is a need for improvement in matching time per frame so that the vehicle can navigate in real time. Saarinen et al. [18] proposed the NDT Monte–Carlo localization (NDT-MCL) algorithm, which combines the NDT scan matching technique with the MCL framework for localization in autonomous robotics. The NDT scan matching algorithm efficiently aligns the robot's sensor measurements with a prebuilt map, while the MCL framework handles the probabilistic localization using a particle filter. Zhou et al. [19] proposed the NDT-Transformer algorithm, which leverages the NDT representation for the large-scale 3-D point cloud localization. The algorithm utilizes the transformer network architecture to process and align the 3-D point clouds with a prebuilt NDT map. The matching time per frame in [19] is relatively high, affecting the real-time navigation when the algorithm takes a few iterations to match the point cloud with the map. Wang et al. [20] proposed an improved version of the pure pursuit algorithm for path tracking in autonomous driving.

III. METHODOLOGY

A. SYSTEM OVERVIEW

LiDAR map-based navigation system includes the following steps as shown in Fig. 1. First, the 3-D Map or the HD Map is constructed by stitching the point cloud data of LiDAR. After that, real-time localization of the autonomous vehicles is done on the prebuilt map by matching the current point cloud with the map. The localization algorithm gives the current position and orientation of the vehicle in real-time, with respect to the map frame. The path-tracking algorithm is used for the navigation process. The mapping, localization, and navigation processes will be explained in detail in the upcoming section. The vehicle used in this experiment is a 14-seater electric vehicle powered by a 72 V/DC battery and propelled by a 7.2 kW motor. Steering and throttle control are managed by a MicroAutoBox controller. A Velodyne HDL-64 LiDAR sensor is mounted on the top of the vehicle for mapping and localization. The vehicle is equipped with an NVIDIA Jetson AGX Orin 32 GB computational board running on the UBUNTU 20 operating system, as shown in Fig. 2.

B. LIDAR MAPPING

The LiDAR creates the 3-D point cloud data which is used to construct the 3-D map of an environment, as shown in Fig. 3. The point cloud registration process constructs the map. Point cloud registration aims to align the different sets of points in the different coordinate systems into the same coordinate system. The drift error in matching is minimized by finding the accurate transformation between the points. We have created the map by extracting the feature and matching the current LiDAR frame to the cloud map. The edge and plane resolution is increased and the key-frames are chosen based on the rotational and translational changes to create a lightweight map as shown in Fig. 4. Also, a map is validated on the Kitti ground truth dataset and verified using the loop closure method in our previous work in [21]. The steps followed to create the map on our point cloud data are as follows.

FIGURE 2. E-vehicle mounted with Velodyne HDL-64 LiDAR, NVIDIA Jetson AGX Orin computing platform, and MicroAutoBox vehicle controller.

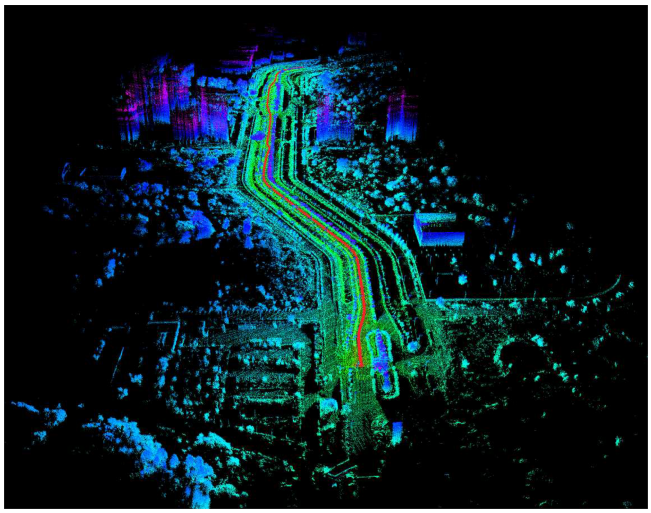

FIGURE 3. Map created by Velodyne 64 channel LiDAR at IIT Hyderabad campus.

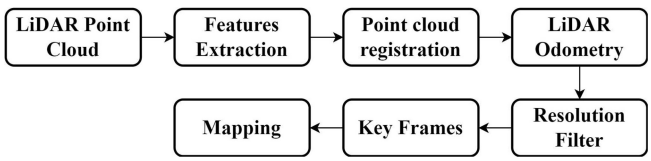

FIGURE 4. Block diagram of the mapping algorithm to create a lightweight map.

1) FEATURE EXTRACTION

LiDAR captures the data at the rate of 5–20 Hz. LiDAR data rate is kept at 10 HZ in this experiment, but the mapping algorithm runs at 1 Hz to minimize the size of the map. The point cloud is segmented into smaller parts, and features, such as the edges and planes are extracted. Edges in LiDAR data correspond to points with a significant change in the surface orientation or height. Edge extraction algorithms typically search for the points with high local curvature [17], [21]. Planar surfaces in the LiDAR data correspond to regions with approximately constant surface normal vectors. Let's suppose L_k is the kth frame of the point cloud data and $L^m_{(k,i)}$ is the ith point from the mth ring of LiDAR. In scan ring m, let $S^m_{(k,i)}$ is the set of neighboring point adjacent to $L_{(k,i)}$. The feature points are selected as the edge or planar points based on the curvature value, which is determined by [17]

$$C = \frac{1}{|N_s| \cdot \left\| L_{(k,i)} \right\|} \left\| \sum_{j \neq i} \left(L_{(k,i)} - L_{(k,j)} \right) \right\|. \tag{1}$$

Here, $L_{(k,j)}$ denotes the adjacent points of $L_{(k,i)}$ in $S^m_{(k,i)}$ and N_s is the number of points in $S^m_{(k,i)}$. Each LiDAR frame is divided into eight identical subregions. From each subregion, a maximum of two edge points and four planar points are selected. Edge points are typically defined as points on the boundary between the two surfaces, such as the edges of buildings, cars, or trees. In general, edge points have high curvature values and are characterized by sharp changes in depth or intensity. Once these regions are identified, the edge points (E_k) are those points that have curvature value larger than the threshold value (C_{th}). Planar points lie on a flat surface, such as the ground, walls, or roofs of buildings. Planar points (P_k) generally have low curvature and are characterized by a consistent depth or intensity value. Planar points are selected when the curvature value exceeds the threshold (C_{th}).

2) LIDAR ODOMETRY

LiDAR odometry is estimated by finding the transformation between the consecutive LiDAR frames of the point cloud data. LiDAR odometry minimizes the distance between the edge line and the planar surface between the two consecutive frames. Yaw rotation and translation in the ground frame are considered to determine the motion of the ground vehicle. Odometry can be estimated by finding the correspondence in feature points. Once the transformation matrix is found between the consecutive LiDAR frame, which consists of the rotational and translational parameters [17], [21], [22]. Each point in the current LiDAR frame can be rotated and translated to accumulate in the reference of the map axis, which is the map's origin.

3) LIDAR MAPPING

The mapping algorithm runs at a lower frequency at 1 Hz as compared to the odometry algorithm, which runs at 10 Hz. The feature points extracted are matched in subsequent frames to create a 3-D map of an environment. Edge and plane features are filtered by increasing resolution parameters. The feature points are updated based on the keyframes to make the mapping algorithm faster. The keyframes are selected based on the rotational and translational changes greater than the predefined threshold to reduce the size of the map. The mapping algorithm matches the current frame of the LiDAR data L_{k+1} to the mapped cloud M_k. M_k is the map formed by matching all the point cloud frames up to L_k by transforming the points in the current LiDAR frame onto the map's origin

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{2}$$

Any feature points (x, y, z) in point cloud can be transformed on the map axis using transformation matrix which consist of the rotational ($R \in \mathbb{R}^{3\times3}$) and translational parameter ($T \in \mathbb{R}^{3\times1}$) to (m_x, m_y, m_z) to form the map cloud M_k as given in (2). Also, the map is downsampled by a voxel grid filter to reduce the computational load. We have achieved the same localization accuracy tested on the Kitti odometry ground truth data by reducing the computational load in our work [21]. Also, the map accuracy can be tested using the loop closure method. Loop closure detects previously visited locations to correct the drift errors in pose estimation. Factors like rapidly changing surroundings or ambiguous features may challenge the algorithm's ability to detect loop closures accurately. IMU data can also be utilized in loop closure detection in a dynamic environment, which helps identify previously visited locations. IMU data, which includes measurements, such as acceleration and angular velocity, can be integrated with the LiDAR data during mapping. By fusing these sensor inputs, the system can better estimate the vehicle's motion over time and the system can detect loop closures and correct accumulated drift errors, thereby improving the overall accuracy of the LiDAR mapping process [23]. Loop closure is performed based on the keyframes, which are selected according to the rotational and translational thresholds [22]. Loop closure is also initiated when the drift error exceeds the threshold required to match the selected keyframes. Point clouds are iteratively aligned by minimizing the distance between their corresponding points, a process known as point cloud registration. This process depends entirely on the dynamics of an environment, including how frequently significant transformations (rotational and translational changes) occur between the point cloud frames. So, once the point cloud data is stitched by compensating for the distortion and drift errors, an environment map is created. We store the map in.pcd format, used while navigating the vehicle on the given map, by localizing the vehicle. All the points stored in the pcd files of the map have coordinates values (x, y, and z) with respect to the map's origin. The map's origin is the starting point when we start collecting the data to create a map.

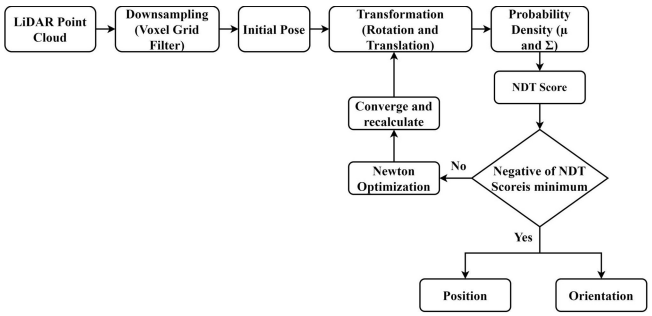

FIGURE 5. Block diagram of the localization algorithm.

C. LOCALIZATION ON PREBUILT MAP

Localization is done by matching the current point cloud data on the 3-D map generated by LiDAR. The localization algorithm gives the current position and orientation of the vehicle which is used for autonomous navigation. GPS-based localization is not reliable in an indoor or obstructed environment where a vehicle is parked inside the parking area. Also, a costly RTK connection is required in a GPS-based system to get the centimeter-level accuracy. So, we have used LiDAR only for localizing the vehicle on the given map. The localization algorithm is made faster by downsampling the point cloud using a voxel grid filter and subsequently applying a nonlinear Newton optimization which converges the point cloud on the map quickly, so multiple iterations can be performed before the next frame of point cloud data arrives. The block diagram of the localization algorithm is shown in Fig. 5.

Normal distribution parameters of points are used to calculate the matching score to localize the vehicle accurately and determine the position and orientation of the vehicle on the given map. Most of the time, localization fails when the current point cloud mismatches with the 3-D map of an environment. Also, the algorithm should perform the matching task in fewer iterations. We have optimized the traditional NDT algorithm by reducing the matching time per frame for the smooth navigation of the autonomous vehicle. To reduce the matching time, the map point cloud and the input point cloud are filtered before matching instead of matching the complete point cloud data, which takes more time to process. First, the point cloud is filtered using a voxel grid with a larger leaf size to reduce the number of points to match with the map to reduce the matching time. The normal distribution parameter mean (μ_k) and the covariance matrix (Σ_k) is given as [13], [24]

$$\mu_k = \frac{1}{N_k} = \sum_{i=1}^{N_k} x_{(k,i)} \tag{3}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=j}^{N_k} \left(x_{(k,i)} - \mu_k\right)\left(x_{(k,i)} - \mu_k\right)^\top. \tag{4}$$

Here, $(x_1, x_2, \ldots, x_k \ldots, x_N)$ is the mapped cloud with the N points and x_k is the kth normal distribution voxel with the N_k points. Similarly, the input point cloud data is filtered using the voxel grid filter as discussed above and transformed into map coordinates using the 3-D coordinate transformation

$$x'_i = \mathbf{R}x_i + \mathbf{T}. \tag{5}$$

$\mathbf{R}$ and $\mathbf{T}$ are the rotational and translational parameters that transform the input point cloud x_i into the map frame. The current LiDAR scan is matched with the NDT grid by finding the grid cells that correspond to the observed points in the LiDAR scan. The likelihood of similarity is computed between the points in the scan and the map and the statistical information stored in the grid cells. Each filtered point is transformed and compared to the statistical information stored in the corresponding grid cell. The similarity between the point and the cell's distribution calculated is known as the NDT score. The NDT score is calculated by matching the transformed point cloud and map point cloud given by

$$N_s = \sum_i^N \exp \frac{-\left(x'_i - \mu_i\right)^\top \Sigma_i^{-1} \left(x'_i - \mu_i\right)}{2}. \tag{6}$$

A higher NDT score (N_s) value means the input point cloud and reference map are well matched. While navigating, we kept the NDT score threshold so that the navigation starts when the NDT score is above the threshold value. Transformation probability is the NDT score of a point obtained by dividing the number of points in the input scan to be matched. After an initial guess of the vehicle's location on the map, a few iterations are performed to align the input point cloud on the map. The best pose (position and orientation) estimate is obtained by optimizing the vehicle's pose parameters using the Newton's nonlinear optimization algorithm.

Newton's method aims to address this issue by generating a sequence x'_i starting from an initial guess. This sequence converges toward a minimizer x'^* of the function f through the use of a series of the second-order Taylor approximations. The second-order Taylor expansion of f at x'_i is then applied in this process

$$f\left(x'_i + k\right) \approx f\left(x'_i\right) + f'\left(x'_i\right)k + \frac{1}{2}f''\left(x'_i\right)k^2. \tag{7}$$

The subsequent iteration x'_{i+1} is determined to minimize this quadratic approximation in terms of k, and it is set as $x'_i + k$. When the second derivative is positive, the quadratic approximation forms a convex function of k, and finding its minimum involves setting the derivative to zero. As the minimum is attained for

$$\frac{d}{dk}\left(f\left(x'_i\right) + f'\left(x'_i\right)k + \frac{1}{2}f''\left(x'_i\right)k^2\right) = 0 \tag{8}$$

$$= f'\left(x'_i\right) + f''\left(x'_i\right)k \tag{9}$$

$$k = -\frac{f'\left(x'_i\right)}{f''\left(x'_i\right)}. \tag{10}$$

Putting everything together, the Newton's method performs the iteration

$$x'_{i+1} = x'_i + k = x'_i - \frac{f'(x'_i)}{f''(x_i)}. \tag{11}$$

This optimization step aims to find the pose that maximizes the likelihood or similarity between the LiDAR scan and the NDT grid. Once the negative of the NDT score is minimum, the transformation parameters are calculated, which provides the rotation and translation of the current LiDAR axis from the origin of the map axis. Rotation parameters, such as roll, pitch, and yaw give the orientation of the ego-vehicle and the translation parameters give the current position of the ego-vehicle.

D. NAVIGATION AND PATH TRACKING

The localization algorithm gives the current position and orientation of the vehicle. The vehicle's current position is given with the (c_x, c_y, c_z) coordinates with respect to the map coordinate, and the origin belongs to the map's starting point. The center of the LiDAR sensor considered the vehicle's current position with respect to the map's origin. Orientation of the vehicle is calculated in the quaternion form (q_x, q_y, q_z, q_w) using the localization algorithm. Navigation requires the rotation of vehicles in the x-y plane with respect to the z-axis. So, we need to calculate the current heading, i.e., yaw (y_θ) using quaternion by

$$y_\theta = \arctan\left(\frac{2q_w q_z + q_x q_y}{1 - 2\left(q_y{}^2 + q_z{}^2\right)}\right). \tag{12}$$

The waypoints are recorded using the localization algorithm by driving the vehicle on the desired path on the LiDAR map. Waypoints array contains the x and y coordinates $[w_x, w_y]$. In autonomous navigation, the ego vehicle's current position and yaw rotation using the localization algorithm is called in the callback function in the navigation code continuously at the rate of 10 Hz. The required offset (x_o) and (y_o) are calculated as the difference between the next waypoint and the vehicle's current position in the x and y directions, respectively. The vehicle's current position (c_x, c_y) is the center of LiDAR from the map's origin

$$y_o = w_y - c_y \tag{13}$$

$$x_o = w_x - c_x. \tag{14}$$

The bearing angle (θ) from the current position of the vehicle with respect to the next waypoint in an anti-clockwise direction is calculated in degrees as

$$\theta = \arctan\left(\frac{y_o}{x_o}\right)\left(\frac{180}{\pi}\right). \tag{15}$$

To keep the value of θ positive, 360 is added when $\theta < 0°$

$$\theta = \theta + 360°; \left(\text{ If } \theta < 0°\right). \tag{16}$$

The required bearing is the angle between the vehicle's heading and the target waypoint is the difference between the bearing angle and heading of the vehicle (yaw) derived in (12) and (15)

$$\alpha = \theta - y_\theta. \tag{17}$$

Now, let us consider a case in which the vehicle is moving along the positive x-axis, then the vehicle heading y_θ will fluctuate as 1° and 359°, and θ is small, so to keep the alpha in between $[-180°, +180°]$

$$\alpha = \alpha + 360°; \left(\text{If } \alpha < -180°\right) \tag{18}$$

$$\alpha = \alpha - 360°; \left(\text{If } \alpha > 180°\right). \tag{19}$$

The steering input which is given to the actuator is calculated after fine tuning the vehicle dynamics using the pure pursuit controller algorithm [12], [25] as

$$S_{ip} = k * \arctan\left(\frac{2 * L * \sin\left(\frac{\pi * \alpha}{180°}\right)}{L_d}\right). \tag{20}$$

Here, L_d is the distance between the rear axle and the target points and $L = 3.5$ is the length from the rear axle to the front axle of the 14-seater campus shuttle vehicle used in this experiment. k is the fine-tuning parameter. dSpace's MicroAutoBox is used as a controller that controls the vehicle's acceleration and steering.

IV. RESULTS AND ANALYSIS

In this experiment, Velodyne HDL-64 LiDAR, Jetson AGX Orin developer kit is a computational board on the 14-seater electric vehicle. MicroAutoBox controller is used to control the actuators in the vehicle. ROS Noetic is used as the computing platform. LiDAR map-based autonomous navigation is tested in three different routes inside the campus of the Indian Institute of Technology Hyderabad, India (IIT Hyderabad). Route 1 covers the indoor and outdoor area of TiHAN (Technology Innovation Hub on Autonomous Navigation) testbed of 250 m. Routes 2 and 3 cover an area of around 2.5 km each inside the campus. The accuracy of the mapping and localization algorithm is also tested on the standard KITTI dataset in our previous work in [21], and the localization accuracy was 2.13 cm on the Kitti odometry dataset. As discussed in Section III-B, a map of all the routes is created. The map of all the routes is shown in Figs. 7(a), 8(a), and 9(a). The line and plane resolution parameter is increased, which does not affect the localization algorithm and also reduces the size of the pcd file, which stores the point of the mapped cloud. Mapping results are shown in Table 1. Creating a LiDAR map for a long stretch is computationally expensive, but we have also created a map of a 2.5 km path in real-time by selecting the key-frames based on the rotational and translational threshold. As the number of point cloud frames increases, the computational time per frame for stitching the map increases. For the route of 2.5 km, the average computational time per frame is 93.54 ms, which is less than the LiDAR data acquisition rate, i.e., 10 Hz (100 ms per frame). For route 1, which is 250 m, the mapping time per frame is 33.85 ms. The graph

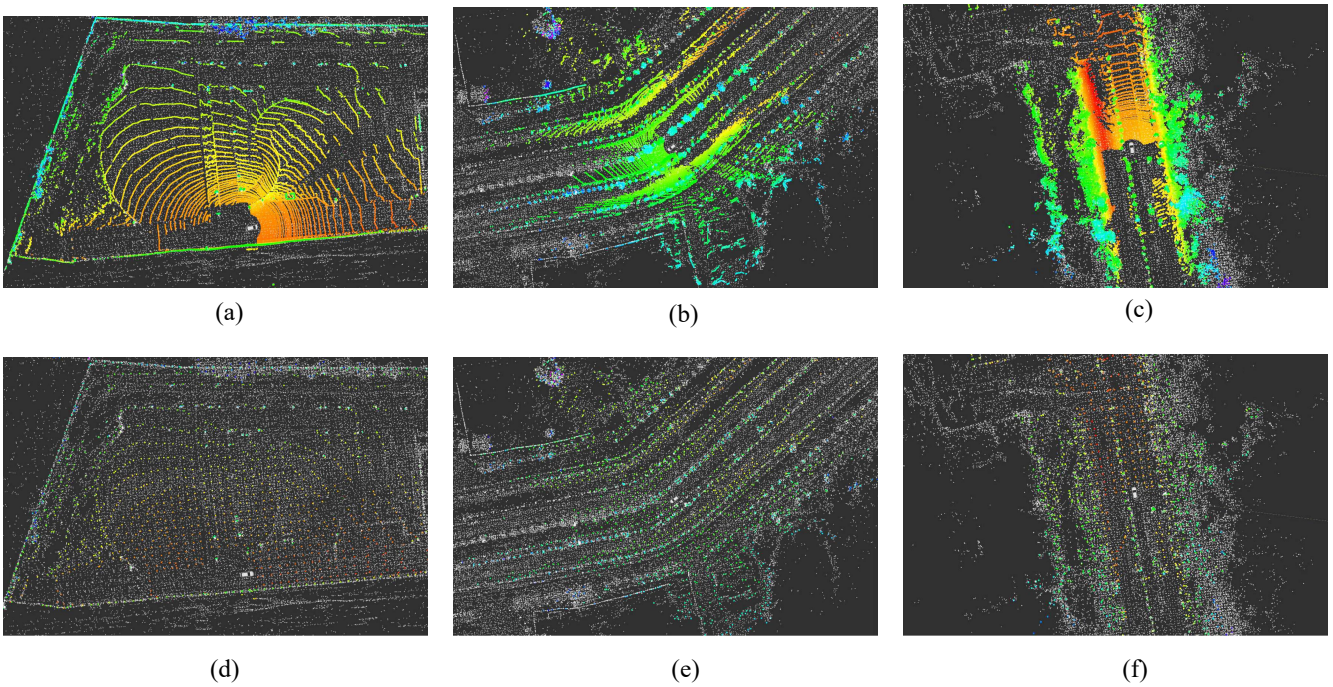

FIGURE 6. Localization on the different routes. (a)–(c) are the point cloud matching on the map and (d)–(f) are the filtered point cloud matching on the map. (a) Route 1. (b) Route2. (c) Route 3. (d) Route 1 (filtered points). (e) Route2 (filtered points). (f) Route 3 (filtered points).

for frame-wise mapping time is shown in Figs. 7(b), 8(b), and 9(b). Thus, the map can be created in real-time. The map file size is very small, which can be loaded into ROS and used for localization and navigation.

While localizing, instead of matching the whole points in the point cloud data, the point cloud is filtered and the optimized NDT localization algorithm is applied as mentioned in Section III-C. The average number of points in the point cloud data of Velodyne HDL-64 is roughly around 120 000 per frame when the data is acquired at 10 Hz. Point cloud data is filtered using a voxel grid filter of leaf size 3 to reduce the number of points to perform NDT matching.

In Table 2, the number of points per frame in the filtered point cloud is given. The number of points to be matched with the map is very less. The average number of points in a filtered point cloud is 1500–1600 per frame. This reduces the points to be matched in a current scan by around 80 times. Also, the point cloud data converges quickly on the map by applying the nonlinear Newton optimization using (11). Thus, the localization algorithm became very fast. Localization on the LiDAR map in different routes is shown in Fig. 6. The filtered point cloud is matched with the map shown in Fig. 6(d)–(f). In the filtered point cloud, the graph for the frame-wise number of points is shown in Figs. 7(c), 8(c), and 9(c). The average matching time per frame is around 5 and 6 ms for the localization algorithm. While matching initially, the input point cloud will converge faster and match with the map. The frame-wise matching time is shown in Figs. 7(d), 8(d), and 9(d). In all the routes, the vehicle is localized accurately with the map without failure. Transformation probability is around more than 50% in all the dynamic routes. The score reduces with the increase in dynamic obstacles. The navigation algorithm runs only when the transformation probability is greater than the threshold value (0.3). The frame-wise transformation probability graph is shown in Figs. 7(e), 8(e), and 9(e).

The localization algorithm obtains the vehicle's current position and orientation. The steering input is calculated as discussed in Section III-D to track the waypoints. The average velocity of the vehicle is around 10–13 km/hr in different routes during localization and autonomous navigation. Route 1 is a mix of outdoor and indoor navigation. LiDAR map-based autonomous navigation works perfectly in both the indoor and outdoor navigation. GPS-based localization does not provide high accuracy in indoor environments. The localization and navigation algorithms are integrated via rostopics. Navigation drift is calculated by finding the deviation of the navigation trajectory from the waypoint trajectory. Navigation trajectory is the trajectory of the vehicle in autonomous mode. An average navigation deviation of 4 and 5 cm is found. The waypoint trajectory and navigation trajectory for all the routes are shown in Figs. 7(f), 8(f), and 9(f).

The point cloud is filtered and downsampled to decrease the computational load, as matching time relies on the number of points in a point cloud frame, as it calculates and optimizes the probability distribution for each point. Additionally, the Newton nonlinear optimization method is employed to accelerate the convergence of the point cloud on the map. The average matching time per frame of 5 and

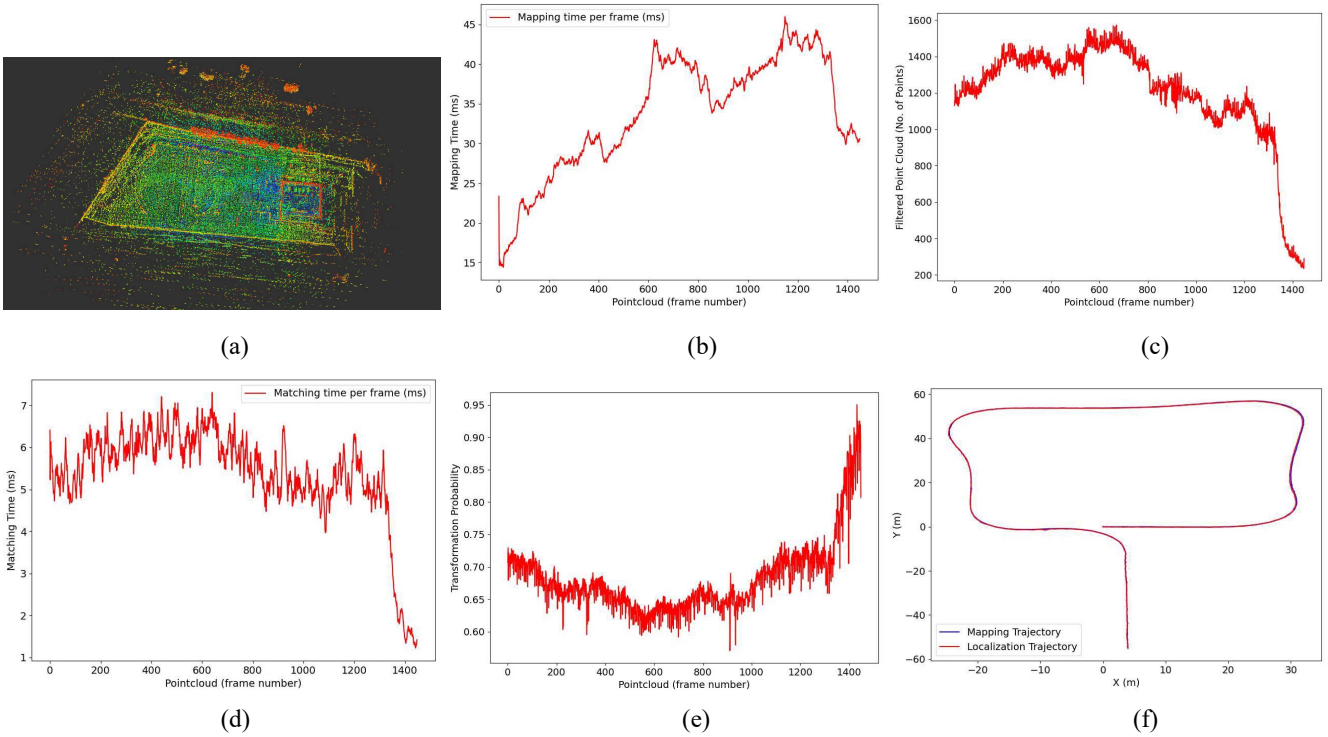

FIGURE 7. (a) Map. (b) Mapping time per frame. (c) Number of points in the filtered point cloud. (d) Matching time per frame during localization. (e) Transformation probability. (f) Mapping and localization trajectory of Route 1.

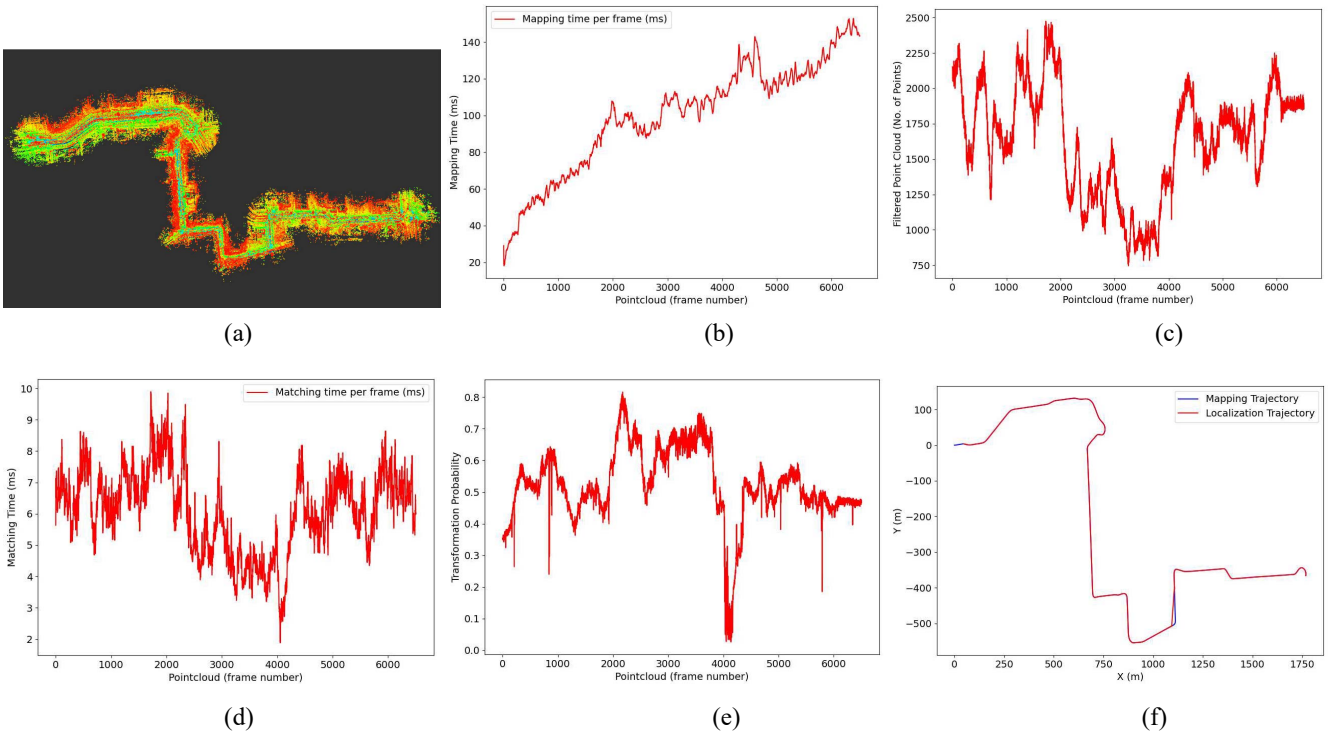

FIGURE 8. (a) Map. (b) Mapping time per frame. (c) Number of points in the filtered point cloud. (d) Matching time per frame during localization. (e) Transformation probability. (f) Mapping and localization trajectory of Route 2.

6 ms is achieved. Also, the matching time drops when the vehicle is stationary after reaching the destination point, as shown in Fig. 7(d) because the current position is the same in the consecutive frames, and there is no transformation between the consecutive frames. ROS clock is used in the algorithm to calculate the matching time per frame.

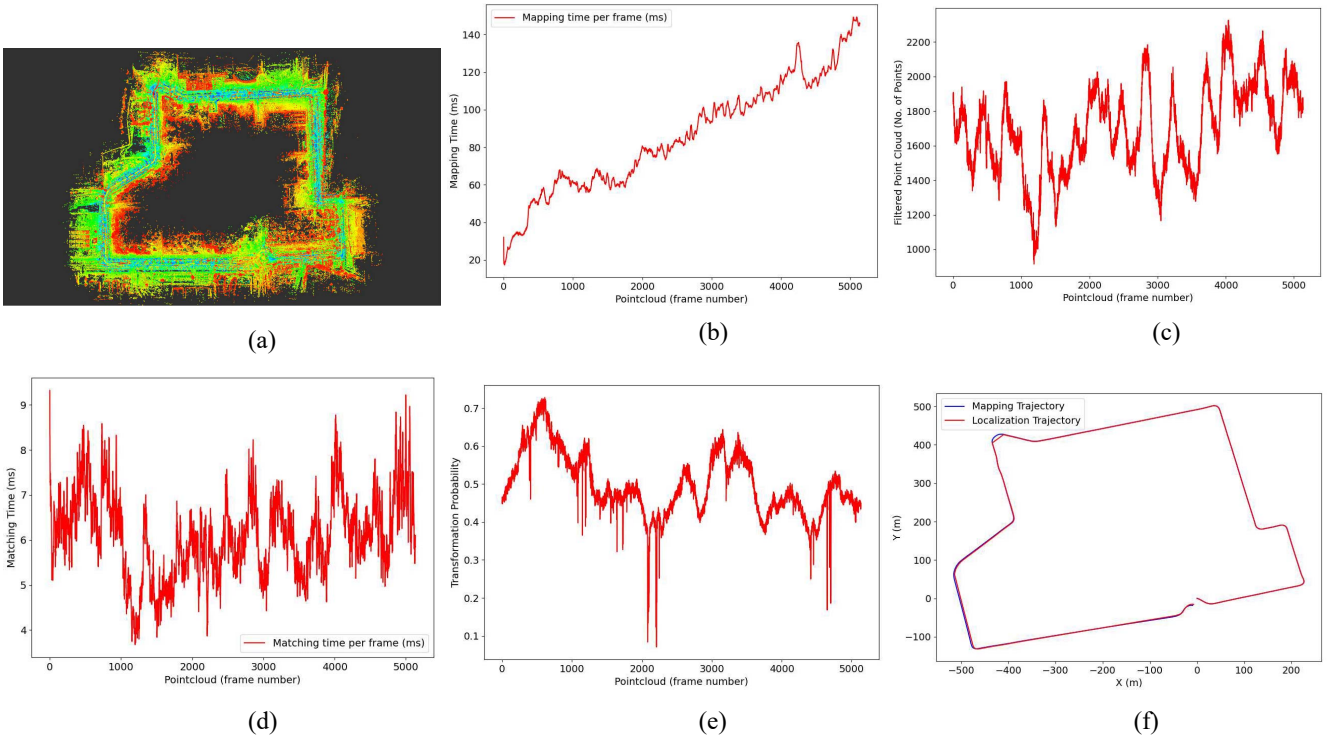

FIGURE 9. (a) Map. (b) Mapping time per frame. (c) Number of points in the filtered point cloud. (d) Matching time per frame during localization. (e) Transformation probability. (f) Mapping and localization trajectory of Route 3.

TABLE 1. Mapping results on different routes at IIT hyderabad campus using velodyne HDL-64 LiDAR.

Map Route	Total no. of point cloud frame	Average mapping time per frame (in ms)	No. of points in map's pcd file	Map file size (in MB)
Route 1	1445	33.85	64531	2.7
Route 2	6502	93.54	739968	29.9
Route 3	5132	80.32	647657	26.8

TABLE 3. Comparison of matching time per frame of the localization algorithm.

	Matching Time per frame in (ms)
Ours	6 ms
In [19]	30 ms
In [13]	90 ms

A. DISCUSSION AND COMPARISON

In Table 3, we have compared the matching time per point cloud frame on the map with the work done in [13] and [19]. Our optimized algorithm localizes the vehicle fifteen times faster than [13] and five times faster than [19]. Also, the work done in [11], [18], and [24] and other research work on the LiDAR-based localization did not mention the matching time per frame in their work.

Also, the experiment is performed on the Kitti odometry dataset sequence number 7 [26], which has ground truth trajectories. The experiment is performed using an HP i7-1165G7 system equipped with 16 GB of RAM. The map of the dataset is created, as shown in Fig. 10, and then the data is played back, evaluating the localization algorithm with the ground truth trajectory. The red trajectory represents the ground truth position of the vehicle during the data recording, the green trajectory depicts the position estimated by the traditional NDT algorithm, and the blue trajectory illustrates the optimized NDT algorithm using our approach. The results were assessed in terms of localization accuracy by comparing the trajectories in the root mean-square error and matching time per frame (ms), which is the time taken to match the point cloud data on a 3-D map per frame, as outlined in Table 4. We achieved a similar level of localization accuracy while achieving a five times faster localization process, as shown in Table 4.

The localization algorithm needs to outpace the data acquisition rate of the LiDAR sensor to run in real-time, which operates at 10 Hz (100 ms/frame). Also, the GPS sensors publish the data at 100 Hz (10 ms/frame). To achieve equivalency in competency with the GPS localization, the LiDAR-based localization algorithm should aim to localize under 10 ms/frame.

The demonstration of LiDAR-based optimized localization on 3-D map for autonomous navigation can be seen using this link: https://youtu.be/hPdYkcmbKTY.

V. CONCLUSION AND FUTURE WORK

This article proposes an optimized NDT localization algorithm on a 3-D map for an autonomous navigation system

TABLE 2. Localization and navigation results on different routes at IIT Hyderabad campus using Velodyne HDL-64 LiDAR.

Localization route	Total no. of point cloud frame	Filtered point per frame			Matching time per frame (in ms)			Transformation probability			Average Speed (km/hr)
		min	max	avg	min	max	avg	min	max	avg	
Route 1	1445	235	1572	1197.47	1.22	7.31	5.38	0.57	0.95	0.68	10.96
Route 2	6502	746	2474	1625.87	1.89	9.89	6.11	0.26	0.81	0.52	12.8
Route 3	5132	915	2326	1684.62	3.67	9.32	6.14	0.20	0.73	0.50	12.6

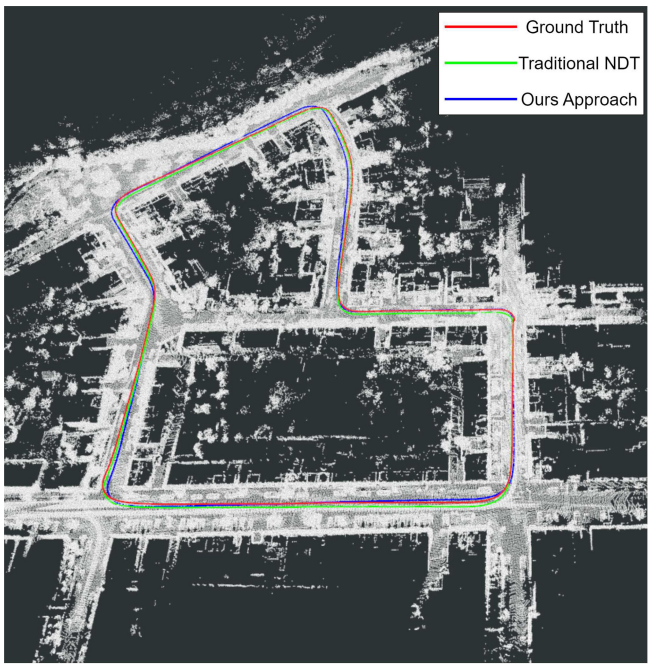

FIGURE 10. Localization algorithm is evaluated on Kitti odometry dataset sequence 7.

TABLE 4. Localization evaluation on kitti odometry dataset sequence 7.

	Average matching time (ms)	Localization Accuracy (rmse in cm)
Traditional NDT [19]	30 ms	2.13
Ours (Optimized NDT)	6.2 ms	2.17

using a LiDAR sensor. Mapping, localization, and navigation are tested on different routes at the IIT-Hyderabad campus. The mapping algorithm is made faster by updating the feature points based on the keyframes and increasing the edge and plane resolution. Mapping a large area of up to 2.5 km is done in real-time. The size of the pcd file of the map is very small, around 25 MB for a 2.5 km route. The lightweight map enabled smooth functionality of the localization algorithm within the ROS platform. In localization, the matching time depends on the number of points to be matched and how fast the point cloud converges and matches with the map. Point cloud data is filtered to reduce the matching time. The point cloud is filtered using a voxel grid filter of a leaf size of 3 to reduce the number of points to be matched. The localization algorithm is optimized, and the average matching time is found to be 5 and 6 ms per frame experimentally, which is much less than the data acquisition rate of LiDAR at 10 Hz, i.e., 100 ms per frame. The localization algorithm is made faster while achieving similar localization accuracy as evaluated on the Kitti odometry benchmark dataset. Hence, the localization algorithm is optimized and made faster for the smooth navigation of the autonomous vehicle.

The limitation of this method is in adverse weather conditions, such as rain or snow where the LiDAR data is affected. Multisensor fusion is a prospective area for further investigation in this project in the future. We intend to extend this research by fusing the LiDAR, camera, and IMU sensors to create an HD map and dynamically plan the path of the autonomous vehicle, incorporating obstacle tracking and avoidance.

REFERENCES

[1] M. Mielle, M. Magnusson, and A. J. Lilienthal, "A comparative analysis of radar and LiDAR sensing for localization and mapping," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, 2019, pp. 1–6, doi: 10.1109/ECMR.2019.8870345.

[2] Y. Li and J. Ibanez-Guzman, "LiDAR for autonomous driving: The principles, challenges, and trends for automotive LiDAR and perception systems," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 50–61, Jul. 2020, doi: 10.1109/MSP.2020.2973615.

[3] B. Anand, M. Senapati, V. Barsaiyan, and P. Rajalakshmi, "LiDAR-INS/GNSS-based real-time ground removal, segmentation, and georeferencing framework for smart transportation," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, Oct. 2021, doi: 10.1109/TIM.2021.3117661.

[4] S. McCrae and A. Zakhor, "3D object detection for autonomous driving using temporal LiDAR data," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2020, pp. 2661–2665, doi: 10.1109/ICIP40778.2020.9191134.

[5] B. Anand, V. Barsaiyan, M. Senapati, and P. Rajalakshmi, "Quantitative comparison of LiDAR point cloud segmentation for autonomous vehicles," in *Proc. IEEE 94th Veh. Technol. Conf. (VTC)*, 2021, pp. 1–4, doi: 10.1109/VTC2021-Fall52928.2021.9625507.

[6] M. Senapati, B. Anand, A. Thakur, H. Verma, and P. Rajalakshmi, "Object detection and segmentation using LiDAR-camera fusion for autonomous vehicle," in *Proc. 5th IEEE Int. Conf. Robot. Comput. (IRC)*, 2021, pp. 123–124, doi: 10.1109/IRC52146.2021.00029.

[7] S. Chen et al., "NDT-LOAM: A real-time LiDAR odometry and mapping with weighted NDT and LFA," *IEEE Sensors J.*, vol. 22, no. 4, pp. 3660–3671, Feb. 2022, doi: 10.1109/JSEN.2021.3135055.

[8] F. Huang, W. Wen, J. Zhang, and L.-T. Hsu, "Point wise or feature wise? A benchmark comparison of publicly available LiDAR odometry algorithms in urban canyons," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 6, pp. 155–173, Nov./Dec. 2022, doi: 10.1109/MITS.2021.3092731.

[9] B. Anand, H. Verma, A. Thakur, P. Alam, and P. Rajalakshmi, "Evaluation of the quality of LiDAR data in the varying ambient light," in *Proc. IEEE Sens. Appl. Symp. (SAS)*, 2022, pp. 1–5, doi: 10.1109/SAS54819.2022.9881373.

[10] Y.-C. Kan, L.-Ta Hsu, and E. Chung, "Performance evaluation on map-based NDT scan matching localization using simulated occlusion datasets," *IEEE Sensors Lett.*, vol. 5, no. 3, pp. 1–4, Mar. 2021, doi: 10.1109/LSENS.2021.3060097.

[11] N. Akai, L. Y. Morales, E. Takeuchi, Y. Yoshihara, and Y. Ninomiya, "Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2017, pp. 1356–1363, doi: 10.1109/IVS.2017.7995900.

[12] R. Wang, Y. Li, J. Fan, T. Wang, and X. Chen, "A novel pure pursuit algorithm for autonomous vehicles based on salp swarm algorithm and velocity controller," *IEEE Access*, vol. 8, pp. 166525–166540, 2020, doi: 10.1109/ACCESS.2020.3023071.

[13] E. Takeuchi and T. Tsubouchi, "A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 3068–3073, doi: 10.1109/IROS.2006.282246.

[14] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 2040–2047, doi: 10.1109/ICRA.2018.8460519.

[15] T. Ort et al., "MapLite: Autonomous intersection navigation without a detailed prior map," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 556–563, Apr. 2020, doi: 10.1109/LRA.2019.2961051.

[16] M. Á. Muñoz-Bañón, E. Velasco-Sánchez, F. A. Candelas, and F. Torres, "OpenStreetMap-based autonomous navigation with LiDAR naive-valley-path obstacle avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24428–24438, Dec. 2022, doi: 10.1109/TITS.2022.3208829.

[17] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," *Robot. Sci. Syst.*, vol. 2, no. 9, pp. 1–9, 2014.

[18] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal distributions transform monte-carlo localization (NDT-MCL)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 382–389, doi: 10.1109/IROS.2013.6696380.

[19] Z. Zhou et al., "NDT-transformer: Large-scale 3D point cloud localisation using the normal distribution transform representation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 5654–5660, doi: 10.1109/ICRA48506.2021.9560932.

[20] W.-J. Wang, T.-M. Hsu, and T.-S. Wu, "The improved pure pursuit algorithm for autonomous driving advanced system," in *Proc. IEEE 10th Int. Workshop Comput. Intell. Appl. (IWCIA)*, 2017, pp. 33–38, doi: 10.1109/IWCIA.2017.8203557.

[21] A. Thakur, B. Anand, H. Verma, and P. Rajalakshmi, "Real time LiDAR odometry and mapping and creation of vector map," in *Proc. 8th Int. Conf. Autom., Robot. Appl. (ICARA)*, 2022, pp. 181–185, doi: 10.1109/ICARA55094.2022.9738576.

[22] H. Wang, C. Wang, C. L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 4390–4396, doi: 10.1109/IROS51168.2021.9636655.

[23] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 5135–5142, doi: 10.1109/IROS45743.2020.9341176.

[24] A. Carballo et al., "Characterization of multiple 3D LiDARs for localization and mapping performance using the NDT algorithm," in *Proc. IEEE Intell. Veh. Symp. Workshops (IV Workshops)*, 2021, pp. 327–334, doi: 10.1109/IVWorkshops54471.2021.9669244.

[25] J. Becker, N. Imholz, L. Schwarzenbach, E. Ghignone, N. Baumann, and M. Magno, "Model- and acceleration-based pursuit controller for high-performance autonomous racing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023, pp. 5276–5283, doi: 10.1109/ICRA48891.2023.10161472.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361, doi: 10.1109/CVPR.2012.6248074.

ABHISHEK THAKUR (Graduate Student Member, IEEE) received the bachelor's degree in electronics and communication engineering from the Indian Institute of Information Technology Manipur, Manipur, India.

He is currently a Research Scholar with the Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Hyderabad, India. His research interests include autonomous navigation, LiDAR data processing, and sensor fusion for autonomous vehicles.

P. RAJALAKSHMI (Senior Member, IEEE) received the Ph.D. degree from the Indian Institute of Technology Madras, Chennai, India, in 2008.

She is currently a Professor with the Indian Institute of Technology Hyderabad, Hyderabad, India. With more than 180 internationally reputed research publications, her research interests include autonomous vehicles, HD maps, drone-based sensing, LiDAR-based traffic sensing, artificial intelligence, cyber–physical systems, and wireless and sensor networks.