IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 June 2024, accepted 30 June 2024, date of publication 5 July 2024, date of current version 15 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3424234

RESEARCH ARTICLE

A Novel Method for Ground Vehicle Tracking With Error-State Kalman Filter-Based Visual-LiDAR Odometry (ESKF-VLO)

SUDEEPTA R. SAHOO AND P. V. MANIVANAN
Mechanical Department, IIT Madras, Chennai, Tamil Nadu 600036, India
Corresponding author: Sudeepta R. Sahoo (me17d018@smail.iitm.ac.in)

ABSTRACT An autonomous vehicle requires a self-tracking system that can operate both indoors and outdoors using only on-board sensors. In this work, a vehicle self-tracking scheme called Error-State Kalman Filter-Based Visual-LiDAR Odometry (ESKF-VLO) that uses camera, LiDAR, and IMU sensors is proposed. In this scheme, to obtain the 3D vehicle transformation between two consecutive time frames, the corresponding feature points are detected in the captured camera images and the respective 3D location of the feature points is obtained from LiDAR points. The relationship between 3D LiDAR points and camera image pixels is established using the sensors' LiDAR-to-Camera calibration parameters. In the final step, IMU data is corrected with this transformation matrix using the Error-State Kalman Filter, and the vehicle position is tracked accurately. The proposed method's efficacy is verified using the KITTI dataset and data obtained through indoor experiments conducted using the P3DX mobile robot. The results show that the proposed method is able to track the vehicle, and the translational and rotational errors are reduced compared to the existing methods.

INDEX TERMS LiDAR, optical flow, road detection, error state Kalman filter, visual-LiDAR odometry (VLO).

I. INTRODUCTION

While autonomous road vehicles are slowly gaining importance in transporting people, autonomous mobile robots are also being increasingly used to transport goods to places where humans may not be able to reach (like hazardous and harsh environments). Hence, one of the important research topics in autonomous vehicle development is developing suitable vehicle navigation systems. Vehicle navigation also includes the vehicle's ability to track its own motion (state estimation) and simultaneously map the environment [1]. Traditionally, the current vehicle state is estimated using GPS or wheel encoder data, and both have their advantages and disadvantages. Though the GPS system works properly in an outdoor environment, its ability to estimate vehicle position accurately reduces or completely stops in the case of indoor applications or GPS signal-denied locations, as the reception of the GPS signal becomes very poor [2]. When using a wheel encoder for state estimation, wheel

TABLE 1. Acronyms details.

Acronym	Full Name
LiDAR	Light Detection and Ranging
GPS	Global Positioning System
IMU	Inertial Measurement Unit
ESKF	Error-State Kalman Filter
VLO	Visual LiDAR Odometry
IPS	Indoor Positioning System
WPS	Wireless Positioning System
IR	Infrared
SLAM	Simultaneous Localization and Mapping
DCNN	Deep Convolutional Neural Network
FCN	Fully Convolutional Network
RGB	Red, Green, Blue
VO	Visual Odometry
EKF	Extended Kalman Filter
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
P3DX	Pioneer 3-DX

The associate editor coordinating the review of this manuscript and approving it for publication was Ángel F. García-Fernández.

slip reduces accuracy [3]. Further, state-estimating sensors can be divided into two types: sensors that give absolute positioning with some position noise and sensors that give incremental positioning. GPS, IPS [4], and WPS [5] sensors usually provide absolute vehicle position values through signal triangulation techniques, but this requires external infrastructure (GPS satellites, anchors, cameras, etc.). Apart from these sensors, LiDAR, camera, sonar, IR sensors, etc., are used to estimate the incremental vehicle state (position, velocity, and orientation values) [6], and these sensors are directly mounted on the vehicle and do not require any external infrastructure for their operation. However, the noise in these sensors produces an error at every timestamp (known as incremental error), and this incremental error accumulates over time, thereby affecting the estimated path accuracy. Hence, depending on requirements (accuracy of state estimation, application, cost of sensors, etc.), the autonomous vehicle can be equipped with any one of the above sensors or multiple sensors to estimate the vehicle position accurately by using data fusion techniques to reduce the incremental error of vehicle odometry. Using multiple sensors also reduces sensor failure-related navigation failures.

In this work, an incremental location-finding technique that fuses data from LiDAR and a vision sensor (monochrome camera) is used for vehicle state estimation. The above estimated vehicle state is corrected using IMU sensor data with the help of an Error-State Kalman Filter to minimize the incremental error. LiDAR, which stands for Light Detection and Ranging, is one of the most widely used perception sensors in the navigation of autonomous vehicles. LiDAR detects and estimates the distance of any object in front of it by illuminating the object and receiving the reflected laser light pulses with multiple laser transceivers. LiDAR estimates the object's distance more accurately compared to the ultrasonic sensor and IR sensor [7]. Also, LiDAR-based SLAM is the best tool for vehicle navigation [8]. Additionally, a variety of algorithms are available for SLAM that utilize neural networks for state estimation; however, for this neural network-based approach, a large amount of pre-data is required to train the system. Further, in the case of SLAM, the whole environment data need to be stored as a map and processed at every instance. Processed current point clouds are compared with an earlier stored map to estimate vehicle position [9]. Similarly, using vision data (camera) also makes vehicle localization possible; however, this method is also computationally expensive.

Robust segmentation of flat surfaces (can be a road surface or flat floor in the case of an indoor environment) from the total environment is a crucial and challenging part of the algorithm that will be developed in this work. Many researchers have primarily used images from a monocular camera for road area segmentation. In the past, road section detection methods have applied various edge detection methods to achieve better performance [10] (for lane marking detection). However, presently deep convolutional neural network (DCNN) approaches such as FCN [11] and DeepLab [12] are being used to obtain much improved performance. Despite the progress made in image-based ground segmentation methods, they have their own disadvantages.

Lane-based road segmentation can't be used in the indoor environment, due to the non-availability of fixed lane markings and also in the case of rural unstructured roads without lane markings. Further, the performance of DCNN deteriorates with visual noise present in the captured image, which can be caused by overexposure, changing lighting, ambiguous illusions, and blurring of images as described in [13]. In indoor applications, obstacles with RGB values similar to the ground plane pose challenges for vision-based flat surface segmentation. Hence in the present work, to avoid these issues and to improve the road detection performance, the LiDAR sensor along with the spatial-based approach (eliminate effect of illumination) is used to perform the flat surface segmentation as in [14].

II. RELATED WORK

Navigating the mobile robot using a monocular or stereo camera is known as Visual Odometry (VO)-based navigation. The authors of [15] have estimated the robot motion using the visual information (feature correspondence) available from the camera. Various methods to determine the relative movement of the frame have been implemented. Those methods can be classified as direct methods (minimizing the photometric error) [15], [16], [17], [18], [19], indirect methods (minimizing the reprojection error) [20], [21], [22], [23], and semantic mapping method (segmenting image and estimating via known pose) [24], [25], [26], [27]. Moreover, the accuracy of the VO can be influenced by camera imaging quality, lighting conditions, and illumination variations. Furthermore, camera calibration parameters may vary due to vibration induced while maneuvering [28].

Besides VO-based navigation, researchers have also explored LiDAR-based navigation systems for autonomous navigation in indoor and outdoor environments [29]. There are two types of LiDARs, namely: 2D LiDAR and 3D LiDAR. The 2D LiDAR consists of a single scanning layer and performs well in indoor applications such as SLAM [9], [30]. Similarly, mapping and localization of the environment have been done using 3D LiDAR [31], [32]. Both 2D and 3D LiDAR create point clouds of the environment continually, and the system matches the geometrical structure obtained from the current point clouds with the structure associated with the previously obtained point cloud for mapping and localization. Other sensors like IMU [33], GPS, or wheel encoder readings are often used to improve the accuracy of LiDAR odometry by an extended Kalman filter [34] or particle filter [35]. The primary advantage of LiDAR-based localization is that as the LiDAR scanning rate is very high compared to the vehicle motion, distortion due to motion can be neglected [31]. The drawback of LiDAR-based SLAM is the high computational cost due to expensive correspondence point searching [36].

Hence, this paper proposes a hybrid approach that uses both LiDAR and camera sensors to compute the 3D corresponding points, instead of the geometrical structure. The 3D corresponding points are computed by estimating the depth details of the matched corresponding feature points from the camera image. Moreover, to estimate the depth of feature points, a fluctuating point cancellation algorithm is used to segment the ground plane from the LiDAR point cloud. Subsequently, using the ground flat surface LiDAR points, the 3D location of 2D matched corresponding features is obtained. Finally, the transformation matrix is calculated for the 3D corresponding points at every time instant; thereby the vehicle motion is detected and recorded.

III. METHODOLOGY

The objective of this paper is to estimate vehicle motion in the form of the transformation matrix at every instant of movement (i.e., at time t to time $t + \delta t$) by fusing the LiDAR point cloud and camera image data. The transformation matrix of the vehicle coordinate system from time t to time $t + \delta t$ is calculated from the 3D corresponding points. These corresponding points are represented as 3D coordinate values $\left(r_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}\right)$ with respect to vehicle coordinate systems for the above two time stamps. Before starting the procedure, an extrinsic calibration has been done to compute the rigid transformation matrix, which provides the relationship between the LiDAR coordinate frames and the camera image pixel coordinate frame. At each timestamp, the LiDAR generates point cloud information, and the camera produces an RGB image. While the LiDAR point cloud data provides 3D depth information, it does not provide the corresponding points between the timeframes. However, RGB images are able to provide pixel corresponding points, not the depth information. In the developed algorithm, each step begins with detecting the flat ground surface by removing fluctuating points (non-flat sections) from the LiDAR point cloud data. This flat section (consisting of cloud points) is then projected over images (captured at the same time instant) using previously calculated calibration parameters and a dataset is prepared for LiDAR 3D points presented in the image with their 2D image pixels.

Using feature selection and matching methods between two captured images at timestamp t and $t + \delta t$, the image pixel corresponding points are identified. These image corresponding points may not have their 3D value (depth information) from LiDAR, so an interpolation method is used to convert pixel values to 3D points using the prepared dataset. The interpolation method only works for points presented on a flat surface. Hence, the flat surface is detected beforehand and the same flat surface is projected over the image. After obtaining the 3D corresponding environment points between the two LiDAR coordinate frames (using the previously generated dataset), the vehicle transformation matrix (vehicle motion) between time t to time $t+\delta t$ ($T_t^{t+\delta t} \in R^{3\times4}$) is calculated. As the LiDAR is rigidly fixed on the vehicle with a pitch angle of (17^o) with respect to the vehicle frame, the relationship between the vehicle frame and LiDAR frame involves only rotation with zero translation.

By fusing IMU sensor data and the previously obtained vehicle transformation matrix ($T_t^{t+\delta t}$) obtained using the camera and LiDAR, the vehicle state (x) is estimated (consisting of vehicle position, velocity, and quaternion) using an error state Kalman filter. The IMU sensor data (which consists of both the linear acceleration and angular velocity data) is used to predict the vehicle state ($\check{x}$) and $T_t^{t+\delta t}$ is used to correct the predicted vehicle state ($\check{x}$) as part of the Error State Kalman Filter method. By accumulating the vehicle position value (from the vehicle state) at each timestamp, vehicle odometry is estimated; therefore it is called Error-State Kalman Filter Visual-LiDAR Odometry (ESKF-VLO). The entire procedure of the method is shown as a flowchart in Figure 1.

IV. LIDAR-BASED FLAT GROUND PLANE SEGMENTATION

LiDAR (Light Detection and Ranging) is a sensor that estimates the distance between itself and an object in front as points; i.e., by sending single/multiple pulsed LASER beams (called channels) and capturing the reflected light pulses using photo sensors. Using a technique called time-of-flight, the LiDAR estimates the distance from the individual reflected pulse to the object's surface and generates a 3D environment map. Velodyne LiDAR has built-in laser heads (varying from 1 to 128 channels) and a motor that gives rotary motion to those laser heads. This section provides the mathematical approach to segregating flat surfaces from the ground plane in LiDAR point clouds. A multi-channel LiDAR sensor carries multiple laser scanners, as many as the number of its channels, usually installed vertically inside the sensor. To tackle the task of flat surface detection, the LiDAR data are rearranged to meet a presentation mode with spatial relationship and low complexity. The calibration file provided by the manufacturer contains the pitch angle ϕ of each laser head. The output of LiDAR data point clouds is in Cartesian form and in the file format of '*.pcd*'. This LiDAR detects 1.33 million sample points per second, so it is necessary to rearrange the 3D LiDAR data before processing them.

Each LiDAR point cloud can be further subdivided into a number of rings for different pitch angles or laser heads (N - number of laser heads) as shown in Figure 2. Each ring has a different radius (ρ) and polar angle (θ). Therefore, each Cartesian form of the point cloud is rearranged for each laser head 'n' (n varies from 1 to N) and the polar angle (θ varies from $-\pi$ to $+\pi$), as seen in Figure 3. The relationship between the Cartesian form (x, y, z) and the polar form (θ, ρ, and z) is expressed in Eqs. 1, 2, and 3, and shown graphically in Figure 4.

$$\theta = \tan^{-1}\frac{x}{y} \tag{1}$$

$$\rho = \sqrt{x^2 + y^2} \tag{2}$$

$$z = z \tag{3}$$

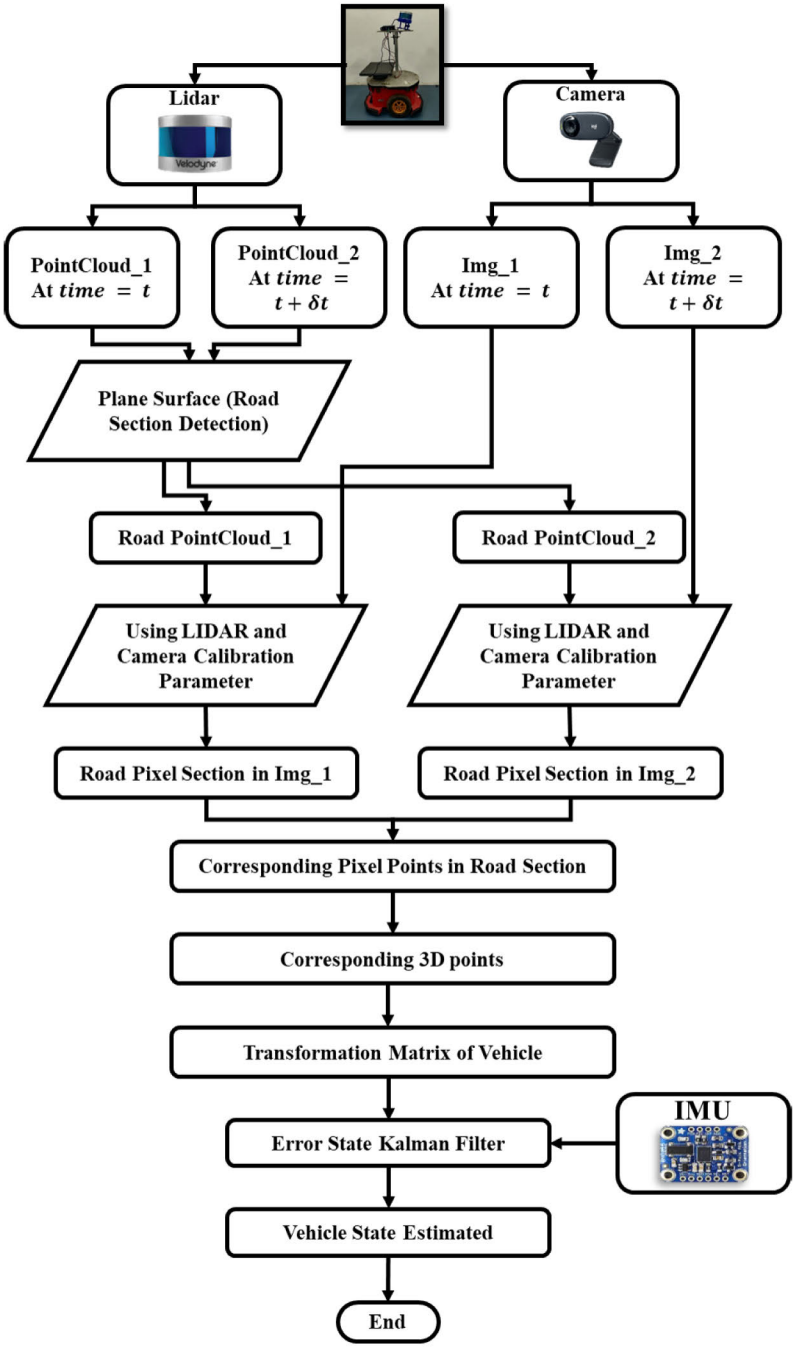

FIGURE 1. State estimation through lidar and camera data: A concise overview of the comprehensive procedure.

The external environment scanned by the LiDAR consists of flat road sections, shoulders with grass, and objects such as vehicles, pedestrians, roadside buildings, trees, etc. Further, in the given environment, the ground plane can be defined as the flat road surface and the areas closer to the road with minimal undulations (i.e., road shoulders). Figure 5 shows the total point cloud data received from all the channels of LiDAR (in this case, LiDAR channels 'n' vary from 1 to N). In the plotted point cloud data, it can be noticed that the flat road section has minimal fluctuations in the Z-direction, while the road shoulders and nearby areas have more fluctuation in the Z-direction. Also, some of the cloud points will have fluctuations in the Z-direction and along with angle variation, when that particular cloud point represents the obstacle surfaces (i.e., other than flat road surface) as shown in Figure 5. In the indoor case, the ground plane consists of flat surfaces for vehicle free movement and obstacles have some angle with respect to the flat surface in the Z-direction.

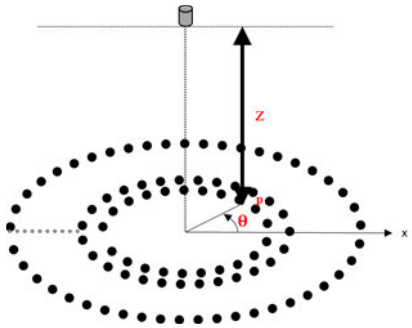

FIGURE 2. LiDAR data segmentation into rings for nuanced analysis in cylindrical coordinates.

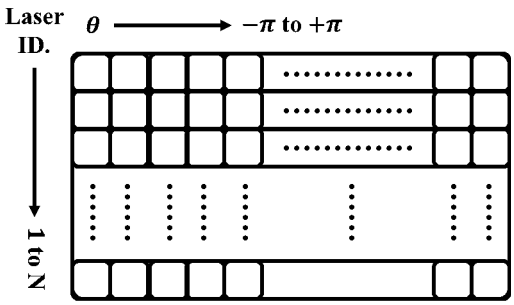

FIGURE 3. LiDAR data points structured in a table for exploration and analysis.

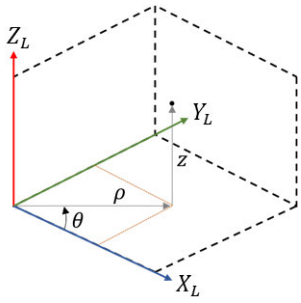

FIGURE 4. LiDAR sample point representation (cartesian and polar coordinates).

The Figure 6a shows the individual LiDAR channel point cloud data in three dimensional Cartesian space. By looking closely at this plot, it is possible to distinguish a flat surface (road section) which has minimal fluctuations in the Z-direction (shown in green colour), compared to the other cloud points (in magenta colour). By converting cloud point data into Polar form and plotting it as a 2D-graph between θ and z (Figure 6b), it's even easier to differentiate the road section with minimum fluctuation (green colour line segment) and the fluctuation part representing non-road section (Magenta colour). Mathematically in this fluctuating part there is a rapid change in the slope value. Using this property for each layer, the fluctuating part is separated from the linear part (green section) shown in Figure 6. The flat surface segment (green segment) is separated from the ground (Magenta colour section), by joining the linear parts for each layer (for each channel), which is shown in the Figure 7.

V. LIDAR POINTS TO IMAGE PIXEL - TRANSFORMATION

This section gives relationship between LiDAR point cloud data and Camera image pixel coordinate frame in the form of transformation matrix (K) that helps to project LiDAR cloud points ($\bar{Q}$) to image pixel points ($\bar{q}$), and the relationship is represented as the Eq.4. This rigid transformation matrix K consists of rotation matrix (R) and translation vector (τ) and it is a product of two parameters (Intrinsic matrix and extrinsic matrix) - Eq. 5.

$$K \times \bar{Q} = \bar{q} \tag{4}$$

$$K = K_{in} \times K_{ex} \tag{5}$$

where

- $\bar{Q} \equiv \begin{bmatrix} \bar{x}_l & \bar{y}_l & \bar{z}_l & \bar{\omega}_l \end{bmatrix}^T \rightarrow$ Generalized LiDAR Co-ordinate points
- $Q \equiv \begin{bmatrix} x_l & y_l & z_l \end{bmatrix}^T = \begin{bmatrix} \frac{\bar{x}_l}{\bar{\omega}_l} & \frac{\bar{y}_l}{\bar{\omega}_l} & \frac{\bar{z}_l}{\bar{\omega}_l} \end{bmatrix}^T \rightarrow$ LiDAR Co-ordinate points
- $\bar{q} \equiv \begin{bmatrix} \bar{x}_i & \bar{y}_i & \bar{\omega}_i \end{bmatrix}^T \rightarrow$ Generalized Image Pixel Coordinate points
- $q \equiv \begin{bmatrix} x_i & y_i \end{bmatrix}^T = \begin{bmatrix} \frac{\bar{x}_i}{\bar{\omega}_i} & \frac{\bar{y}_i}{\bar{\omega}_i} \end{bmatrix}^T \rightarrow$ Image Pixel Coordinate points
- $K_{in} \in IR^{3\times4} \rightarrow$ Camera Intrinsic Parameter Transformation Matrix
- $K_{ex} \in IR^{4\times4} \rightarrow$ Extrinsic Parameter Transformation Matrix
- $K \in IR^{3\times4} \rightarrow$ LiDAR points to image pixel Transformation Matrix

The transformation matrix (K) is formed using Intrinsic matrix (K_{in}) and Extrinsic matrix (K_{ex}) (Eq.5), which gives the relationship between three coordinate systems i.e. World / LiDAR Coordinate $\left(\begin{bmatrix} x_l & y_l & z_l \end{bmatrix}^T\right)$ with Camera 3D coordinate $\left(\begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T\right)$ and Camera 3D co-ordinate $\left(\begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T\right)$ with the Camera Pixel coordinate $\left(\begin{bmatrix} x_i & y_i \end{bmatrix}^T\right)$ respectively. This can be easily understood by Figure 8.

The intrinsic and extrinsic parameters is calculated by calibrating both the Camera and LiDAR using a checker-board-like pattern and each box size is of 50 × 50 mm. For better calibration, multiple checker-board Camera images (in *.png* format) and several LiDAR point clouds (in *.pcd* format) were captured using MATLAB code, with different checkerboard positions at the same timestamp.

In the first step, the Camera intrinsic parameter is calculated using the method in [34] by automatically detecting the checkerboard section in all images. Camera intrinsic parameters are: focal length, principal point, image size, radial distortion, tangential distortion, skew, and intrinsic matrix. The MATLAB Single-Camera Calibration App makes it easy to calculate intrinsic parameters. By removing faulty images, optimized Camera intrinsic parameters for low reprojection error are calculated. Initially, a batch

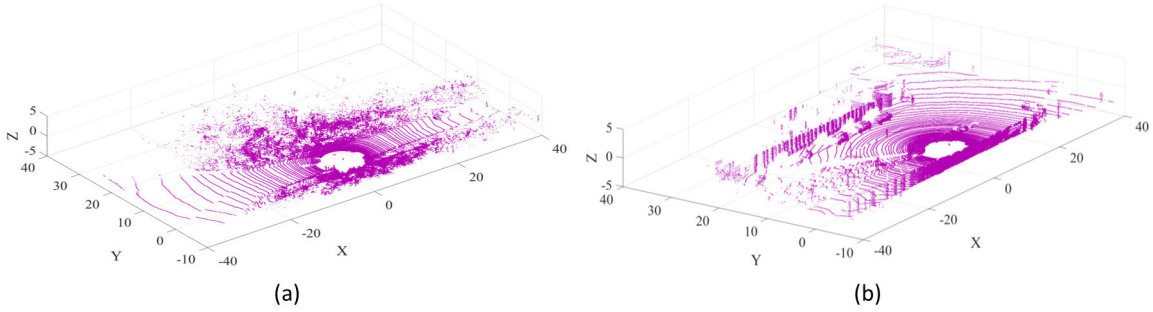

FIGURE 5. LiDAR point cloud at any instant timestamp (with KITTI Dataset).

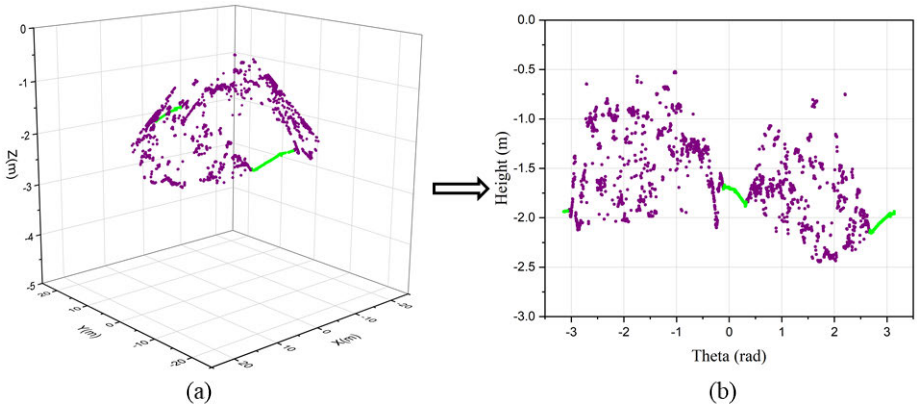

FIGURE 6. LiDAR Single Layer view (a) Cartesian Co-ordinate (x-y-z) and (b) Polar (θ vs z).

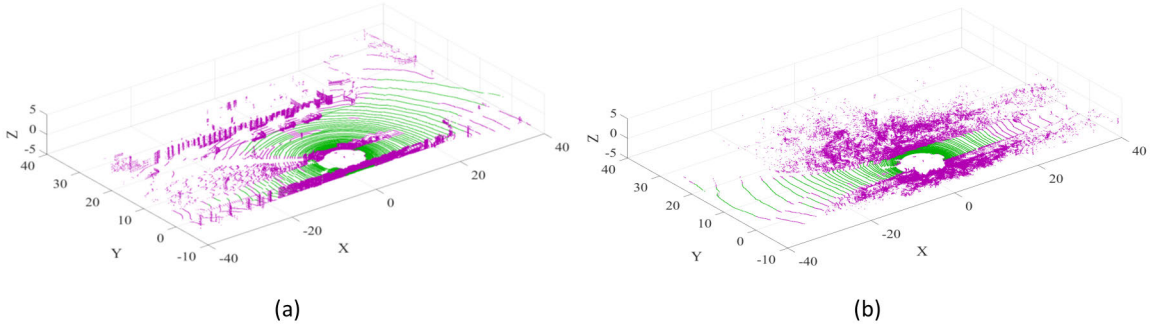

FIGURE 7. Road detection in LiDAR point cloud data highlighted in blue (KITTI Dataset).

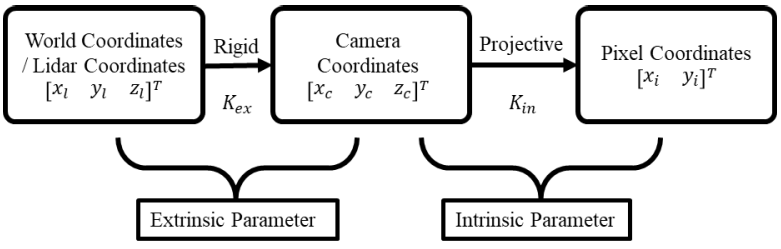

FIGURE 8. Formation of Transformation Matrix (K) merging Intrinsic and Extrinsic parameters.

of 33 images with checkerboards were considered for calibration; however, only 25 images were used as the eight images found to be defective. Finally, the mean error of those 25 images are plotted as a bar diagram (Figure 9)

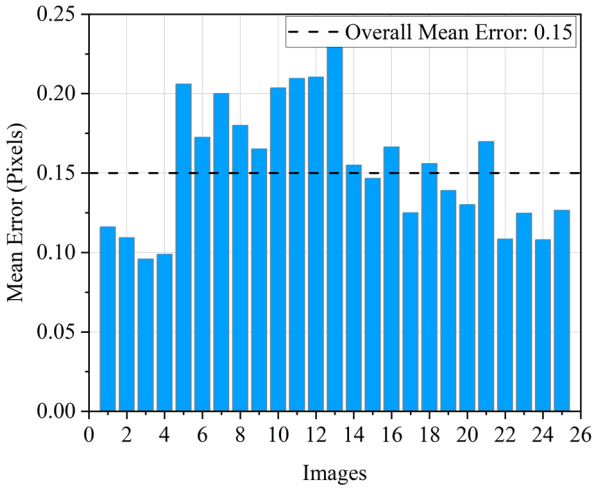

FIGURE 9. Calculating intrinsic parameters: mean reprojection error analysis.

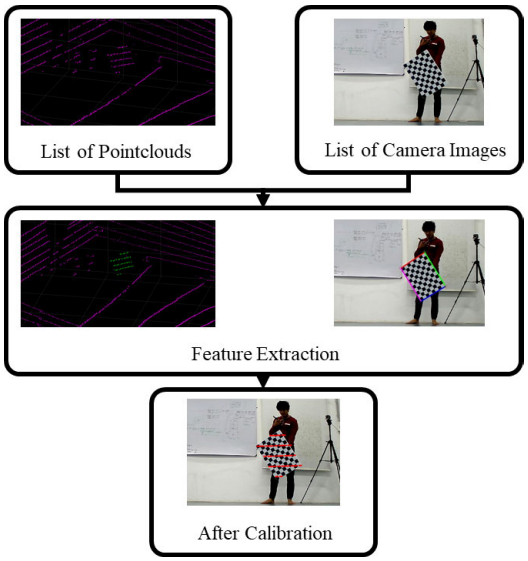

FIGURE 10. Mapping extrinsic parameters: calculating transformation from camera frame to LiDAR coordinates.

and overall mean error of the calculated intrinsic parameters is 0.15.

In the next step, extrinsic parameters are calculated, which is nothing but a rigid transformation matrix. This transformation matrix gives the relationship between the LiDAR coordinate and Camera image pixel coordinate. Similar to intrinsic parameter calibration, checkerboard selection and matching method are used to calculate extrinsic parameters. In the present case, the MATLAB Toolbox for automatic checkerboard detection is not able to detect the checkerboard, as there are fewer number cloud points available above the checkerboard (this is due to the LiDAR (Velodyne VLP16) used has less number of LiDAR channels i.e., 16-channels only). Therefore the checkerboard cloud points are traced manually for each LiDAR point cloud. Initially, 30 combinations of LiDAR point cloud and image datasets were considered. Using the method described in [37], the extrinsic parameters and the rigid transformation matrix were calculated. The calibration process involved iteratively removing datasets with higher translational and rotational errors. The calibration was concluded with an overall translational error of 0.079m and a rotational error of 1.7801 degrees, resulting in 13 datasets being used. The rigid transformation matrix (K) consists of both intrinsic parameter (K_{in}) and extrinsic parameter (K_{ex}) is given below:

$$K = \begin{bmatrix} 0.4112 & 0.9111 & 0.0082 & 0.0395 \\ -0.12 & 0.0631 & -0.9908 & -0.2572 \\ -0.9033 & 0.4072 & 0.1353 & 0.0946) \end{bmatrix}$$

The overall procedure used for extrinsic parameter calculation is shown in the Figure 10

To check the accuracy of the above procedure that calculates extrinsic parameter, three types of errors mentioned below are calculated using the method available in [37] and presented as a bar charts (Figure 11).

1) Translation Error (Overall mean Error = 0.079019 m)
2) Rotation Error (Overall mean error = 1.7801 degree)
3) Re-projection Error (Overall mean error = 13.8397 pixel)

Using the transformation matrix (K) calculated above, earlier detected road surface's LiDAR cloud points are projected onto the corresponding Camera image (that was captured at the same time stamp at which LiDAR point cloud was recorded). This projection operation gives a correspondence between 3D LiDAR points and Image 2D pixel points for the road surface.

VI. OPTICAL FLOW TO GET IMAGE CORRESPONDING POINTS

Normally, video images captured in each successive timestamp will have number of common pixel points among themselves. These common points are identified by the feature selection & matching algorithm, otherwise known as optical flow algorithm. Texture information is assigned as feature value to each pixel and the pixels with high feature value are considered Interesting Feature Point (IFP). To find these Interesting Feature Points (IFPs), the minimum Eigen value method [38] is used. In this method the Hessian matrix is computed for all the pixel points, then its Eigen value matrix is computed. The Hessian matrix (C) is calculated using its surrounding pixel values (grayscale values) for each image pixel point using the Eq. 6.

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_y I_x \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \tag{6}$$

where, I_x and I_y are the grey-scale value of pixels in the 'x' and 'y' direction from the current pixel. After computing the Hessian matrix, its eigenvalues λ_1 and λ_2 are calculated. Based on the values of λ_1 and λ_2, three cases are possible:

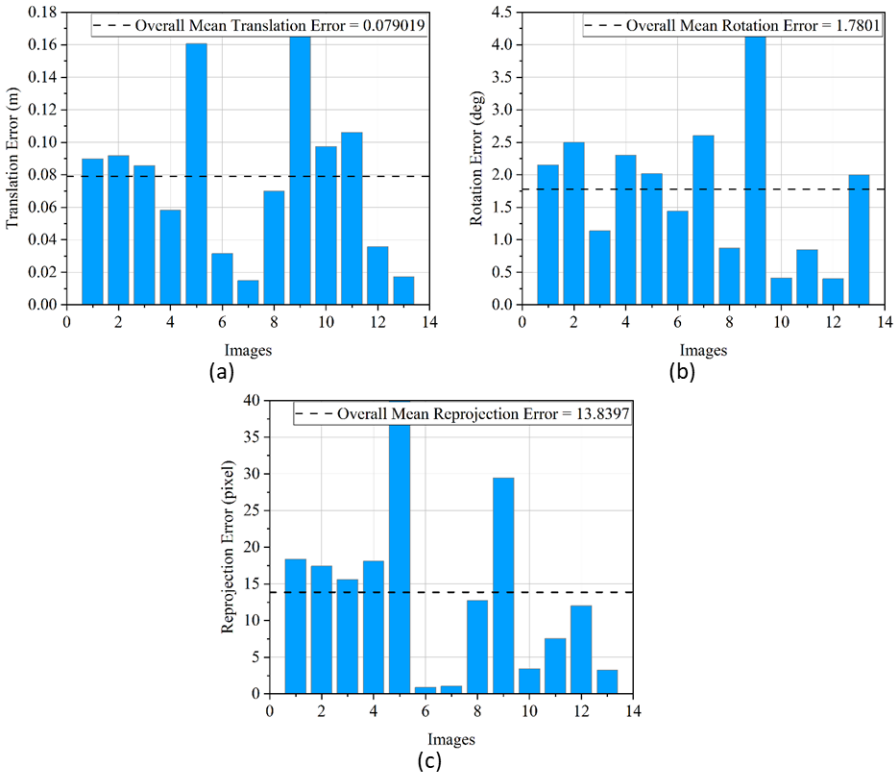

FIGURE 11. LiDAR and camera extrinsic parameter (a) translation error (b) rotation error (c) reprojection error.

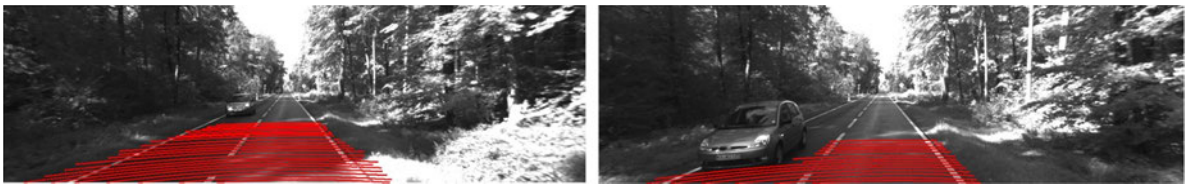

FIGURE 12. Detected road for two frames (Road LiDAR points projected over the camera images).

1) When both λ_1 and λ_2 are very small; The Hessian matrix is constant in all directions and the pixel point is a flat area and cannot be considered as interesting feature point.
2) Either λ_1 and λ_2 is high and this is the condition of Edge pixel and can be considered as interesting feature point.
3) Both λ_1 and λ_2 are large, the Hessian matrix increases in all directions, so this is the corner point condition and is considered as the interesting feature point.

As mentioned earlier, the Interesting Feature Points (IFPs) depend on the texture information presented in the images and the flat road segment usually has less texture information. Hence, image correction (contrast adjustment by histogram equalization [39]) is needed to reveal the hidden texture information as following method. The results of histogram equalization of the both the original and corrected images are shown in the Figure 13.

After finding the Interesting Feature Points (IFPs) of the images captured at time t and $t + \delta t$ (say img_1 and img_2), its feature value matrix is calculated with the method described in [31]. However, as every vehicle has a speed limit, the feature points of Camera image can't move more than a particular radius in next frame. Hence, feature value matching has to be done within this radius using [40] to find the corresponding points (pixels) in the img_1 and img_2.

VII. INTERPOLATION USING DELAUNAY TRIANGULATION

Figure 12 shows the projected LiDAR cloud points on the camera image. The image pixels on which the cloud points are projected have 3D coordinate values with respect to the LiDAR coordinate frame $\{L\}$. However, these pixel points may not be the interesting feature points. Therefore, a 2D interpolation method is used to find the 3D location of matched interesting feature points of both images

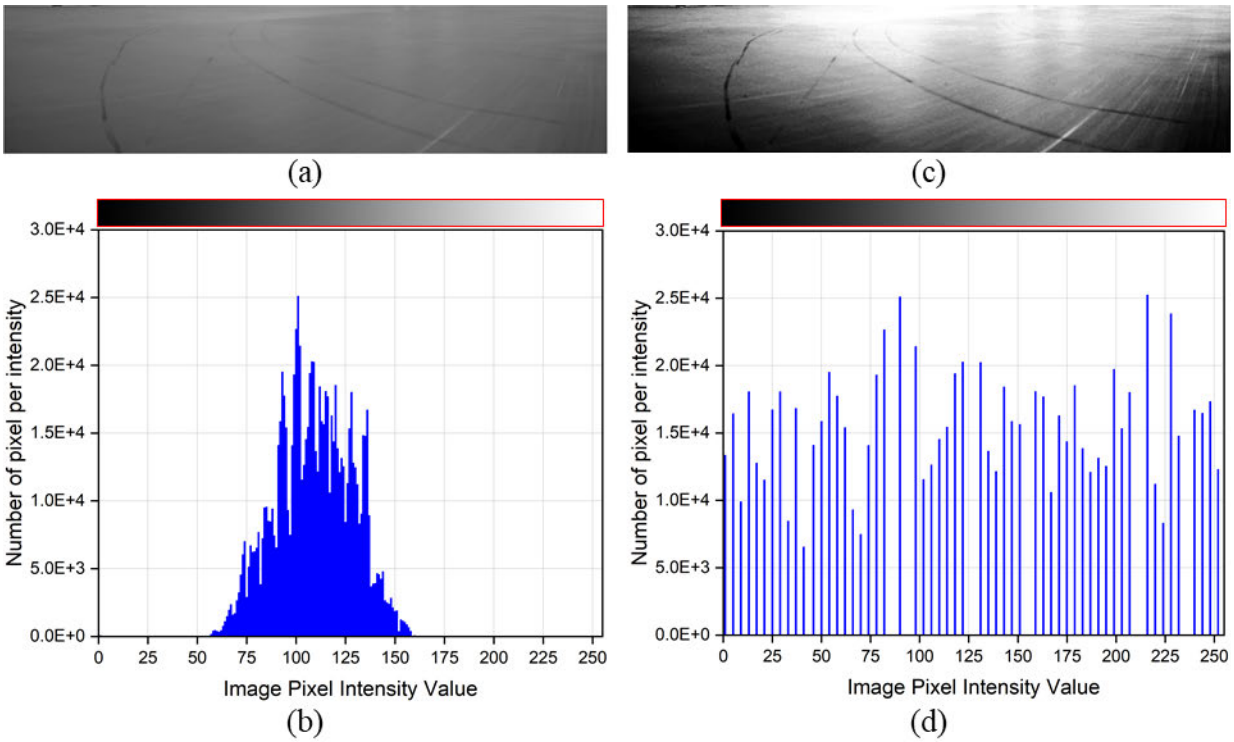

FIGURE 13. Image correction results (a) Original image (b) Corrected image (c) Histogram diagram for original image (d) Histogram diagram for corrected image.

(between img_1 and img_2), and this interpolation method can only work for flat surfaces. As described in the previous section, the flat surface (road surface) has already been separated from the aggregate point clouds. For these flat sections, applying interpolation to the two matching interesting feature points set (from img_1 and img_2), the 3D corresponding points with respect to the LiDAR coordinate system are found. MATLAB provides a function to perform interpolation called `scatteredInterpolant` on 2D or 3D scattered data sets. This function returns an interpolant F for the given data set and utilizes Delaunay triangulation, a powerful tool for interpolation. The process is done separately for x, y, and z data sets, which offers several advantages. Advantages of Separating x, y, and z Components for Interpolation are below:

1. **Preserves Locality**: Delaunay triangulation in 2D considers geometric relationships between points in the x-y plane. This local property is crucial for accurate interpolation, especially with non-uniformly distributed data.

2. **Computational Efficiency**: Separate triangulations are more efficient to compute,

A. UNDERSTANDING DELAUNAY TRIANGULATION

Delaunay triangulation is a geometric technique for subdividing a set of points in a plane (2D) or space (3D) into triangles (2D) or tetrahedra (3D). It ensures that no point lies inside the circumcircle (2D) or circumsphere (3D) of any other triangle or tetrahedron in the triangulation. This results in well-shaped elements and avoids elongated or skinny triangles/tetrahedra, which can lead to interpolation errors. The complete process is explained from [41] Steps for Interpolation with Delaunay Triangulation is explaned below.

STEP 1: TRIANGULATE POINTS IN THE X-Y PLANE

- For all 2D image points (x_i, y_i) and each corresponding lidar point (x_l, y_l, z_l), set up the interpolation for each lidar coordinate (x_l, y_l, or z_l).
- Triangulate the points in the x_i, y_i plane to create a piecewise triangular surface over the plane.

STEP 2: DEFINE TRIANGLE VERTICES AND VALUES

- Each triangle's vertices have coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3) and values z_1, z_2, z_3.
- The value z at any point P within a triangle is determined using the plane equation $z = ax + by + c$.

STEP 3: SOLVE FOR PLANE COEFFICIENTS

- Form a system of linear equations using the coordinates and values of the vertices:

$$\begin{cases} z_1 = ax_1 + by_1 + c \\ z_2 = ax_2 + by_2 + c \\ z_3 = ax_3 + by_3 + c \end{cases}$$

- Solve for the coefficients a, b, c.

Applying the method described above to the x_l, y_l, and z_l coordinates, and repeating the process for the matched

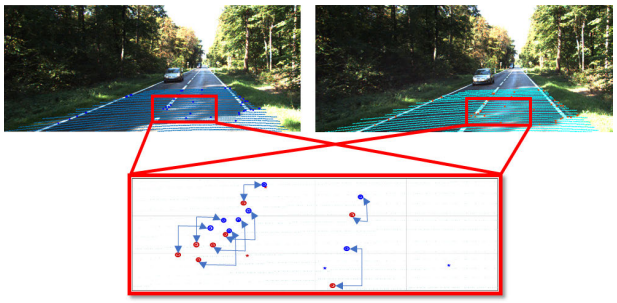

FIGURE 14. Road points tracked across two frames captured at different times.

feature image pixel points captured at times t and $t + \delta t$, two sets of corresponding 3D points are calculated. Figure 14 illustrates the resulting 3D-corresponding points and their corresponding 2D pixels from the same sections of the road images (frames).

B. COMPARISON WITH GRID-BASED INTERPOLATION METHODS

Although MATLAB provides grid-based interpolation methods, tests have shown that these methods often incur higher computational costs compared to `scatteredInterpolant`.

1. **Data Transformation**: Grid-based methods require the scattered data to be transformed into a regular grid. This transformation process can be computationally expensive, particularly for large datasets or when dealing with non-uniformly distributed data.
2. **Computational Complexity**: Grid-based methods involve more complex algorithms to fit the data into a structured grid, which increases computational time and resources.

VIII. FINDING 3D TRANSFORMATION MATRIX OF VEHICLE

This section describes the method for finding the homogeneous transformation matrix of two consecutive coordinate frames of the vehicle during its motion using the method explained in [42]. Consider two instantaneous coordinate frames for two time stamps with an interval of "δt" i.e., "t" and "$t + \delta t$". For any 3D corresponding point "r" (obtained in the previous section), assume the coordinate value concerning t^{th} frame is $r^t = \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T$ and $(t + \delta t)^{th}$ frame is $r^{t+\delta t} = \begin{bmatrix} x_{t+\delta t} & y_{t+\delta t} & z_{t+\delta t} \end{bmatrix}^T$. An augmented transformation matrix ($T_t^{t+\delta t}$) as in Eq.7 is considered to show the homogeneous transformation between t^{th} and $(t + \delta t)^{th}$ frames. The augmented transformation matrix ($T_t^{t+\delta t}$) is a 4×4 matrix (i.e. $T_t^{t+\delta t} \in IR^{4\times4}$), in which the top left 3×3 matrix (i.e. $R_t^{t+\delta t} = T_t^{t+\delta t}(j,k)$ where $j,k = 1,2,3$) is defines the rotation vehicle and top right 3×1 matrix (i.e. $\tau_t^{t+\delta t} = T_t^{t+\delta t}(j,k)$ where $j = 1,2,3$ and $k = 4$) defines the translation of the vehicle. The bottom row consists perspective factor ($\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$) i.e., orthonormal) and scaling factor (having value of 1, i.e., 1 : 1 transformation) respectively. Transformation matrix:

$$T_t^{t+\delta t} = \begin{bmatrix} R_t^{t+\delta t} & \tau_t^{t+\delta t} \\ 0 \quad 0 & 0 \quad 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & \tau_x \\ r_{yx} & r_{yy} & r_{yz} & \tau_y \\ r_{zx} & r_{zy} & r_{zz} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The section below explains, how the transformation matrix ($T_t^{t+\delta t}$) is calculated by applying the linear regression method and using the corresponding points.

Converting matrix multiplication form of frame transformation shown in Eq. 8 into a set of linear equations, three equation formed as shown in Eq. 9.

$$r^{t+\delta t} = T_t^{t+\delta t} r^t \tag{8}$$

$$\Rightarrow \begin{bmatrix} x_{t+\delta t} \\ y_{t+\delta t} \\ z_{t+\delta t} \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & \tau_x \\ r_{yx} & r_{yy} & r_{yz} & \tau_y \\ r_{zx} & r_{zy} & r_{zz} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix}$$

$$\Rightarrow x_{t+\delta t} = r_{xx}x_t + r_{xy}y_t + r_{xz}z_t + \tau_x$$

$$y_{t+\delta t} = r_{yx}x_t + r_{yy}y_t + r_{yz}z_t + \tau_y$$

$$z_{t+\delta t} = r_{zx}x_t + r_{zy}y_t + r_{zz}z_t + \tau_z \tag{9}$$

As a part of linear regression here, the square of error is considered as cost function and minimizes to get solution. For a linear form of equation, $y_l = mx_l + b$ (for, $l = 1$ to n) the cost function (J) is represented as Eq. 10

$$J = \epsilon^2 = \sum_{l=1}^{n} [y_l - (mx_l + b)]^2 \tag{10}$$

Writing cost function of above linear equations (Eq. 9) in the form of Eq. 10, three equation obtained as shown in Eq. 11. These equation basically square of location error in x, y and z directions.

$$\epsilon_x^2 = \sum_{l=1}^{n} [x_{(t+\delta t)l} - (r_{xx}x_{tl} + r_{xy}y_{tl} + r_{xz}z_{tl} + \tau_x)]^2 \tag{11a}$$

$$\epsilon_y^2 = \sum_{l=1}^{n} [y_{(t+\delta t)l} - (r_{yx}x_{tl} + r_{yy}y_{tl} + r_{yz}z_{tl} + \tau_y)]^2 \tag{11b}$$

$$\epsilon_z^2 = \sum_{l=1}^{n} [z_{(t+\delta t)l} - (r_{zx}x_{tl} + r_{zy}y_{tl} + r_{zz}z_{tl} + \tau_z)]^2 \tag{11c}$$

To find the minimum value of the square of the error ϵ_x^2, ϵ_y^2 and ϵ_z^2, its derivative must be zero. Thus, take the partial derivative of Eq. 11 and make the results equal to zero. For the first part of Eq. 11a, differentiating it concerning r_{xx}, r_{xy}, r_{xz} and t_x:

$$\frac{\partial \epsilon_x^2}{\partial r_{xx}} = -2 \sum_{l=1}^{n} [x_{(t+\delta t)l} - (r_{xx}x_{tl} + r_{xy}y_{tl} + r_{xz}z_{tl} + \tau_x)]x_{tl} = 0 \tag{12}$$

$$\frac{\partial \epsilon_x^2}{\partial r_{xy}} = -2 \sum_{l=1}^{n} [x_{(t+\delta t)l} - (r_{xx}x_{tl} + r_{xy}y_{tl} + r_{xz}z_{tl} + \tau_x)]y_{tl} = 0 \tag{13}$$

$$\frac{\partial \epsilon_x^2}{\partial r_{xz}} = -2\sum_{l=1}^{n}[x_{(t+\delta t)l} - (r_{xx}x_{tl} + r_{xy}y_{tl} + r_{xz}z_{tl} + \tau_x)]z_{tl} = 0 \tag{14}$$

$$\frac{\partial \epsilon_x^2}{\partial \tau_{xz}} = -2\sum_{l=1}^{n}[x_{(t+\delta t)l} - (r_{xx}x_{tl} + r_{xy}y_{tl} + r_{xz}z_{tl} + \tau_x)] = 0 \tag{15}$$

Writing the above equations in matrix form, they will become as given below:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \\ \tau_x \end{bmatrix} = [M]^{-1} \begin{bmatrix} \sum_{l=1}^{n} x_{(t+\delta t)l}x_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l}y_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l}z_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l} \end{bmatrix} \tag{16}$$

where, the matrix M is a symmetric matrix and take the form as given below:

$$M = \begin{bmatrix} \sum x_{tl}^2 & \sum x_{tl}y_{tl} & \sum x_{tl}z_{tl} & \sum x_{tl} \\ \sum x_{tl}y_{tl} & \sum y_{tl}^2 & \sum y_{tl}z_{tl} & \sum y_{tl} \\ \sum x_{tl}z_{tl} & \sum y_{tl}z_{tl} & \sum z_{tl}^2 & \sum z_{tl} \\ \sum x_{tl} & \sum y_{tl} & \sum z_{tl} & n \end{bmatrix} \in IR^{4\times 4} \tag{17}$$

Similarly, repeating the above process for the 2nd and 3rd parts of Eq. 11 (for 11b, 11c), they will become Eq. 18 and Eq. 19 respectively, where M is defined as in Eq. 17.

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \\ \tau_y \end{bmatrix} = [M]^{-1} \begin{bmatrix} \sum_{l=1}^{n} y_{(t+\delta t)l}x_{tl} \\ \sum_{l=1}^{n} y_{(t+\delta t)l}y_{tl} \\ \sum_{l=1}^{n} y_{(t+\delta t)l}z_{tl} \\ \sum_{l=1}^{n} y_{(t+\delta t)l} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \\ \tau_z \end{bmatrix} = [M]^{-1} \begin{bmatrix} \sum_{l=1}^{n} z_{(t+\delta t)l}x_{tl} \\ \sum_{l=1}^{n} z_{(t+\delta t)l}y_{tl} \\ \sum_{l=1}^{n} z_{(t+\delta t)l}z_{tl} \\ \sum_{l=1}^{n} z_{(t+\delta t)l} \end{bmatrix} \tag{19}$$

Combining eqs. 16, 18 and 19, the resulting equation will be the Eq. 20.

$$\begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \\ \tau_x & \tau_y & \tau_z \end{bmatrix} = [M]^{-1}[N] \tag{20}$$

where

$$N = \begin{bmatrix} \sum_{l=1}^{n} x_{(t+\delta t)l}x_{tl} & \sum_{l=1}^{n} y_{(t+\delta t)l}x_{tl} & \sum_{l=1}^{n} z_{(t+\delta t)l}x_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l}y_{tl} & \sum_{l=1}^{n} y_{(t+\delta t)l}y_{tl} & \sum_{l=1}^{n} z_{(t+\delta t)l}y_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l}z_{tl} & \sum_{l=1}^{n} y_{(t+\delta t)l}z_{tl} & \sum_{l=1}^{n} z_{(t+\delta t)l}z_{tl} \\ \sum_{l=1}^{n} x_{(t+\delta t)l} & \sum_{l=1}^{n} y_{(t+\delta t)l} & \sum_{l=1}^{n} z_{(t+\delta t)l} \end{bmatrix} \in IR^{4\times 3} \tag{21}$$

Now the final transformation matrix ($T_t^{t+\delta t}$) of the vehicle, for the frames t to $t+\delta t$ can be computed using the Eq. 22.

$$T_t^{t+\delta t} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & \tau_x \\ r_{yx} & r_{yy} & r_{yz} & \tau_y \\ r_{zx} & r_{zy} & r_{zz} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \left([M]^{-1}[N]\right)^T \\ 0\ 0\ 0 \quad 1 \end{bmatrix} \tag{22}$$

IX. ERROR STATE KALMAN FILTER (ESKF)

The transformation matrix obtained in above section using LiDAR and Camera data contains measurement noise. Hence, the error state Kalman filter is used to fuse the above results with IMU sensor data to improve the results. Inertial Measurement Unit (IMU - Model) primarily consists an accelerometer, a gyroscope, and a magnetometer sensors. By integrating linear acceleration (accelerometer reading) and angular velocities (pitch and roll rate from gyroscope and yaw rate from the magnetometer) over time, the vehicle position and orientation (Hamilton quaternion) are obtained. These obtained position and orientation data are prone to drifting error. Traditionally this drifting error is corrected using the absolute position obtained from GPS system. However, in the present work, as the LiDAR and the Camera-based primary positioning system is an incremental position measuring system (not providing absolute position data), the incremental error of the vehicle odometry is minimized by data fusing the LiDAR-Camera system and IMU data using the error state Kalman filter.

The error-state Kalman filter (ESKF) is an advanced version of Kalman filtering algorithm and according to [43]. The following are advantages of ESKF over the traditional Kalman Filter:

1) Here number of parameters are the same as degrees of freedom, so ESKF avoids redundancy of parameters and risk of singularity in the covariance matrix.
2) By neglecting second-order terms, ESKF makes Jacobian matrix calculations very easy.

A. VEHICLE SYSTEM KINEMATICS

The vehicle speed is tracked relative to the global reference frame. Here the global reference frame is assumed to be in a north-east-down direction according to the classical approach, and the position is the same as that of the initial vehicle frame. IMU reading (linear accelerations (a_m) and angular velocities(ω_m)) is defined w.r.t inertial reference frame; however, the vehicle state is defined w.r.t global reference frame. The vehicle state consists of three parameters: Position($p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \in IR^{3\times 1}$), Velocity ($v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \in IR^{3\times 1}$) and Quaternion ($q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \in IR^{4\times 1}$). The vehicle control parameters include: acceleration ($a_m \in IR^{3\times 1}$) and angular velocity ($\omega_m \in IR^{3\times 1}$). Vehicle state

$$x = \begin{bmatrix} p \\ v \\ q \end{bmatrix} \in IR^{10\times 1} \tag{23}$$

Vehicle control parameter

$$u = \begin{bmatrix} a_m \\ \omega_m \end{bmatrix} \in IR^{6\times 1} \tag{24}$$

Vehicle kinematic equation of motion in discrete-time format for a varying time interval $\delta t > 0$ considering classical mechanic's approach can be given as below:

$$x_{t+\delta t} = \begin{bmatrix} p_{t+\delta t} \\ v_{t+\delta t} \\ q_{t+\delta t} \end{bmatrix} = \begin{bmatrix} p_t + v_t \delta t + \frac{1}{2}\left(R_t\{q_t\}a_m + g\right)\delta t^2 \\ v_t + \left(R_t\{q_t\}a_m + g\right)\delta t \\ q_t \circ q\{\omega_m \delta t\} \end{bmatrix} \tag{25}$$

In above equation, $x_t = \begin{bmatrix} p_t & v_t & q_t \end{bmatrix}^T$ and $x_{t+\delta t} = \begin{bmatrix} p_{t+\delta t} & v_{t+\delta t} & q_{t+\delta t} \end{bmatrix}^T$ are vehicle states at two consecutive timestamps i.e. within the time interval of δt. This discrete-time state-space model is in nonlinear format and can be represented as:

$$x_{t+\delta t} = f(x_t, u, \delta t) \tag{26}$$

In Eq. 26, the term $q_t \circ q$ signifies the multiplication of quaternions, where q_t and q represent quaternion values. Additionally, the expressions $q\{\}$ and $R\{\}$ are detailed in the corresponding appendix for reference.

B. ERROR-STATE KINEMATICS

The error-state analysis is less complex than the normal state analysis. The vehicle error state can be defined as $\delta x_{t+\delta t} = \begin{bmatrix} \delta p_{t+\delta t} & \delta v_{t+\delta t} & \delta\theta_{t+\delta t} \end{bmatrix}^T = x_{t+\delta t} \ominus x_t \in IR^{9\times 1}$ and by expanding this error state formation using Taylor series expansion (only using the first derivative), the final equation can be expressed as:

$$\begin{aligned} \delta x_{t+\delta t} &= \begin{bmatrix} \delta p_{t+\delta t} \\ \delta v_{t+\delta t} \\ \delta\theta_{t+\delta t} \end{bmatrix} \\ &= \begin{bmatrix} \delta p_t + \delta v_t \delta t \\ \delta v_t + (-R_t\{q_r\}[a_m]_X \delta\theta_t + \delta g)\delta t + v_i \\ R\{\omega_m \delta t\}^T \delta\theta_t + \theta_i \end{bmatrix} \end{aligned} \tag{27}$$

In the above equation, the terms v_i and θ_i are random impulse inputs of velocity and orientation modelled as white Gaussian Processes [42]. $[a_m]_X$ represents the skew operation explained in appendix. The above error state equation can be further expressed in a non-linear function format as given below:

$$\delta x_{t+\delta t} = \mathcal{F}(x_t, \delta x_t, u, i) = F_x(x_t, u)\,\delta x_t + L_i i \tag{28}$$

In above equation F_x and L_i are the Jacobians of function $\mathcal{F}(x_t, \delta x_t, u, i)$ w.r.t vector $\delta x_t = \begin{bmatrix} \delta p_t & \delta v_t & \delta\theta_t \end{bmatrix}^T$ and $i = \begin{bmatrix} v_i & \theta_i \end{bmatrix}^T$ with x_t and u as constant.

$$F_x = \left.\frac{\partial \mathcal{F}}{\partial \delta x}\right|_{x_t,u} = \begin{bmatrix} I_{3\times3} & I_{3\times3}\delta t & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & -R_t\{q_t\}[a_m]_X \delta t \\ 0_{3\times3} & 0_{3\times3} & R_t\{\omega_m \delta t\}^T \end{bmatrix} \in IR^{9\times9} \tag{29}$$

$$L_i = \left.\frac{\partial \mathcal{F}}{\partial i}\right|_{x_t,u} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \in IR^{9\times6} \tag{30}$$

The ESKF prediction equation for error state and covariance matrix will be:

$$\delta\check{x}_{t+\delta t} = F_x(x_t, u_m)\delta\check{x}_t \tag{31}$$

$$\check{P} = F_x P F_x^T + L_i Q_i L_i^T \in IR^{9\times9} \tag{32}$$

C. FUSING IMU DATA WITH VISUAL-LIDAR DATA

After prediction of the error state ($\delta\check{x}_{t+\delta t}$)) and state covariance matrix($\check{P}$), both needs to be corrected using the available measurement data. In the present case, the transformation matrix ($T_t^{t+\delta t}$) obtained from Camera and LiDAR data fusion will be used for correcting the predicted error state and covariance matrix of the ESKF. The transformation matrix ($T_t^{t+\delta t}$) consists two parts: a translation ($\tau_t^{t+\delta t}$) and change in angle in the form of Rotation matrix ($R_t^{t+\delta t}$) as shown in Eq. 33

$$T_t^{t+\delta t} = \begin{bmatrix} R_t^{t+\delta t} & \tau_t^{t+\delta t} \\ 0_{1\times3} & 1 \end{bmatrix} \in IR^{4\times4} \tag{33}$$

The measurement update, Eq.33 (data obtained from LiDAR and Camera sensors) should be converted into the form $y_{t+\delta t} = h(x_t) + v$, where $y_{t+\delta t} = \begin{bmatrix} \tau_t^{t+\delta t} & q\{R_t^{t+\delta t}\} \end{bmatrix} \in IR^{7\times1}$ and v is white Gaussian noise with covariance V ($v \approx N\{0, R\}$). Here is the new transformation matrix $T_{t+\delta t}$ of the vehicle will be calculated using the old vehicle transformation matrix T_t and current transformation matrix ($T_t^{t+\delta t}$).

$$T_{t+\delta t} = T_t \oplus T_t^{t+\delta t} = T_t \times T_t^{t+\delta t} \tag{34}$$

$$\Rightarrow \begin{bmatrix} R_{t+\delta t} & p_{t+\delta t} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} R_t & p_t \\ 0_{1\times3} & 1 \end{bmatrix} + \begin{bmatrix} R_t^{t+\delta t} & \tau_t^{t+\delta t} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{35}$$

$$\tau_t^{t+\delta t} = R_t^T p_{t+\delta t} - R_t^T p_t \tag{36}$$

Similarly the rotational matrix is calculated from Eq. 35 and is given below:

$$R_{t+\delta t} = R_t \times R_t^{t+\delta t} \tag{37}$$

As $y_{t+\delta t}$ is consist of Quaternion instead of Rotation matrix, the rotation matrix is converted into Quaternion as given below [44]:

$$q_{t+\delta t} = q_t \circ q\{R_t^{t+\delta t}\} = Q_t \times q\{R_t^{t+\delta t}\} \tag{38}$$

$$\Rightarrow Q_t^{-1} q_{t+\delta t} = q\{R_t^{t+\delta t}\} \tag{39}$$

The appendix contains a detailed explanation of $Q_t \in IR^{4\times4}$ which represents the tensor form of the quaternion q_t. Writing Eq.36 and Eq.39 in matrix form to represent as $y_{t+\delta t} = h(x_{t+\delta t}, x_t)$

$$y_{t+\delta t} = \begin{bmatrix} \tau_t \\ q\{R_t^{t+\delta t}\} \end{bmatrix} = \begin{bmatrix} R_t^T & 0_{3\times3} & Q_t^{-1} \end{bmatrix} \begin{bmatrix} p_{t+\delta t} \\ v_{t+\delta t} \\ q_{t+\delta t} \end{bmatrix} + \begin{bmatrix} -R_t^T p_t \\ 0_{3\times3} \end{bmatrix} \tag{40}$$

Using the measurement equation (Eq.40), following filter correction Eqs. (41,42,43) are obtained:

$$K = \check{P}H^T \left(H\check{P}H^T + \Re\right)^{-1} \in IR^{9\times7} \tag{41}$$

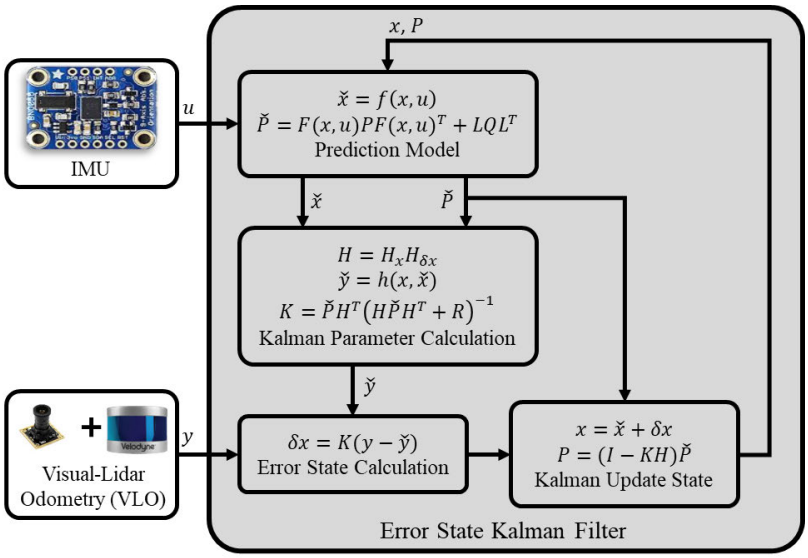

FIGURE 15. Error state kalman filter based sensor fusion process for VLO and IMU data.

$$\delta x_{t+\delta t} = K\left(y - y\left(\check{x}_t\right)\right) \tag{42}$$

$$P = (I - KH)\,\check{P} \tag{43}$$

However, the Eqs. (41, 43) requires Jacobean matrix H, which is defined as differentiation of h w.r.t error state δx and is calculated using chain rule defined in [45]

$$H \triangleq \left.\frac{\partial h}{\partial \delta x}\right|_x = \left.\frac{\partial h}{\partial x_{t+\delta t}}\right|_x \left.\frac{\partial x_{t+\delta t}}{\partial \delta x}\right|_x = H_x X_{\delta x} \in IR^{7\times 9} \tag{44}$$

$$H_x = \left.\frac{\partial h}{\partial x_{t+\delta t}}\right|_x = \left[R_t^T \;\; 0_{3\times 3} \;\; Q_t^{-1}\right] \in IR^{7\times 10} \tag{45}$$

$$X_{\delta x} = \left.\frac{\partial x_{t+\delta t}}{\partial \delta x}\right|_x = \begin{bmatrix} I_{6\times 6} & 0_{6\times 3} \\ 0_{4\times 6} & Q_{\delta\theta} \end{bmatrix} \in IR^{10\times 9} \tag{46}$$

In the above equations, the Quaternion term $Q_{\delta\theta}$ is defined as

$$Q_{\delta\theta} = \frac{1}{2} Q_t \begin{bmatrix} 0_{1\times 3} \\ I_{3\times 3} \end{bmatrix} \in IR^{4\times 3} \tag{47}$$

Using the filter correction error-state $\delta x_{t+\delta t}$ the actual vehicle state (position, velocity and quaternion) is updated using Eq.48:

$$x_{t+\delta t} = x_t \oplus \delta x_{t+\delta t} \tag{48}$$

The Error State Kalman Filter (ESKF) explained above can be represented in the flowchart form and shown in Figure 15.

X. EXPERIMENTAL SETUP

The developed algorithm has been verified using sets of data. First, using KITTI Dataset [46] and subsequently using the experimental data obtained from our laboratory setup in IIT Madras (referred as indoor case). The KITTI dataset [46] consists of different vehicle mounted sensor readings obtained by driving a vehicle (for autonomous driving application - in Karlsruhe University, Germany) for around 6 hours. In the case KITTI data, the vehicle mounted sensors are two grey scale Cameras, 2 Colour Cameras, 4 Edmund Optics lenses, one Velodyne HDL-64E (64 Laser head), and an OXTS system (Inertial and GPS navigation system). The data set also provides the Camera-to-Camera calibration and Camera LiDAR calibration for Intrinsic and extrinsic parameters.

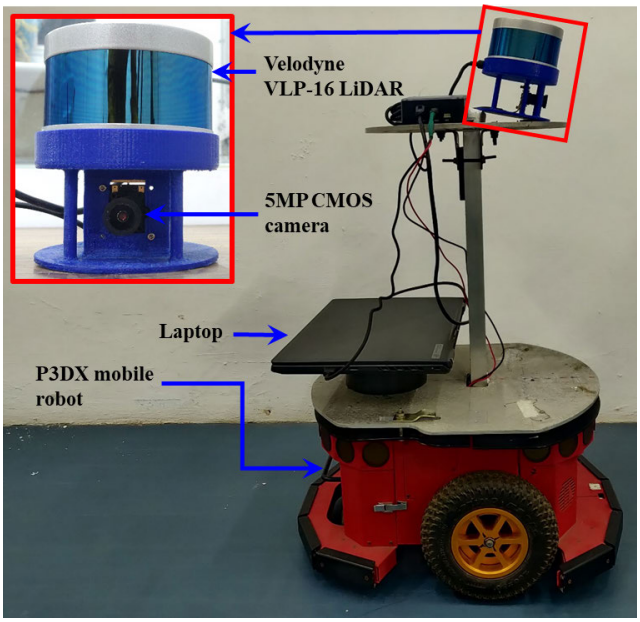

FIGURE 16. Mobile robot (P3DX) equipped with LiDAR and camera sensors.

The Figure 16 shows the experimental setup used for testing and verifying the developed algorithm for the indoor case (i.e. without GPS data). The experimental system consists P3DX Mobile robot, which is mounted with a

16-channel Velodyne VLP-16 LiDAR and a 5-megapixel Camera. The LiDAR and Camera combined structure is placed with a pitching angle of 17^o with respect to vehicle co-ordinate frame, to cover more ground surface. The P3DX mobile robot is controlled via programmes written using the ARIA and MATLAB environment. The mobile robot's encoder reading in the form of instantaneous location is saved w.r.t to a global coordinate frame for obtaining the vehicle ground truth. The vehicle's starting coordinate frame is considered a global coordinate frame. The vehicle is programmed to follow different standard paths: Straight, Circular, 'S' shape trajectory, and lemniscate trajectory) and data from various sensors are recorded for all further analysis. Additionally, it is also possible to mount the GPS sensor to obtain ground truth by processing GPS data, when the mobile robot is operated in the outdoor environment.

XI. RESULTS

In order to validate the developed algorithm and find its effectiveness in tracking the given path, KITTI data and experimental data are given as the inputs and the vehicle state (position, velocity and orientation) is obtained over the time. Using these results and the ground truth, the percentage translation error and rotational error are calculated. The following section presents the obtained results in the graphical and tabular formats for easy understanding.

The developed algorithm is able to find the vehicle path and represents it as the transformation matrix for each timestamp. This transformation matrix consists of rotation angle and translation for each time stamp. Here percentage translation error and rotational error per meter are calculated using Eq.49 and 50, by comparing with the ground truth (considered as actual path vehicle travelled). Considering

- $\tau \in IR^{3\times1}$ = Calculated translation
- $\tau_a \in IR^{3\times1}$ = Actual translation
- $\Theta \in IR^{3\times1}$ = Calculated Rotation angle in degree
- $\Theta_a \in IR^{3\times1}$ = Actual Rotation angle in degree

Percentage translational error:

$$\frac{||\tau - \tau_a||}{||\tau_a||} \times 100(\%) \tag{49}$$

Rotational Error:

$$\frac{||\Theta - \Theta_a||}{||\tau_a||}(deg/m) \tag{50}$$

A. ESTIMATION OF TRACKING ERROR WITH KITTI DATASET

By inputting different samples of the KITTI data-set (obtained when the vehicle was following the given path), the vehicle state is estimated. Subsequently, the mean percentage translational error and rotational error are computed and presented through the Figures 17 to 20. As mentioned earlier, these errors are the difference between tracked path and the ground truth. Further, the obtained overall percentage translational error and overall rotational error (deg/m) values are presented along with the same type errors reported for some of the existing methods for comparison purpose in the Table.2

In the below Table.2, EB3DTE (Example-based 3D Trajectory Extraction) and MonoDepth2 represents errors of monocular stereo odometry system, and it may be noted that the monochrome systems produce more errors (both the translation and rotation errors). This is because, generally it is difficult to get depth information using single Camera. Even with self-supervised learning methods, the monocular vision based system errors can't be reduced. VOFS (Stereo Visual Odometry with Flow Separation) and CFORB (Circular FREAK-ORB Visual Odometry) are the stereo vision odometry systems and they provide better results with lesser errors than the monocular Camera system as indicated in the table. However, the depth information provided by these systems are affected by their Cameras resolution. BLF (BUT-LOAM-FULL, LOAM-lidar odometry and mapping), BCC (BUT-CNN-CLASS), and D3DLO (Deep 3D LiDAR Odometry) are LiDAR based tracking systems and the tracking error of these systems mainly depend on the identification geometrical structure in the environment and structure matching. Any structure mismatching leads to tracking errors in the form of translational and rotational errors. Algorithms such as Traj-LIO (Trajectory - LiDAR-inertial Odometry) and mVLINS (Multilevel Visual-LiDAR-Inertial Navigation System) can deliver significantly better results; however, they rely on multiple LiDAR and IMU sensors for their operation. These methods are also computationally intensive, as can be seen from Table 2. It's worth mentioning that LiDAR odometry techniques such as SLAMesh (Simultaneous Localization and Meshing) and PUMA (Poisson Surface Reconstruction for LiDAR Odometry and Mapping), which are compared with our proposed method in Table 2, demonstrate superior performance in terms of translational and rotational accuracy. These methods create an offline mesh map of the environment from LiDAR data and subsequently utilize it for LiDAR odometry. However, it's important to note that these methods are considerably more computationally intensive compared to our proposed approach, as evident from Table 2. Additionally, there are other methods within these categories, such as Voxblox+ALOAM (Advanced implementation of LOAM) [56] and ULF-ESGVI (Unsupervised Lidar Feature Learning with ESGVI) [57].

Hence, by using LiDAR, Camera and IMU sensors data along with the Error State Kalman Filter (ESKF-VLO) to estimate the vehicle position, vehicle tracking errors are reduced (Translational Error = 3.142% and Rotational Error = 0.007 deg/m - as indicated in Table.2) compared with the other systems that use anyone of the sensors alone. However, even with this new vehicle tracking method, the tracking error can't be completely eliminated and this is due to the lack of identifiable features available in the road section even after image correction using histogram equalization. Any further reduction of tracking error can be achieved by

TABLE 2. Performance comparison with existing methods (KITTI dataset).

Method	Translational Error (%)	Rotational Error (deg/m)	Computational Time(in sec)	Platform
ESKF-VLO	3.142	0.007	0.17	1 cores @ 2.3 Ghz (MATLAB)
BLF [32]	3.49	0.012	0.7	1 core @ 2.5 Ghz (C/C++)
BCC [32]	4.59	0.017	1	1 core @ 2.5 Ghz (C/C++)
D3DLO [47]	5.4	0.015	0.1	GPU @ 2.5 Ghz (Python)
VOFS [48]	3.94	0.009	0.51	1 core @ 2.0 Ghz (C/C++)
CFORB [49]	3.73	0.01	0.9	8 cores @ 3.0 Ghz (C/C++)
EB3DTE [50]	5.45	0.027	1	1 core @ 2.5 Ghz (Matlab)
MonoDepth2 [51]	12.59	0.031	1	1 core @ 2.5 Ghz (C/C++)
Traj-LIO [52]	0.57	0.0015	0.1	4 cores @ 2.5 Ghz (C/C++)
mVLINS [53]	2.219	0.13	-	8 Cores up to 5.1 GHz (maximum)
SLAMesh [54]	1.25	0.0296	0.7	8 Cores @ 3.6GHz
PUMA [55]	3.38	0.074	1.2	8 Cores @ 3.6GHz

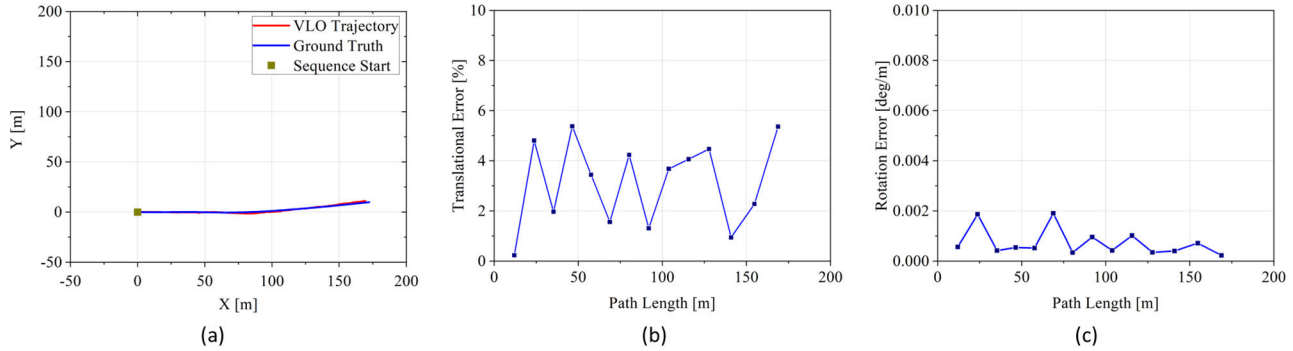

FIGURE 17. KITTI data-set Sample-1 (a) Trajectory track (b) Translational Error (c) Rotational Error.

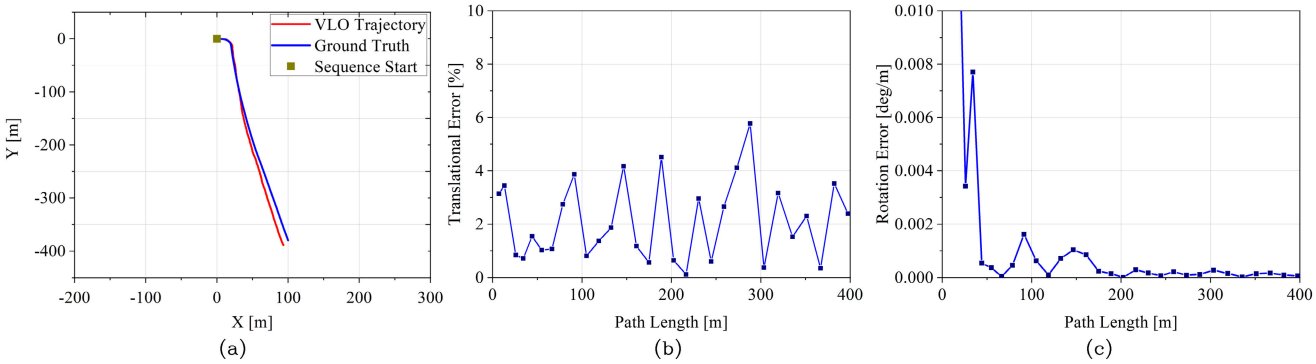

FIGURE 18. KITTI data-set Sample-2 (a) Trajectory track (b) Translational Error (c) Rotational Error.

doing accurate calibration of Camera and LiDAR sensors, since any small error in calibration parameters can create a significant error in motion estimation.

Due to the limitation of computational power, the code was executed on a laptop with a single-core 2.30 GHz processor. Despite this limitation, the laptop provided a fairly good computational speed. The Table 3 lists the computational cost of each major module of the process in MATLAB. The overall mean computational cost is 0.17 seconds, which is quite good compared to other methods that utilize higher processing power. It is worth noting that the processing speed could be improved by using a multi-core system or a GPU.

TABLE 3. Time taken by each step in a single process.

Process	Time (in sec)
Ground plane Segmentation	0.066
Feature matching	0.070
Interpolation method (for both two time stamps)	0.031
Error State Kalman Filter	0.003
Other	0.003
Overall Time	**0.17**

B. ESTIMATION OF TRACKING ERROR WITH EXPERIMENTAL DATASET

Similarly, by inputting the experimental data set, the mean percentage translational error and rotational error in deg/m

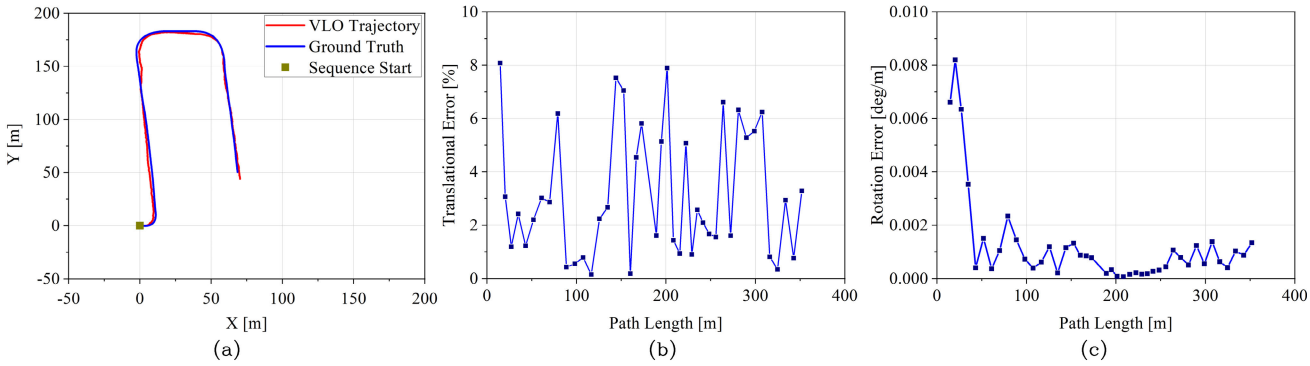

FIGURE 19. KITTI data-set Sample-3 (a) Trajectory track (b) Translational Error (c) Rotational Error.

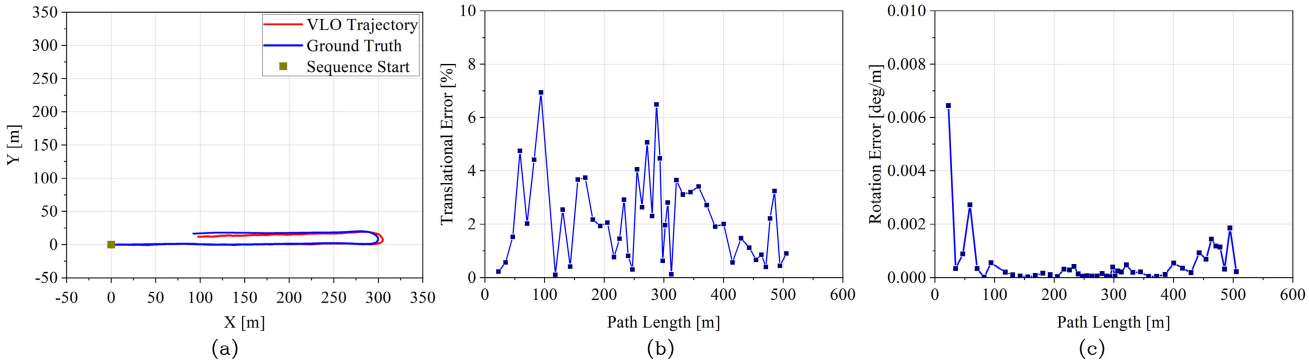

FIGURE 20. KITTI data-set Sample-4 (a) Trajectory track (b) Translational Error (c) Rotational Error.

TABLE 4. Error comparison for different datasets.

	No. of dataset Considered	Translational Error (%)	Rotational Error (deg/m)
Residential	15	3.03	0.0058
Road	8	2.9	0.0045
City	10	3.4	0.008
Campus	8	3.24	0.009

TABLE 5. Translational and rotational error comparison KITTI and experimental datasets (with ESKF-VLO method).

Method	Translational Error (%)	Rotational Error (deg/m)
ESKF-VLO (KITTI dataset)	3.142	0.007
ESKF-VLO (real-time)	4.695	0.052

along with tracked path compared with ground truth are plotted in the Figures 21 to 24.

From Table.5, it can be observed that the overall translational and rotational errors for the KITTI dataset are lesser than the results obtained with the data from real-time experiments conducted in the laboratory condition (indoor condition). This is mainly due to the use of High-resolution LiDAR (64-channel) in the KITTI dataset compared to 16-channel low-resolution LiDAR used for the real-time experiments. Moreover, outdoor road contains various components like lane markings, shadow, and other textures, so it is possible to get more useful features (edges, corners) than the indoor condition with less texture even after the image contrast enhancement. In the KITTI dataset, the vehicle speed (maximum speed obtained can be 150 KM/hr) is much higher than the mobile robot (P3DX) experiment performed in the laboratory (maximum speed of P3DX mobile robot is 4 KM/hr). Therefore the rotation per unit translation of the mobile robot is higher than that of the KITTI dataset vehicle. This causes a higher rotational error per meter in the laboratory vehicle than in the KITTI dataset.

XII. OBSERVATION

The algorithm exhibits a general resilience to various environmental conditions, yet its efficacy hinges significantly on the process of feature matching. Within different contexts, the algorithm encounters distinct challenges:

In Indoor Environments with Sparse Features:

- **Surfaces Uniformity:** Within indoor spaces characterized by vast expanses of uniform surfaces like monochromatic floors or plain walls, the algorithm grapples with identifying sufficient distinctive features essential for reliable matching. This dearth of features invariably undermines the algorithm's accuracy.

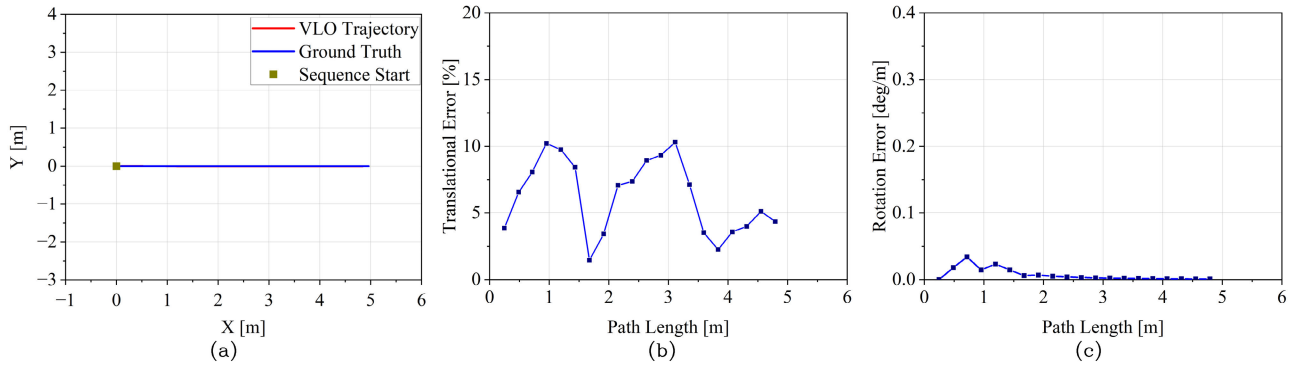

FIGURE 21. Straight line Trajectory (a) Trajectory track (b) Translational Error (c) Rotational Error.

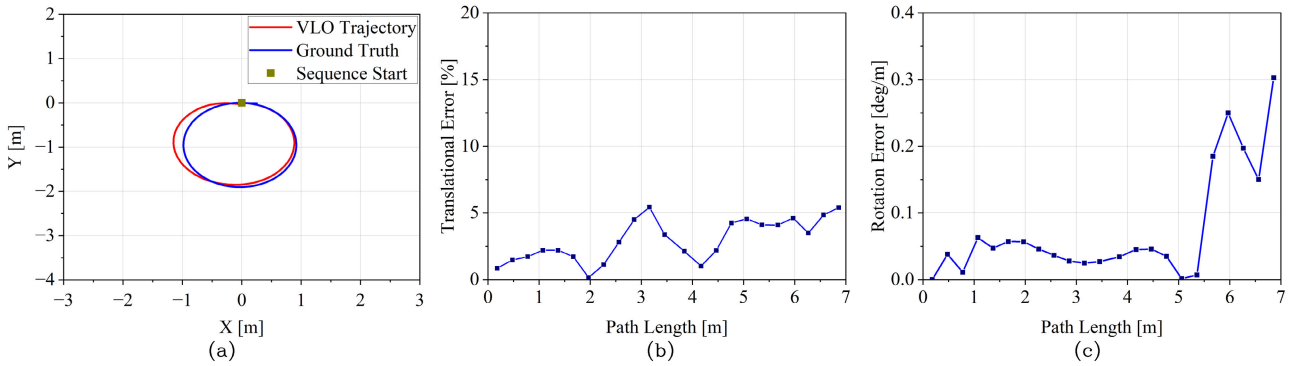

FIGURE 22. Circular Trajectory (a) Trajectory track (b) Translational Error (c) Rotational Error.

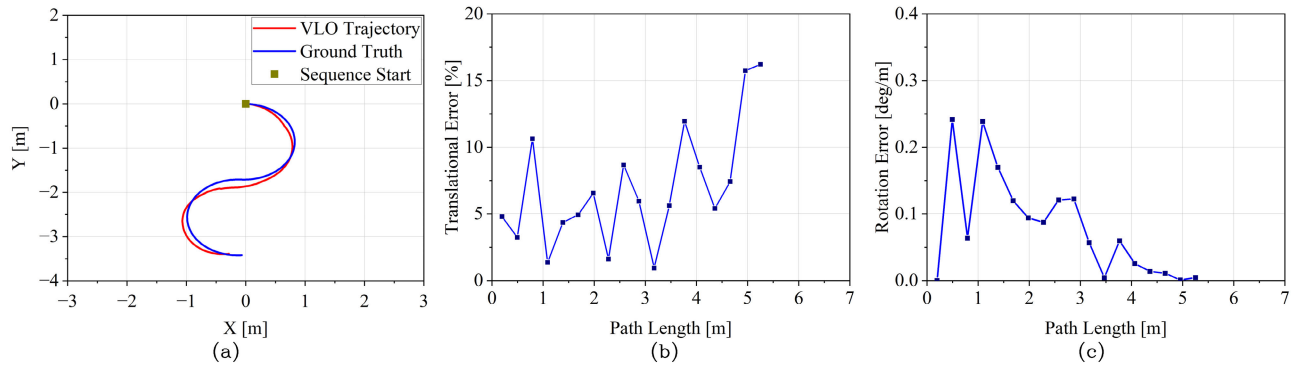

FIGURE 23. S-shape Trajectory (a) Trajectory track (b) Translational Error (c) Rotational Error.

- **Lighting Dynamics:**
  - **Dim Illumination:** Operating in dimly lit settings presents a formidable obstacle for the algorithm to discern features accurately. Dark surfaces pose a particular challenge as their limited light reflection complicates feature capture.
  - **Shadows:** Paradoxically, shadows sometimes offer assistance by engendering contrasts useful for feature matching.
  - **Fluctuating Lighting:** Swift alterations in lighting conditions, encompassing phenomena like moving shadows or fluctuating light sources, introduce inconsistencies in feature detection. These discrepancies engender errors in matching, thereby compromising the accuracy of state estimation.

Within Outdoor Environments with High Congestion:

- **Surface Obstructions:** In scenarios replete with vehicular and pedestrian traffic, the algorithm contends with obstructed ground surfaces, impeding its ability to detect ground features. This obstruction culminates in a paucity of reliable features for matching, thereby undermining the algorithm's overall performance.

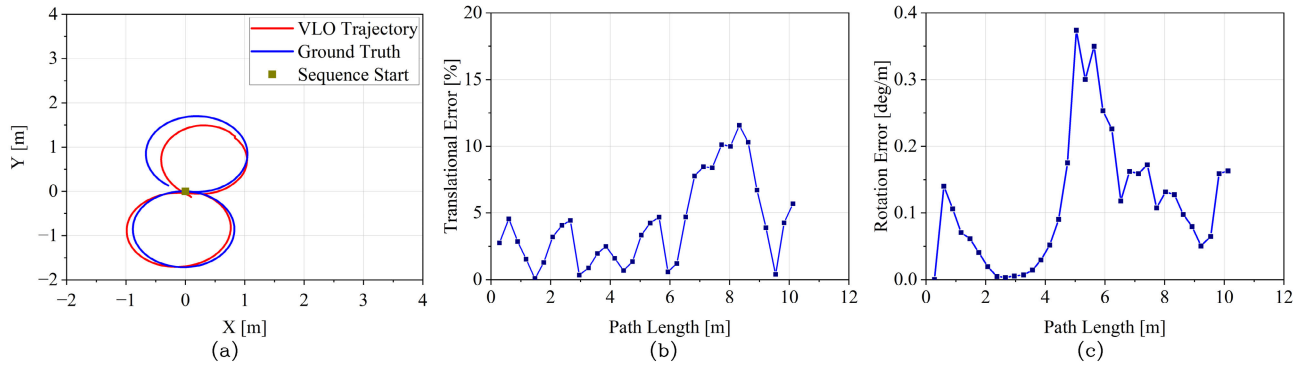

FIGURE 24. Lemniscate Trajectory (a) Trajectory track (b) Translational Error (c) Rotational Error.

Problem of Detection of Multiple Flat Surface:

The purpose of identifying flat surfaces is to enable precise interpolation between 2D image points and 3D LiDAR data. This is crucial for determining the 3D coordinates (relative to the LiDAR frame) of matched feature points between images taken at times t and $t + \delta t$. Interpolation works effectively on flat surfaces, which is why it is important. Feature matching algorithms typically detect a limited set of points on an image, and it is preferable for these points to be located on flat surfaces. The presence of numerous flat surfaces is advantageous because it provides a larger area for feature matching in images, resulting in more corresponding points for finding the 3D transformation matrix, as explained in Section VII. To manage computational complexity, the LiDAR data undergoes a filtering process based on specific height ranges, considering the Z values prominent for the ground plane. This filtering primarily focuses on identifying the most prominent flat surface, which is usually the ground plane. The ground plane can also accommodate sloping roads as long as they remain flat.

XIII. CONCLUSION

In this work, a self-tracking method (ESKF-VLO) that uses Camera, LiDAR and IMU sensors data alone, without using data from any external infrastructure (i.e. GPS) has been developed for effective vehicle navigation. From the LiDAR point-cloud information, movable free space for the vehicle is segmented using the ground plane segmentation method. A LiDAR to Camera dataset is prepared for finding the relationship between the 3D location of the segmented ground plane and Camera-image pixel 2D coordinate frame, using the calculated LiDAR to Camera calibration parameters. Developed feature matching algorithm finds the correspondence points between the two consecutive Camera image frames (for segmented ground plane) and using interpolation in above dataset its (correspondence points), 3D locations of the Interested Features Points (IFPs) are obtained. This corresponding 3D location points are used to compute the vehicle transformation matrix of the two consecutive vehicle coordinate frames. Finally, Error State Kalman filter is used to reduce the tracking error by data fusing the IMU sensor data with the vehicle transformation matrix, to find the state of the vehicle.

The developed algorithm has been verified in terms of tracking error (translation error in % and rotational error in deg/m) with publicly available KITTI dataset. Further, to find the usefulness of developed algorithm in tracking vehicle at indoor environment, vehicle state and state errors were estimated using the data obtained from real time experiments conducted with a P3DX Mobile robot that is instrumented with 16-channel Velodyne LiDAR – VLP-16, Monochrome Camera and IMU sensor. The results show algorithm is working and is able to track the vehicle with minimal error. For the KITTI dataset, the overall translational error is 3.142 %, and the rotational error is 0.007 deg/m, and for the indoor real-time experiments, the overall translational error is 4.695 %, and the Rotational error is 0.052 deg/m.

APPENDIX

1) Quaternion Product:

$$q \circ p = (q_0 + q_1 i + q_2 j + q_3 k)\,(p_0 + p_1 i + p_2 j + p_3 k) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = Q \times P$$

2) Tensor form of quaternion q is:

$$Q = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \in I\!R^{4\times 4}$$

3) Skew Operator:

$$[a]_X \triangleq \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

4) $q\{v\}$ = Quaternion associated with the rotation v
5) $R\{q\}$ = Rotation matrix associated with the current nominal orientation q

REFERENCES

[1] S. R. Sahoo and P. V. Manivannan, *A Hybrid Approach for Dynamic Observer to Detect and Track Dynamic Obstacles*, vol. 10, no. 2, 2020.

[2] G. Dedes and A. G. Dempster, "Indoor GPS positioning–challenges and opportunities," in *Proc. IEEE 62nd Veh. Technol. Conf. (VTC–Fall)*, vol. 1, Sep. 2005, pp. 412–415.

[3] S. Lee and J.-B. Song, "Robust mobile robot localization using optical flow sensors and encoders," in *Proc. IEEE Int. Conf. Robot. Autom.*, Sep. 2004, pp. 1039–1044.

[4] J. Kunhoth, A. Karkar, S. Al-Maadeed, and A. Al-Ali, "Indoor positioning and wayfinding systems: A survey," *Hum.-Centric Comput. Inf. Sci.*, vol. 10, no. 1, Dec. 2020.

[5] M. A. Nassar, M. Hasan, M. Khan, M. Sultana, M. Hasan, L. Luxford, P. Cole, G. Oatley, and P. Koutsakis, "WiFi-based localisation datasets for no-GPS open areas using smart bins," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107422.

[6] J. Wang, R. K. Ghosh, and S. K. Das, "A survey on sensor localization," *J. Control Theory Appl.*, vol. 8, no. 1, pp. 2–11, Feb. 2010.

[7] K. J. Chandan and A. M. Akhil, "Investigation on accuracy of ultrasonic and LiDAR for complex structure area measurement," in *Proc. 6th Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2022, pp. 134–139.

[8] K. Yokoyama and K. Morioka, "Autonomous mobile robot with simple navigation system based on deep reinforcement learning and a monocular camera," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2020, pp. 525–530.

[9] M. Mihálik, B. Malobický, P. Peniak, and P. Vestenický, "The new method of active SLAM for mapping using LiDAR," *Electronics*, vol. 11, no. 7, p. 1082, Mar. 2022.

[10] P Srinivas, Y. L. Malathilatha, and M. V. N. K. Prasad, "Image processing edge detection technique used for traffic control problem," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 1, pp. 17–20, 2013.

[11] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[13] Z. Chen, J. Zhang, and D. Tao, "Progressive LiDAR adaptation for road detection," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 693–702, May 2019.

[14] F. Xu, L. Chen, J. Lou, and M. Ren, "A real-time road detection method based on reorganized LiDAR data," *PLoS ONE*, vol. 14, no. 4, Apr. 2019, Art. no. e0215159.

[15] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3923–3931.

[16] H. Alismail, B. Browning, and S. Lucey, "Photometric bundle adjustment for vision-based SLAM," in *Proc. Asian Conf. Comput. Vis.*, vol. 10114, 2017, pp. 324–341.

[17] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[18] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[19] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Proc. Joint Pattern Recognit. Symp.*, vol. 6376, 2010, pp. 11–20.

[20] H. Jin, P. Favaro, and S. Soatto, "Real-time 3D motion and structure of point features: A front-end system for vision-based control and interaction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Feb. 2000, pp. 778–779.

[21] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[23] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[24] I. Cherabier, J. L. Schönberger, M. R. Oswald, M. Pollefeys, and A. Geiger, "Learning priors for semantic 3D reconstruction," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 11216, 2018, pp. 325–341.

[25] N. Savinov, C. Häne, L. Ladický, and M. Pollefeys, "Semantic 3D reconstruction with continuous regularization and ray potentials using a visibility consistency constraint," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5460–5469.

[26] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from RGB-D video," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 599–609, Dec. 2015.

[27] C. Toft, C. Olsson, and F. Kahl, "Long-term 3D localization and pose from semantic labellings," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 650–659.

[28] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: Types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, no. 1, Dec. 2016.

[29] A. Nüchte, K. Lingemann, and J. Hertzberg, "6D SLAM3D mapping outdoor environments," *J. Field Robot.*, vol. 33, no. 1, pp. 1–17, 2014.

[30] H. Zhan, C. S. Weerasekera, J.-W. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4203–4210.

[31] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," *Auto. Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[32] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2018, pp. 71–77.

[33] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: State estimation, river detection, and obstacle mapping," *Auto. Robots*, vol. 33, nos. 1–2, pp. 189–214, Aug. 2012.

[34] M. Duarte-Silva, J. Henriques-Calado, and V. Camotim, "FastSLAM: A factored solution to the simultaneous localization and mapping problem Michael," *Women Therapy*, vol. 35, nos. 3–4, pp. 221–232, 2012.

[35] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[36] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, Apr. 2020.

[37] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 5562–5569.

[38] V. K. Nagaraja, "Feature selection using eigenvalue optimization and partial least squares," *Cgis.Cs.Umd.Educ.*, 2013.

[39] S. S. Bagade, "Use of histogram equalization in image processing for image enhancement," *Int. J. Softw. Eng. Res. Practices*, vol. 1, no. 2, pp. 6–10, 2011.

[40] M. Muja and D. G. Lowe, "Fast matching of binary features," in *Proc. 9th Conf. Comput. Robot Vis.*, May 2012, pp. 404–410.

[41] I. Amidror, "Scattered data interpolation methods for electronic imaging systems: A survey," *J. Electron. Imag.*, vol. 11, no. 2, p. 157, Apr. 2002.

[42] J. Cashbaugh and C. Kitts, "Automatic calculation of a transformation matrix between two frames," *IEEE Access*, vol. 6, pp. 9614–9622, 2018.

[43] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal, "Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation," in *Proc. AIAA Guid., Navigat., Control Conf.*, Aug. 2011, p. 6615.

[44] B. Graf, "Quaternions and dynamics," 2008, *arXiv:0811.2889*.

[45] J. Solà, "Quaternion kinematics for the error-state Kalman filter," 2017, *arXiv:1711.02508*.

[46] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2011.

[47] P. Adis, N. Horst, and M. Wien, *D3DLO: Deep 3D LiDAR Odometry Institute of Imaging and Computer Vision*. Aachen, Germany: RWTH Aachen Univ., 2021, pp. 3128–3132.

[48] M. Kaess, K. Ni, and F. Dellaert, "Flow separation for fast and robust stereo odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3539–3544.

[49] D. J. Mankowitz and E. Rivlin, "CFORB: Circular FREAK-ORB visual odometry," 2015, *arXiv:1506.05257*.

[50] Z. Boukhers, K. Shirahama, and M. Grzegorzek, "Example-based 3D trajectory extraction of objects from 2D videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2246–2260, Sep. 2018.

[51] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3827–3837.
[52] X. Zheng and J. Zhu, "Traj-LIO: A resilient multi-LiDAR multi-IMU state estimator through sparse Gaussian process," 2024, *arXiv:2402.09189*.
[53] B. Zhang, W. Yao, Y. Wang, P. Li, X. Shao, and G. Sun, "MVLINS: A multilevel visual-LiDAR-inertial navigation system with completely decoupled odometry and adaptive environmental mapping," *IEEE Trans. Intell. Vehicles*, pp. 1–13, 2024.
[54] J. Ruan, B. Li, Y. Wang, and Y. Sun, "SLAMesh: Real-time LiDAR simultaneous localization and meshing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 3546–3552.
[55] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for LiDAR odometry and mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5624–5630.
[56] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1366–1373.
[57] D. J. Yoon, H. Zhang, M. Gridseth, H. Thomas, and T. D. Barfoot, "Unsupervised learning of LiDAR features for use ina probabilistic trajectory estimator," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2130–2138, Apr. 2021.

SUDEEPTA R. SAHOO received the bachelor's degree in mechanical engineering from NMIET, Bhubaneswar, in 2014, and the master's degree in mechanical system design from IIT Bhubaneswar, in 2017. He is currently a Ph.D. Scholar with the Mechanical Engineering Department, IIT Madras, India. His master's project concentrated on motion planning for ground vehicles. His Ph.D. research involves the navigation and control of Swarm Aerial vehicles. His primary research interests include developing a cooperative navigation and control system for unmanned ground and aerial vehicles.

P. V. MANIVANNAN received the Ph.D. degree in control system for SI engines from IIT Madras, India, and the master's degree in applied electronics from the College of Engineering, Anna University, Chennai, India. He is currently an Associate Professor with the Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai, India. He has extended his expertise globally. He has contributed to academia as a Visiting Faculty Member of the University of South Australia, Adelaide, Australia; the University of Nebraska, Lincoln, USA; and the University of Kaiserslautern, Germany. Notably, he has taught the summer term course "Mechatronic Systems." He was a distinguished recipient of the DAAD Fellowship and ERASMUS MUNDUS Teaching Fellowship. His instructional and research interests include mechatronics, robotics, automotive control systems, embedded system design, and sensor networks.