Online LiDAR-to-Vehicle Alignment Using Lane Markings and Traffic Signs

Yao Hu, Xinyu Du, Shengbing Jiang
General Motors
GM Global Technical Center, Warren, MI 48092, USA
{yao.hu, xinyu.du, shengbing.jiang}@gm.com

Abstract

Highly automated vehicles with multiple environmental sensors require all the sensors aligned online to the same coordinate to ensure driving performance and improve customer convenience, especially when misalignment occurs during driving due to degradation, ageing, vibration, or accidents. The alignment between the LiDAR and the ego vehicle is one of several types of alignments. In this paper, an online alignment approach using road elements, e.g., lane markings and traffic signs, in aggregated LiDAR point cloud is developed. The optimization process to minimize the variance of aggregated point cloud for each road element is employed to automatically calculate the alignment parameters. To improve the algorithm robustness and accuracy, several excitation conditions occurred in daily driving are identified by algorithm sensitivity analysis with small input perturbations. The road elements are detected using unique designed heuristic algorithms from the distorted point cloud due to the inaccurate alignment parameters during optimization. The whole solution is validated by the data collected from several test vehicles, and the validation results demonstrate the effectiveness and robustness of the proposed solution.

1. Introduction

LiDAR (Light Detection And Ranging) is an important sensor for highly automated vehicles (HAVs) since it directly measures 3D coordinates of the surrounding objects. It is critical for the perception and localization in HAVs. The LiDAR emits a laser beam in a given direction defined by a vertical emission angle and a horizontal emission angle (azimuth), and measures the time interval from the emission to its return after being reflected by any object surface, thereby determines the distance of the object surface to the LiDAR. By scanning in different directions, a point cloud is generated by the LiDAR, which captures the 3D position information of the surfaces of the objects in the environment. In addition, the LiDAR also measures the intensity of the reflected laser beam, which indicates the object surface characteristics such as color, texture, material, etc.

The raw LiDAR points are recorded in the LiDAR coordinate system. In the HAV application, we need the positions of detected objects with respect to the vehicle coordinate system. Thus, the LiDAR-to-vehicle (extrinsic) alignment is needed to map the LiDAR points into the vehicle coordinate system. Moreover, in an application such as localization, the detected objects need to be reported in an earth-fixed coordinate system. The IMU (Inertial Measurement Unit, also known as INS - Inertial Navigation System) and GPS/GNSS (Global Positioning System / Global Navigation Satellite System) are used to provide the transform to further map the LiDAR points into the earth-fixed coordinate system.

The LiDAR-to-vehicle alignment can be represented by a 4x4 transformation matrix, or equivalently 6 parameters based on Euler angle definition, *i.e.,* translation Tx, Ty and Tz, and orientation Roll, Pitch and Yaw. There are three types of approaches to determine the alignment. The first type is to minimize the distance between the LiDAR point cloud of an object to the ground-truth coordinate of the object. The ground truth can be obtained using the GPS/GNSS equipment or the camera positioning system (*e.g.,* VICON system [1]). This type of approaches requires accurate ground truthing, so it's suitable for manufacturing alignment but not online alignment. The second type is to minimize the object vagueness in the aggregated LiDAR point cloud for an object [2]. This type of approaches requires that the vehicle maneuver contains enough excitation, and the desired objects can be identified. The third type, known as hand-eye calibration, is to minimize the difference of the transformation matrices between two positions estimated using the data from the vehicle inertial system and using the LiDAR frame to frame point cloud registration [3, 4, 5]. This type of approaches also needs enough vehicle maneuver excitation and may suffer from inaccurate point cloud registration due to degeneracy [6].

Our approach discussed in this paper is of the second type. We propose to use common road elements, *i.e.,* lane markings and/or traffic signs, for the LiDAR-to-vehicle alignment. With analytical study on sensitivity, we derive the conditions of vehicle maneuvers, which provide sufficient excitations to determine the 6 parameters of alignment respectively. We develop specific algorithms to

extract lane markings and traffic signs from the LiDAR point cloud as the targets for alignment. Then, an optimization is performed to determine the optimal alignment parameters to minimize the vagueness of the extracted targets. The whole algorithm is validated with the data collected by our test vehicle. The results show that the mean error of alignment is less than 0.2-degree error on orientation and 2cm error on translation. To the best of our knowledge, the proposed online LiDAR-to-vehicle alignment algorithms using traffic signs and lane markings and the associated sensitivity analysis are the first practical solution for the large scale of highly automated vehicles.

2. Related work

Hand-eye calibration: Hand-eye calibration is a popular method for extrinsic calibration for many different sensors [3, 4, 7]. For LiDAR-to-vehicle calibration, it determines the sensor pose transform A between two timestamps using point cloud registration methods, and vehicle pose transform B between the same timestamps. Then the calibration is to solve the equation $AX=XB$, where X is the vehicle-to-LiDAR transform. [8] formulates the hand-eye calibration into a least square problem using dual-quaternions, and provides the ways to determine the optimal solutions. To improve the robustness, [9] uses Schur matric decomposition in solving the hand-eye equation and detecting the outliers of sensor data when determining A and B. More studies incorporate extrinsic calibration into localization and mapping. [10] uses a graph optimization to formulate the autocalibration and localization to optimize A, X and B at the same time, which can reduce the impact of errors from B, but needs more computational resource. [11] estimates extrinsic parameters together with IMU bias and vehicle dynamics with Extended Kalman Filter (EKF), which uses Normal Distributions Transform (NDT) to register LiDAR scans as observations. [12] and [13] formulate an optimization problem to perform extrinsic and dynamic parameters together, which uses B-spline to constraint the dynamic trajectories and NDT to estimate the LiDAR trajectory. They also extract planes to build a surfels map for point-to-plane distance loss. These methods address the noise in B to some extent, but still suffer from A inaccuracy due to registration degeneracy and accumulated error.

Other LiDAR-to-vehicle calibration methods: To avoid the error accumulation issue in hand-eye methods, other methods are developed based on scanning the same stationary object and obtain the aggregated point cloud over time. The LiDAR-to-vehicle misalignment generates vagueness in the aggregated point cloud and the methods are to minimize the vagueness [14, 15, 16, 17, 2]. To measure the vagueness, [14] defines the distance between the planar surfaces. [17] uses the distance between correspondences of matched points. [16, 2] use PCA (principal component analysis) to determine the variance of the point cloud. [15, 16] require predefined locations. [14, 17] use ICP (iterative closest point) on the point clouds not associated with specific targets, which has the challenge to find the accurate correspondences when misalignment is relatively big. Our method follows the idea of PCA. We select lane markings and traffic signs as targets to minimize their variance. Our algorithm achieves 0.2-degree error, comparing to [2] which doesn't use specific targets and achieves around 2-degree error.

Lane marking detection: Methods to detect lane markings in a LiDAR point cloud are discussed in [18, 19, 20, 21, 22, 23]. Most of the methods use intensity of the LiDAR points to segment lane marking points and detect ground before detecting lane markings. [18, 20] also use the elevation information to determine lane marking points. [23] use clustering to better utilize the characteristics of the line shape. Many of these methods assume accurate alignment, which have the challenge with the inaccurate alignment. Most of these methods output a line model representing the position and direction of a lane marking, while our application requires capturing the shape or outline of the lane marking segment. Therefore, we develop a new method to better capture the lane marking boundaries in each single LiDAR frame.

Traffic sign detection: So far, most traffic sign (object) detection algorithms are based on deep neural networks, *e.g.* VoxelNet [24], PointNet [25], LaserNet [26]. However, such methods are challenging to be applied to the online alignment due to computation resource concerns.

3. Problem formulation

Denote a point in LiDAR coordinate system at time i as $P_L^i = \left[x_L^i, y_L^i, z_L^i\right]^T$. Convert this point into the vehicle coordinate system:

$$P_I^i = R_\beta * P_L^i + t \quad (1)$$

where $P_I^i = \left[x_I^i, y_I^i, z_I^i\right]^T$ is the point's coordinates in the vehicle coordinate system, R_β is the 3x3 rotation matrix for LiDAR-to-vehicle alignment, and t is the translation vector for LiDAR-to-vehicle alignment. The x, y and z axes of the vehicle coordinate system are in the directions of vehicle forward, left, and up, respectively. Convert this point into the earth-fixed coordinate system:

$$P^i = R^i * P_I^i + p^i \quad (2)$$

where P^i is the point's coordinates in the earth-fixed coordinate system, and R^i and p^i are the rotation matrix and translation vector from the vehicle to the earth-fixed coordinate system based on IMU and GPS, respectively. We define the earth-fixed coordinate system to be the vehicle coordinate system at the initial state of a drive. Once the vehicle moves after the initial state, the two coordinate systems become different, so are R^i and p^i. For

a stationary point in the earth-fixed coordinate system, its location should not change from time i to time j. The calibration is formulated as an optimization problem:

$$Calibrated\ (R_\beta, t) = \underset{(R_\beta,t)}{\operatorname{argmin}} \sum_{(i,j)} \| P^i - P^j \| \quad (3)$$

Consider a stationary target with a board shape, such as a traffic sign or a segment of a lane marking. Denote $\{P\}$ as the set of LiDAR points associated with this target in earth-fixed coordinate system collected at different time. $\{P\}$ represents an aggregated point cloud of a target in the earth-fixed coordinate system. The position change $\| P^i - P^j \|$ of each point in $\{P\}$ increases the 3D location variance of the point cloud. Thus, the optimization can be formulated as:

$$Calibrated\ (R_\beta, t) = \underset{(R_\beta,t)}{\operatorname{argmin}}\ var\{P\} \quad (4)$$

where $var\{P\}$ represents the 3D location variance of the aggregated point cloud of this target.

Notice that (R_β, t) is derived from 6 parameters (Tx, Ty, Tz, Roll, Pitch and Yaw) representing LiDAR-to-vehicle transformation.

4. Sensitivity analysis

To understand how the vehicle routes and the targets impact the results of the alignment calibration, the sensitivity analysis is performed as follows: (1) introduce small perturbations for the 6 parameters; (2) calculate the change on the object location in the earth-fixed system; (3) identify the appropriate routes/targets for the calibration of different parameters.

Without loss of generality, in the following we assume the roll (α_x^i) and pitch (α_y^i) angles of the vehicle with respect to the earth-fixed coordinate system are very small and can be neglected, *i.e.*, $(\alpha_x^i \cong \alpha_y^i \cong \alpha_x^j \cong \alpha_y^j \cong 0)$. Thus, the rotation matrix R^i is the one based on only yaw α_z^i.

4.1. Translational alignment sensitivity analysis

For the translational parameters, we introduce the perturbation $t_\Delta = [\Delta_x, \Delta_y, \Delta_z]'$. From equations (2) and (1), we have the new object coordinates due to the perturbation as follows

$$P_\Delta^i = R^i * \left(R_\beta * P_L^i + t + t_\Delta\right) + p^i = P^i + R^i * t_\Delta \quad (5)$$

Consequently, we have

$$P_\Delta^i - P_\Delta^j = (R^i - R^j) * t_\Delta \cong$$
$$-2\sin\left(\frac{\alpha_z^i - \alpha_z^j}{2}\right) \begin{pmatrix} \sin\left(\frac{\alpha_z^i + \alpha_z^j}{2}\right) * \Delta_x + \cos\left(\frac{\alpha_z^i + \alpha_z^j}{2}\right) * \Delta_y \\ \cos\left(\frac{\alpha_z^i + \alpha_z^j}{2}\right) * \Delta_x - \sin\left(\frac{\alpha_z^i + \alpha_z^j}{2}\right) * \Delta_y \\ 0 \end{pmatrix} \quad (6)$$

From the above equation, we have the following results: s-i) z-axis translational parameter t_z cannot be calibrated by minimizing object localization errors, because the change Δ_z has no effect on the localization error $(P_\Delta^i - P_\Delta^j)$. S-ii) The localization error $(P_\Delta^i - P_\Delta^j)$ by translational misalignment t_Δ does not depend on the vehicle locations, but depends on the vehicle orientation (yaw: α_z^i, α_z^j), which implies the distance from the vehicle to the object has no effect on the translational calibration. s-iii) In order to have big impacts on the localization error from translational misalignment, we cannot have $|\alpha_z^i - \alpha_z^j| \cong 0$ but better have $|\alpha_z^i - \alpha_z^j| \cong 180°$. In other words, the translational calibration cannot be performed during straight-line driving but at turning (*e.g.,* left turn, right turn or U-turn).

4.2. Orientational alignment sensitivity analysis

From Equations (1) and (2) we have

$$P_L^i = \left(R_\beta\right)' * [(R^i)' * (P^i - p^i) - t] \quad (7)$$

In Equation (7), $\left(R_\beta\right)' = \left(R_\beta\right)^{-1}$ and $(R^i)' = (R^i)^{-1}$. For any angular perturbation $\beta' = \beta + \delta$, from Equations (1), (2) and (7),

$$P_\delta^i = R^i * \left(R_{\beta+\delta} * P_L^i + t\right) + p^i \quad (8)$$

Further we have

$$P_\delta^i - P^i = R^i * \left(R_{\beta+\delta} - R_\beta\right) * \left(R_\beta\right)' * (R^i)' * (P^i - p^i - R^i * t) \quad (9)$$

Let $(P^i - p^i - R^i * t) = (a_i, b_i, c_i)'$, from Equations (1) and (2), $(a_i, b_i, c_i)' = R^i * R_\beta * P_L^i$, which is the static object coordinate as detected by LiDAR with the rotation to the earth-fixed system but without the translation to the earth-fixed system. $\left(a_j, b_j, c_j\right)' - (a_i, b_i, c_i)'$ is the position change of the object between time i and j in the earth-fixed coordinate system, which is also the opposite position change of the vehicle during this time in the earth-fixed coordinate system. We have $c_i - c_j \cong 0$ when the vehicle is on a flat road, which is usually the case.

Now for the roll angular perturbation $\beta_x' = \beta_x + \delta$, assuming δ is small, we have $cos\delta \cong 1$ and $sin\delta \cong \delta$. Next, we consider two different routes: (a) straight-line driving; (b) at turning. When doing straight-line driving starting from the initial state, vehicle yaw $\alpha_z^i \cong \alpha_z^j \cong 0$, vehicle forward movement $|a_j - a_i| > 0$, and vehicle left movement $b_i - b_j \cong 0$. From Equation (9), we can have (the detailed inference is omitted here due to the paper length restriction)

$$P_\delta^i - P_\delta^j \cong \delta * \begin{pmatrix} 0 \\ sin\beta_y * (a_j - a_i) \\ cos\beta_y * \sin\beta_z * (a_j - a_i) \end{pmatrix} \quad (10)$$

Since normally the pitch angular alignment β_y is small, the dominate localization error is on z-axis. S-iv) If the yaw angular alignment β_z is also small, then the roll angular boresight alignment cannot be calibrated during straight-line driving. S-v) If $\beta_z \cong 90°$, then the roll angular alignment can be calibrated during straight-line driving as long as $|a_j - a_i| \gg 0$, which is the forward movement of the vehicle.

When at turning, considering β_y is small and $c_i \cong c_j$, similar to Equation (10) we have

$$P_\delta^i - P_\delta^j \cong \delta * cos\beta_y * \begin{pmatrix} 2 * \sin\left(\frac{\alpha_z^i - \alpha_z^j}{2}\right) * \cos(\beta_z + \frac{\alpha_z^i + \alpha_z^j}{2}) * c_i \\ 2 * \sin\left(\frac{\alpha_z^i - \alpha_z^j}{2}\right) * \sin(\beta_z + \frac{\alpha_z^i + \alpha_z^j}{2}) * c_i \end{pmatrix} \quad (11)$$

$$w = -2 * \sin\left(\frac{\alpha_z^i - \alpha_z^j}{2}\right) * (\cos\left(\beta_z + \frac{\alpha_z^i + \alpha_z^j}{2}\right) * a_i + \sin\left(\beta_z + \frac{\alpha_z^i + \alpha_z^j}{2}\right) * b_i) + \sin(\beta_z + \alpha_z^j) * (a_j - a_i) + \cos(\beta_z + \alpha_z^j) * (b_i - b_j) \quad (12)$$

When the object is at the similar height as the LiDAR, then c_i is small and the dominate localization error is on z-axis. S-vi) From the above equation we know roll angular alignment can be calibrated at turning.

For the pitch angular perturbation $\beta_y' = \beta_y + \delta$, again, we consider the cases of straight-line driving and at turning. For straight-line driving, $\alpha_z^i \cong \alpha_z^j \cong 0$ and $b_i \cong b_j$. With the assumption of $c_i \cong c_j$, like Equation (10) we have

$$P_\delta^i - P_\delta^j \cong \delta * \begin{pmatrix} 0 \\ 0 \\ \cos\beta_z * (a_j - a_i) \end{pmatrix} \quad (13)$$

s-vii) If $\beta_z \cong 90°$ then pitch alignment cannot be calibrated during straight-line driving. S-viii) When $\beta_z \cong 0°$, the pitch alignment can be calibrated during straight-line driving, and it is better to have large vehicle location changes, *i.e.*, $(|a_j - a_i|) \gg 0$. s-ix) When at turning, considering $c_i \cong c_j$, similar as in Equation (11), the pitch angular alignment can be calibrated at turning.

For the yaw angular perturbation $\beta_z' = \beta_z + \delta$, like Equation (10) we have

$$P_\delta^i - P_\delta^j \cong \begin{pmatrix} \delta * (b_j - b_i) \\ \delta * (a_i - a_j) \\ 0 \end{pmatrix} \quad (14)$$

Therefore, s-x) the yaw angular alignment can be calibrated as long as the vehicle location change is big enough. Since the vehicle orientation has no impact on the localization error, the yaw alignment can be calibrated either during straight-line driving or at turning.

5. Algorithm

Based on the sensitivity analysis, at turning, the 5 parameters (Tx, Ty, Roll, Pitch and Yaw) are estimated using traffic sign data, namely 5P-optimization. During straight-line driving maneuvers, the 2 parameters (Roll and Yaw, due to $\beta_z \cong 90°$ in our test vehicle) are estimated using either lane markings or traffic signs, namely 2P-optimization. In the optimization (section 5.3), we derive R_β and t using all 6 parameters combining variables (2P or 5P) with fixed parameters. Due to s-i), Tz is calculated separately based on the ground point position after optimization and the predetermined vehicle-to-ground distance.

The whole algorithm flow is shown in Figure 1. The LiDAR data and GPS/IMU data are loaded. Each traffic sign and lane markings are extracted using the algorithms in 5.1 and 5.2. The raw LiDAR data in front of the vehicle is loaded for Tz calculation. For each frame, up to 10 points with the lowest z value are saved into the buffer. A moving buffer with up to N (*e.g.,* 5000) points is used to save the latest ground points.

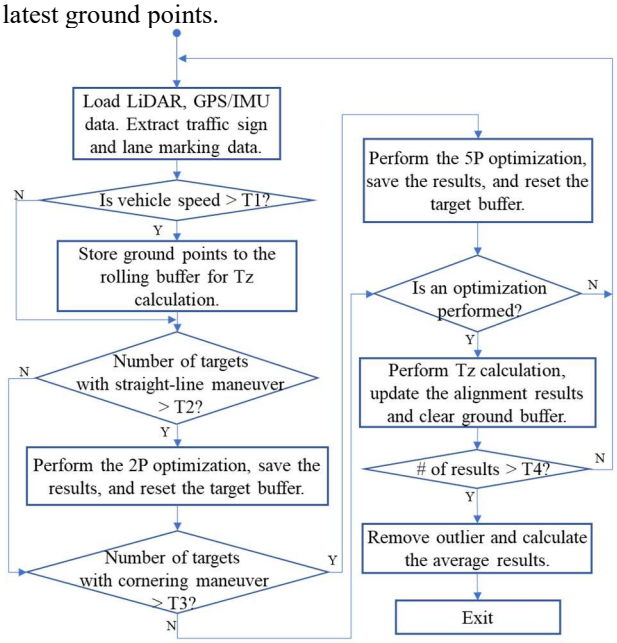


Figure 1: The flowchart of the traffic-sign-based alignment algorithm.

If the number of road elements (traffic signs or lane marks) during straight-line driving is greater than a threshold (*e.g.,* 7), an 2P optimization, described in section 5.3, will be conducted. Please note the traffic signs and lane marks are processed separately due to the difference of loss functions. In order to improve the algorithm robustness, multiple road elements are used for optimization rather than a single one in [16]. Similarly, 5P optimization is conducted with enough of traffic sign data at turning.

After an optimization is performed, Tz is calculated. The ground data are projected to the vehicle coordinate using the current estimated alignment parameters. Only the data with z values within [-0.4, 0.4] are kept. The mean value (noted as Avg_G) of filtered data is calculated. Tz is then calculated as $T_{z\_init} - Ins_{ground} - Avg_G$, where T_{z_init} is the current estimate of Tz, and Ins_{ground} is the distance between the vehicle center of gravity and ground, which is a predefined constant. This is because the z value of ground points in the LiDAR coordinate equals to Avg_G - T_{z_init},

and also equals to $-\mathrm{Tz}\ -Ins_{ground}$. The updated Tz result is pushed to the result buffer as well, and the buffer to store all ground point cloud data is cleared.

Once there are enough results in the buffer, the average and the standard deviation of all alignment results are calculated. The results, which are more than 1 standard deviation from the average value, are filtered. The average of filtered results will be used as the final alignment result.

5.1. Lane marking detection

A lane marking segment is of a rectangle shape on the ground. Lane markings captured during straight-line driving allow us to determine the Yaw and Roll. With GPS/IMU data, we enable the lane marking detection only when the vehicle is driving in a straight line.

An aggregated point cloud of a lane marking segment is generated in four steps: 1) detect candidate lane marking points in each LiDAR frame, 2) aggregate candidate points from consecutive frames in the earth-fixed coordinate system, 3) select a lane marking target based on current vehicle position, and 4) remove outlier points.

For each LiDAR frame, we first perform ground segmentation to select the points on the ground based on [27], which is based on the elevation and slope conditions of each LiDAR point. Next, we detect the candidate lane marking points from the ground points based on intensity. To precisely capture the boundaries of the lane markings, we detect the rising edge and falling edge of the intensity across the azimuth angle. Figure 2 shows an example of the intensity of the ground points associated with the same vertical emission angle (a scan line).

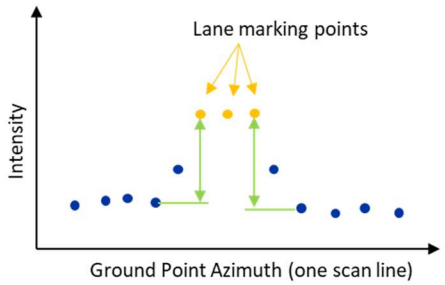


Figure 2: An example of intensity of ground points in one scan line.

We use a state machine to determine lane marking points shown in Figure 3. Starting with initial State 0 and sweeping towards right, if a rising edge is detected, the state goes to State 1, representing that a left bound of a lane marking is detected. We update the saved left bound if a new rising edge is detected and it is close to the previous rising edge. If we detect a falling edge, we save this falling edge as a right bound of the lane marking and go to State 2. We update the saved right bound if a new falling edge is detected and it is close to the previous falling edge. If we detect a rising edge, we save it as a new left bound, and go to State 1.

If we detect a falling edge while in State 0, we may start from the middle of a lane marking segment. Then we save the first ground point as the left bound and the detected falling edge as the right bound, and go to State 2.

The sweep most likely ends in State 2, with pairs of left and right bounds. If it ends in State 1, we may end the sweep in the middle of a lane marking segment. If the distance between the last left bound and the last ground point is small, we save the last ground point as a right bound; otherwise, we ignore the last left bound.

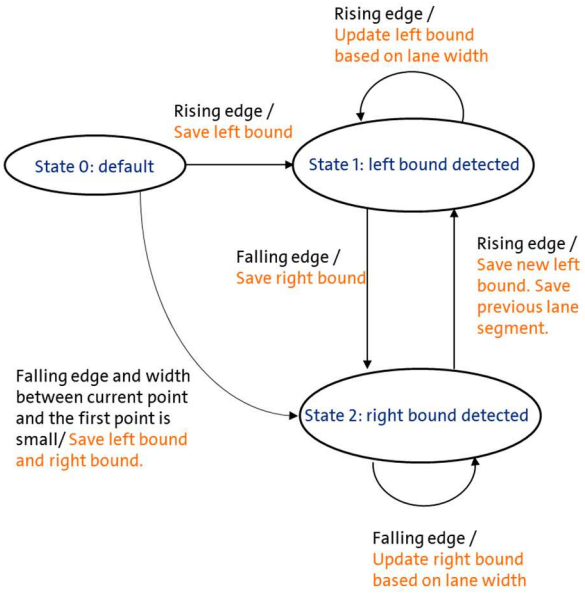


Figure 3: The state machine of sweeping a scan line of ground points.

After the sweep, we filter out the bound pairs whose widths are too large, and combine the consecutive pairs if they are very close and their widths are small. Finally, all the points located within each of bound pair are all labeled as candidate lane marking points. We perform the same process in each scan line in the current LiDAR frame. The detected candidate lane marking points are stored in a buffer.

Next in step 2), with Equation (1) and (2), we aggregate the candidate points from consecutive frames into the earth-fixed coordinate system with an initial guess of the LiDAR-to-vehicle alignment and the GPS/IMU data associated with each point. Then in step 3), we select a lane marking segment based on the region with respect to the current vehicle position. The points within the region form an aggregated point cloud of a target.

We observe some incorrect lane marking points with a high elevation above the actual ground, and they are far from the sensor. This is due to the accumulated elevation difference between the neighbors during ground

segmentation, and a point far from the sensor may have a high elevation while its slope is close to its neighbor.

Since the lane marking segment is close to a straight line in the 3D space, we perform 3D line fitting based on Hough transform [28] to remove the outliers. The kept inlier points form a point cloud of one target. They are saved as one target in a target buffer.

5.2. Traffic sign detection

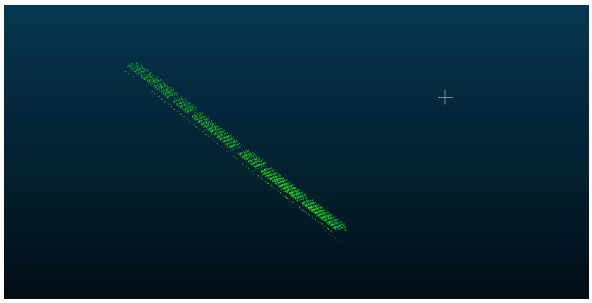
The traffic signs are detected based on the aggregated LiDAR point cloud using the initial guess of alignment. The aggregated data is then filtered by spatial and intensity. Considering the typical positions of traffic signs, we only keep the data with the distance between 6 meters and 100 meters in LiDAR coordinate. Since the traffic sign is usually covered by the highly reflective material, only the data with the intensity greater than 190 are kept.

Due to the sparsity of the LiDAR data, a certain number (*e.g.,* 500) of frames of data are aggregated in the earth-fixed coordinate. An unsupervised clustering algorithm, Density-based spatial clustering of applications with noise (DBSCAN [29]), is applied to label each LiDAR data point in the earth-fixed coordinate. The points that are close to each other are clustered as one traffic sign candidate. Since the same traffic sign may be observed in different batches, the center of each traffic sign candidate is calculated. If the center of the current traffic sign is close to the previous identified traffic sign, the data from both signs are combined as one target.

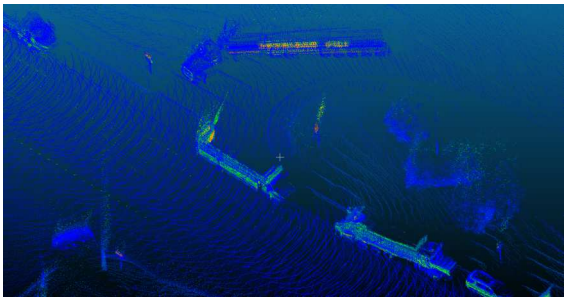
From the LiDAR alignment accuracy perspective, we remove some traffic sign candidates from the buffer, if the minimum speed of the ego vehicle is less than 5 m/s. When the ego vehicle speed is very low, the wheel speed sensor may generate wrong measurement due to the resolution limitation, which may generate inaccurate alignment results. Based on sensitivity analysis above, the traffic sign candidates are removed, if the change of the ego vehicle position is less than 40m. We also remove the candidates with the small amount of traffic sign points (*e.g.,* 120) or the large amount of traffic sign points (*e.g.,* 20,000).

The data spatial distribution is also checked to ensure a sign-like dimension. PCA is applied to each traffic sign candidate and generates three eigenvectors. These eigenvectors represent the orthogonal direction of the largest three variations for the traffic sign point cloud. Then all the data corresponding to that traffic sign candidate are projected to three eigenvectors. If the range of largest variation is too big (*e.g.,* greater than 8m), but the range of the smallest variation is too small (*e.g.,* less than 2m), we consider these are fake signs and remove them from the buffer (*e.g.,* semi-truck shown in Figure 4 (b) or other moving vehicles in Figure 4 (a)). The traffic sign candidates are removed, if the range of all three directions are large (*e.g.,* more than 6m).

Sometimes, there are multiple traffic signs detected, but close to each other. It may be due to multiple signs close to each other or one sign but with sparse LiDAR points. If it's the former case, multiple signs should be separated; if it's the latter case, these clusters should be combined. Figure 5 shows an example on the road.



(a)



(b)

Figure 4: Examples of fake traffic sign candidates.

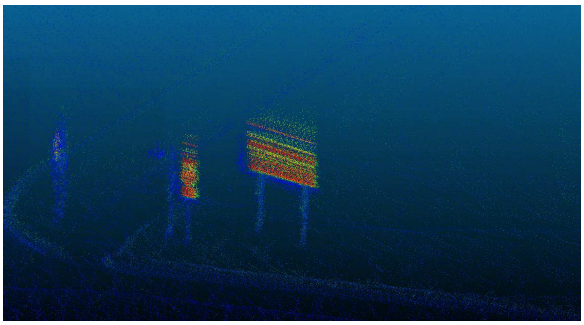


Figure 5: Examples of multiple traffic signs close to each other.

A clustering with a smaller threshold is employed and the following algorithm is used to separate or combine clusters. Firstly, the minimum distance between any two clusters is calculated. Here the distance is defined as the Euclidean distance in the earth-fixed coordinate. Then for each cluster, a list is generated with the indices including itself and the other clusters, whose distance to this cluster is less than a threshold (*e.g.,* 0.5 m). We then combine the lists sharing the same indices. For instance, cluster 1 is close to the clusters 3 and 5 (*i.e.* list(1) = {1,3,5}), and the cluster 2 is close to the clusters 5 and 6 (*i.e.* list(2) =

{2,5,6}). Since the list 1 and the list 2 share the same index 5, these two lists are combined into one list with the indices {1,2, 3, 5, 6}. Each list after combination is corresponding to an actual cluster (traffic sign). The distance between two clusters is at least 0.5m. We then combine all the data belonging to these indices in each list.

5.3. Optimization

The variance of the target point cloud is employed for the loss function, and the Adam optimization algorithm [30] is used to determine the alignment parameters by minimizing the loss. Denote each point k in the LiDAR coordinate for target n as $M_{n,k}$. With equations (1) and (2), we map it into the earth-fixed coordinate system, and denote its coordinates as $M_{n,k}^{w}$. We put all the projected points into one matrix M_n^w, each column of which is $M_{n,k}^{w}$. To determine the 3D variances of M_n^w, We calculate the center point of M_n^w:

$$M_{n,c}^{w} = \underset{k}{\text{mean}}\, M_{n,k}^{w} \tag{15}$$

Then we perform singular value decomposition:

$$\left(M_n^w - M_{n,c}^{w}\right)^T = U_n \begin{bmatrix} s_{n,1} & & \\ & s_{n,2} & \\ & & s_{n,3} \end{bmatrix} \begin{bmatrix} v_{n,1}^T \\ v_{n,2}^T \\ v_{n,3}^T \end{bmatrix} \tag{16}$$

where $s_{n,1}$, $s_{n,2}$ and $s_{n,3}$ are singular values in a descending order. Denote N_n as the number of points of M_n^w. Therefore, $\frac{s_{n,1}}{\sqrt{N_n}}$, $\frac{s_{n,2}}{\sqrt{N_n}}$ and $\frac{s_{n,3}}{\sqrt{N_n}}$ are corresponding to the standard deviations in the length, width and height direction of a road element (*e.g.,* lane mark segment or traffic sign), respectively. The optimal alignment $\left(R_\beta, t\right)^*$ should generate the minimal variances in the width and height directions for an aggregated lane mark segment. For a traffic sign, the variances of all three directions are minimized. Therefore, the optimization problem to determine the alignment parameters is formulated as (17) and (18), respectively:

$$\left(R_\beta, t\right)^* = \underset{(R_\beta, t)}{\text{argmin}} \sum_{n=1}^{L} \frac{s_{n,2} + s_{n,3}}{\sqrt{N_n}} \tag{17}$$

$$\left(R_\beta, t\right)^* = \underset{(R_\beta, t)}{\text{argmin}} \sum_{n=1}^{L} \frac{s_{n,1} + s_{n,2} + s_{n,3}}{\sqrt{N_n}} \tag{18}$$

where L is the number of targets used to generate each calibration result. For the traffic sign optimization, both 2P and 5P optimization are performed with (18).

6. Experiments

For validation, the proposed algorithms are tested with several recordings collected from different test vehicles on different days. The LiDAR used in these tests is Velodyne® VLP-32, mounted on the roof of the vehicle, whose vertical field of view is 48°. The RTK sensor is used to generate GPS/IMU information. A Dell laptop with Ubuntu 16.04, 31.3 GB memory and Intel Xeon CPU E3-1505M @2.8GHz is used to run all the tests. The Adam optimizer learning rate is set as 0.001, the number of steps is 1500. The ground truth values are calibrated and validated manually by our test fleet engineers.

6.1. Alignment results based on lane markings

This subsection presents the test results of the alignment algorithm using lane markings. This data is collected from a suburban area with 1040 lane marking segments detected as targets. We tune the algorithm based on this recording. With the same data, we perform 6 tests defined by a combination of initial guess and L value. For the initial guess of alignment parameters, we add 0.3-degree error on the ground truth values in Test 1, 2 and 3, and add 3-degree error in Test 4, 5 and 6. For L, the number of targets used in each optimization, we select L=10 in Test 1 and 4, L=20 in Test 2 and 5, and L=40 in Test 3 and 6.

Error is calculated by the output parameter value minus its ground truth value. 0 error means perfect accuracy. The mean and standard deviation of these errors of each test are listed in Table 1. Overall, the mean error of roll is about -0.3 degree, and the mean error of yaw is about ±0.1 degree. The mean error of roll is slightly bigger than the mean error of yaw.

Table 1: Alignment results using lane markings.

	Mean roll error (deg)	Mean yaw error (deg)	STD roll error (deg)	STD yaw error (deg)
Test 1	-0.320	0.096	0.097	0.575
Test 2	-0.314	0.098	0.073	0.511
Test 3	-0.316	0.062	0.055	0.356
Test 4	-0.294	-0.153	0.061	0.616
Test 5	-0.291	-0.119	0.049	0.409
Test 6	-0.295	-0.106	0.035	0.298

With bigger errors in the initial guess, the roll error doesn't change much, but the yaw error (absolute value) increased slightly. Moreover, the standard deviation of the yaw error is bigger than the standard deviation of the roll error. This can be interpreted by the geometry characteristic of the lane marks. The range in width is bigger than the range in height for a lane mark. In LiDAR point clouds, the lane mark points have a bigger variance in the width direction than the height direction. In addition, the standard deviations decrease when we increase the number of targets used in each calibration, which indicates more stable results.

The comparison with other existing methods is ignored in this test since no methods are found configurable to align only roll and pitch under straight line motion.

6.2. Alignment results based on traffic signs

The first test we conducted is to show the effectiveness

of using multiple traffic signs during optimization. The initial guess of alignment parameters is offset by 0.2 m for translation parameters and 3 degrees for rotation parameters from the ground truth values, which also applied to other tests in this section. The aggregated point cloud at turning is shown in Figure 6. Three traffic signs, noted in yellow, are used in alignment estimation. Our test results indicate that the estimation error with three traffic signs together is less than 0.4 degree, but the estimation error from each individual sign can be as high as 3.2 degree.

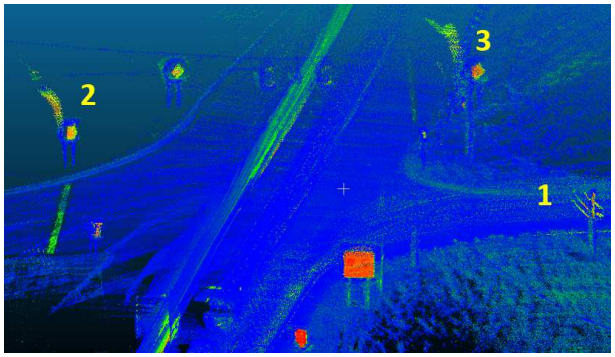


Figure 6: Aggregated LiDAR point cloud at turning.

We tune the algorithm using one recording, and test on three different recordings. The 2P and 5P optimization errors for one recording from one test vehicles are shown in Table 2 and Table 3 respectively. Each column is the estimation error. The translation error unit is meter and the rotation error unit is degree. Each row is the optimization result using 7 traffic signs. There are totally 27 2P results and 10 5P results. Due to the space limitation, only the maximum, minimum, average and standard deviation of 27 2P results are shown in Table 3. The vehicle was driven for 17.27 minutes, and the total driving distance is 11,955.9m. Total number of LiDAR frames is 156,200.

Table 2: Results for 5P optimization and Tz calculation

Sign	Tx error (m)	Ty error (m)	Tz error (m)	Roll error (deg)	Pitch error (deg)	Yaw error (deg)
1	-0.017	0.022	0.011	0.114	0.032	0.032
2	-0.126	0.055	0.003	0.080	-0.035	-0.046
3	0.055	-0.055	0.018	-0.101	-0.042	0.403
4	-0.044	-0.049	0.007	0.091	-0.109	-0.053
5	-0.032	-0.143	0.011	0.056	0.054	0.147
6	0.013	-0.045	0.012	-0.022	0.189	0.074
7	0.037	-0.012	0.013	-0.013	0.024	0.189
8	0.100	-0.015	0.033	0.034	-0.067	0.316
9	0.052	-0.041	0.027	0.023	0.047	0.226
10	-0.074	-0.048	0.014	0.126	0.541	0.267
Avg.	-0.004	-0.033	0.015	0.039	0.063	0.156
Std.	0.068	0.053	0.009	0.070	0.187	0.153

The alignment results of multiple recordings from various vehicles are shown in Table 4. Each row indicates the average alignment error using all the data from the whole recording. It can be clearly observed that our dynamic LiDAR alignment algorithm can achieve high alignment accuracy.

We compare with another two existing methods in this test using the same data of the test of Table 2 and 3. Hand-eye + LOAM uses the registration method in [31] and solution in [3] to perform the hand-eye calibration. Linkalibr is introduced in [11] using NDT and EKF to perform alignment with localization. The results are shown in Table 5. The results show our approach achieves better accuracy in translation and roll. Pitch is better than Linkalibr. Yaw is close among these three approaches. With further case study, the root cause of inaccuracy in hand-eye + LOAM and Linkalibr is mainly the LiDAR point cloud registration error. This shows our approach is robust under such challenging scenarios.

Table 3: Results for 2P optimization and Tz calculation.

Results	Tz error (m)	Roll error (deg)	Yaw error (deg)
Maximum	0.042	0.211	0.461
Minimum	-0.001	-0.103	-0.096
Average	0.020	0.045	0.188
Standard Deviation	0.012	0.080	0.136

Table 4: Average alignment error for different vehicles.

Veh	Distance (m)	Processing Time (mins)	Tx error (m)	Ty error (m)	Tz error (m)	Roll error (deg)	Pitch error (deg)	Yaw error (deg)
1	11676	191.4	0.000	0.007	0.004	0.048	0.005	0.009
2	11956	146.2	-0.004	-0.033	0.018	0.044	0.063	0.179
3	12313	142.9	0.012	0.003	0.000	0.017	0.012	0.032

Table 5: Average alignment error from different methods.

Method	Tx error (m)	Ty error (m)	Tz error (m)	Roll error (deg)	Pitch error (deg)	Yaw error (deg)
Ours based on traffic signs	-0.004	-0.033	0.018	0.044	0.063	0.179
Hand-eye [3] + LOAM [31]	0.399	0.215	2.467	0.131	0.077	0.188
Linkalibr [11]	0.427	0.161	1.161	1.515	1.517	0.155

7. Conclusion

This paper presents an online LiDAR-to-vehicle alignment approach using road elements including lane markings, traffic signs, and ground point position. Based on the sensitivity analysis, appropriate vehicle routes with the most excitation for different parameters are selected. The approach is validated by the test vehicle data. The results show that the mean error during about 12KM driving can achieve up to 0.2-degree error on orientation and 2cm error on translation, which demonstrates the effectiveness of developed approaches. This approach relies on the availability and accuracy of the road elements. In the future, we will study on methods to cover other different targets to improve the usability and robustness.

Acknowledgement

This work is supported by Wende Zhang, Hao Yu, Yilu Zhang, Paul Krajewski, Wen-Chiao Lin, Shiming Duan, Sarah Gagnon, Michael Wahlstrom, Joaquin Carcache, and Joseph Jang at General Motors.

References

[1] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur and X. Savatier, "A study of Vicon system positioning performance," *Sensors,* vol. 17, no. 7, pp. 1591-1609, 2017.

[2] A. Diehm, J. Gehrung, M. Hebel and M. Arens, "Extrinsic self-calibration of an operational mobile LiDAR system," *Proceedings of SPIE, Laser Radar Technology and Applications XXV,* vol. 11410, pp. 1-17, 2020.

[3] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International of Robotics Research,* vol. 14, no. 3, pp. 195-210, 1995.

[4] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," *IEEE Transacations on Robotics and Automation,* vol. 14, no. 4, pp. 617-622, 1998.

[5] A. Gostar, C. Fu, W. Chuah, M. Hossain, R. Tennakoon, A. Bab-Hadiashar and R. Hoseinnezhad, "State transition for statistical SLAM using planar features in 3D point clouds," *Sensors,* vol. 19, pp. 1614-1630, 2019.

[6] J. Zhang, M. Kaess and S. Singh, "On degeneracy of optimization-based state estimation problems," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

[7] M. Shah, R. D. Eastman and T. Hong, "An overview of robot-sensor calibration methods for evaluation of perception systems," in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, College Park, MD, USA, 2012.

[8] A. Dekel, L. H¨arenstam-Nielsen and S. Caccamo, "Optimal least-squares solution to the hand-eye calibration problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[9] J. Liu, J. Wu and X. Li, "Robust and accurate hand–eye calibration method based on Schur matric decomposition," *Sensors,* vol. 19, no. 20, 2019.

[10] C. L. Gentil, T. Vidal-Calleja and S. Huang, "IN2LAAMA: inertial LiDAR localisation autocalibration and mapping," *IEEE Transactions on Robotics,* vol. 37, no. 1, pp. 275-290, 2021.

[11] S. Mishra, G. Pandey and S. Saripalli, "Target-free Extrinsic Calibration of a 3D-Lidar and an IMU," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Karlsruhe, Germany, 2021.

[12] J. Lv, J. Xu, K. Hu, Y. Liu and X. Zuo, "Targetless Calibration of LiDAR-IMU System Based on Continuous-time Batch Estimation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2020.

[13] J. Lv, X. Zuo, K. Hu, J. Xu, G. Huang and Y. Liu, "Observability-Aware Intrinsic and Extrinsic Calibration of LiDAR-IMU Systems," *IEEE Transactions on Robotics,* vol. 38, no. 6, pp. 3734 - 3753, 2022.

[14] P. Rieger, N. Studnicka, M. Pfennigbauer and G. Zach, "Boresight alignment method for mobile laser scanning systems," *Journal of Applied Geodesy,* vol. 4, no. 1, pp. 13-21, 2010.

[15] F. Yuan, G. Li, Z. Zuo, D. Li, Z. Qi, W. Qiu and J. Tan, "Airborne LIDAR borsight error calibration based on surface coincide," *IOP Conference Series: Earth and Environmental Science,* vol. 17, 2014.

[16] R. Sands and B. Leslie, "OxTS boresight alignment determination with Velodyne VLP-16 LiDAR (AE021)," 2017. [Online]. Available: https://www.oxts.com/app/uploads/2017/11/ae021_report_171127.pdf.

[17] Z. Li, J. Tan and H. Liu, "Rigorous boresight self-calibration of mobile and UAV LiDAR scanning systems by strip adjustment," *Remote Sensing,* vol. 11, no. 4, 2019.

[18] S. Kammel and B. Pitzer, "Lidar-based lane marker detection and mapping," in *2008 IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, 2008.

[19] M. Thuy and F. P. León, "Lane detection and tracking based on lidar data," *Metrology and Measurement Systems,* vol. 17, no. 3, pp. 311-321, 2010.

[20] B. Yang, L. Fang, Q. Li and J. Li, "Automated extraction of road markings from mobile LiDAR point clouds," *Photogrammetric Engineering & Remote Sensing,* vol. 78, no. 4, pp. 331-338, 2012.

[21] A. Hata and D. Wolf, "Road marking detection using LiDAR reflective intensity data and its application to vehicle localization," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, China, 2014.

[22] F. Ghallabi, F. Nashashibi, G. El-Haj-Shhade and M.-A. Mittet, "LiDAR-based lane marking detection for vehicle positioning in an HD map," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, 2018.

[23] C. Lin, Y. Guo, W. Li, H. Liu and D. Wu, "An automatic lane marking detection method with low-density roadside LiDAR data," *IEEE Sensors Journal,* vol. 21, no. 8, pp. 10029 - 10038, 2021.

[24] Y. Zhou and O. Tuzel, "Voxelnet: end-to-end learning for point cloud based 3D object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] D. Xu, D. Anguelov and A. Jain, "Pointfusion: deep sensor fusion for 3D bounding box estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] G. P. Meyer, a. Laddha, E. Kee, C. Vallespi-Gonzalez and C. K. Wellington, "LaserNet: an efficient probabilistic 3D object detector for autonomous driving," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[27] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3D laser scans," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science,* vol. 85, pp. 41-52, 2017.

[28] P. V. C. Hough, "Method and means for recognizing complex patterns". United States of America Patent US3069654A, 18 December 1962.

[29] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[30] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *The 3rd International Conference for Learning Representations (ICLR)*, San Diego, 2015.

[31] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Proceedings of Robotics: Science and Systems (RSS '14)*, 2014.