Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5866868, 12 pages
https://doi.org/10.1155/2022/5866868

Research Article

High-Precision Motion Compensation for LiDAR Based on LiDAR Odometry

Peilin Qin, Chuanwei Zhang, Xiaowen Ma, and Zhenghe Shi

School of Mechanical Engineering, Xi'an University of Science and Technology, Xi'an, 710054 Shaanxi, China

Correspondence should be addressed to Chuanwei Zhang; 20105016012@stu.xust.edu.cn

Received 8 March 2022; Revised 29 March 2022; Accepted 7 April 2022; Published 9 May 2022

Academic Editor: Chia-Huei Wu

LiDAR plays a pivotal role in the field of unmanned driving, but in actual use, it is often accompanied by errors caused by point cloud distortion, which affects the accuracy of various downstream tasks. In this paper, we first describe the feature of point cloud and propose a new feature point selection method Soft-NMS-Select; this method can obtain uniform feature point distribution and effectively improve the result of subsequent point cloud registration. Then, the point cloud registration is completed through the screened feature points, and the odometry information is obtained. For the motion distortion generated in a sweep, the prior information of the LiDAR's own motion is obtained by using two linear interpolations, thereby improving the effect of motion compensation. Finally, for the distortion caused by the motion of objects in the scene, Euclidean clustering is used to obtain the position and normal vector of the center point of the point cloud cluster, and the motion pose of the object is calculated according to the offset between adjacent sweeps and eliminated distortion. Essentially, our method is a general point cloud compensation method that is applicable to all uses of LiDAR. This paper inserts this method into three SLAM algorithms to illustrate the effectiveness of the method proposed in this paper. The experimental results show that this method can significantly reduce the APE of the original SLAM algorithm and improve the mapping result.

1. Introduction

As the most important sensor in the robot perception system, LiDAR plays an indispensable role in object detection, localization, and mapping in the field of unmanned driving. According to its working principle, LiDAR can be divided into conventional LiDAR, solid-state LiDAR, and hybrid solid-state LiDAR. Among them, conventional LiDAR, with the longest development time, is also the most widely used.

However, the conventional LiDAR is limited by the working principle of its rotary scanning, so the point cloud compensation problem in one sweep must be considered. For example, Figure 1 shows the position relation of a vehicle carrying LiDAR at three moments. Assuming that these three moments are all within a LiDAR scanning cycle, for obstacle P, there are multiple observation positions in a sweep, which makes it difficult to accurately describe the location information of the obstacle.

Furthermore, if objects in the scene are moved during scanning, the point cloud of the moving object will be distorted, as shown in Figure 2. At this point, while the LiDAR is moving, the object to be measured is also moving, which causes the measurement deviation to increase. Most LiDAR downstream tasks choose to ignore the impact of this problem. For example, in the well-known LOAM [1] framework, only the distortion caused by LiDAR movement is calibrated. However, we find that the negative impact of the distortion caused by the movement of objects in the scene on various applications of LiDAR cannot be ignored.

Based on the above problems, we propose a LiDAR motion compensation method. Through a new feature point selection method of point cloud, the planar entropy and linear entropy of the points are firstly calculated, and then, Soft-NMS-Select is used to obtain the feature description of the current sweep. Then, the prior information of LiDAR's own motion was obtained by linear interpolation twice to improve the orthodontic effect. At the same time,

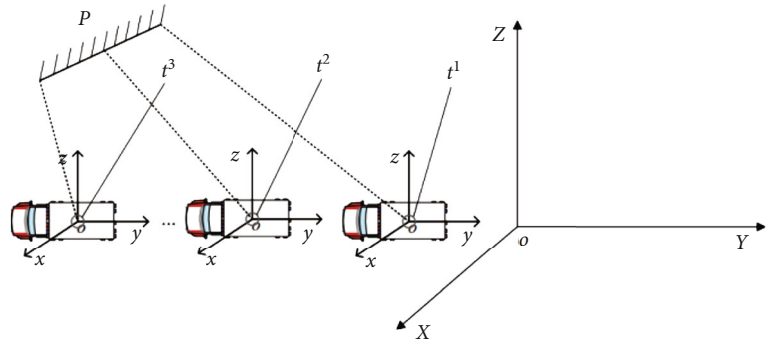


Figure 1: Distortion caused by LiDAR's own motion. In the figure, the coordinate system on the right is the initial coordinate system of a sweep. The vehicles in the figure have multiple positions in the current sweep period, so there are multiple observation coordinate systems, resulting in the description of object P under measurement in multiple observation coordinate systems, thus affecting the measurement result.

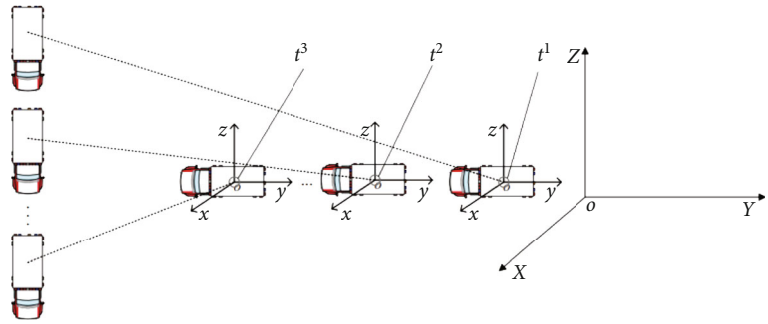


Figure 2: Distortion caused by the motion of the object to be measured.

the coordinates and normal vectors of the center point of the point cloud cluster are calculated by Euclidean clustering, and then, the movement posture of objects in the scene is estimated, and the point cloud of moving objects in the scene is also compensated, so that the scene information can be described more accurately.

In addition, we selected two laser SLAM algorithms, A-LOAM and LEGO-LOAM, as the benchmark, and compared absolute pose error (APE) to quantitatively analyze and demonstrate the effectiveness of the proposed method. The structure of the rest of this paper is as follows: in Section 2, we introduce the methods commonly used to calculate motion posture and the general motion compensation methods of laser SLAM. In Section 3, the proposed motion compensation algorithm is introduced in detail. In Section 4, we provide the experimental effect under the actual scene. The limitations and deficiencies of this method are discussed in Section 5. Finally, we summarize this article.

2. Related Works

When solving motion pose, it is often necessary to match the point cloud between adjacent frames. As one of the mainstream registration algorithms, the ICP algorithm [2, 3] iteratively optimizes the conversion between source point cloud and target point cloud by minimizing the error measure between the two point clouds. Based on the matching of the ICP algorithm, other scholars further proposed point-to-line ICP [4], point-to-point ICP [5], and generalized ICP [6], making great contributions to the accuracy of point cloud registration. As the accuracy of point cloud registration is improved, the ICP algorithm is used in LiDAR odometer to reduce the accumulated error of continuous registration by combining the closed-loop mechanism and pose construction process [7]. The ICP algorithm is improved by means of downsampling and point cloud matching rejection, and the location drift of LiDAR odometer is reduced obviously. Although the ICP algorithm can be very accurate in point cloud conversion, it has a slightly insufficient requirement on real-time performance in automatic driving. Therefore, in order to improve the efficiency of the ICP algorithm, some articles use parallel computing for operation [8–11]. When ICP is used for matching scanning, if the scanning motion is relatively slow, serious motion distortion will occur. Therefore, researchers use laser intensity return to create visual images and match visually different features between images [12] to restore the motion of ground vehicles [13–16]. However, these methods require dense point cloud images. In [17–19], sparse point cloud images can be used for motion recovery, and point clouds can be used to match the geometric structure of local point clusters [19].

After completing interframe matching, LiDAR motion compensation is required. Mainstream laser SLAM

algorithms [20–22] unify the point cloud within a sweep to a timestamp, and this unified time point is often the start time of the scan. Then, using the result of the last interframe laser odometer as the motion between the current two frames and assuming that the current frame is also moving at a uniform speed, we can also estimate the position and pose of each point relative to the starting time. That is, the motion of $k-1$ to K frame and k to $K+1$ frame is one to one, and the pose transformation of $k-1$ to K frame is used as the pose transformation of K to $K+1$ frame, and the pose transformation of each point of K to $K+1$ frame can be calculated. However, this method does not accurately describe the motion posture of the current frame and lacks motion compensation for the objects to be measured in the scene. Therefore, based on the shortcomings of the above method, this paper improves the LiDAR motion compensation method and uses twice linear interpolation to obtain more accurate motion posture of the current frame. Using the description of the center point of the object category and the normal vector to solve its motion posture, complete the distortion removal of the moving object.

3. Methodology

3.1. Selection of Point Cloud Feature Points. The LiDAR used in this paper is Velodyne 16, and the laser points under a sweep reach more than 30,000. If all points are used for point cloud registration, it will cause great computing overhead. Therefore, we use the idea of LOAM for reference and select key points in the scene to participate in the subsequent registration work. In addition, before the feature description of the point cloud in a sweep, this paper did not adopt any downsampling filtering method as the pretreatment, because we found that the filtering method would reduce the compensation effect of subsequent moving objects.

Before calculating the feature description of point cloud, PCA should be performed on the processed point, and the eigenvalues of its covariance matrix should be calculated, which are arranged in descending order as follows: λ_1, λ_2, and λ_3. Its planarity is described as follows:

$$R = \frac{\lambda_2 - \lambda_3}{\lambda_1} = \frac{e_2 - e_3}{e_1} \text{which } e_i = \frac{\lambda_i}{\sum_{i=1}^{i} \lambda_i}. \tag{1}$$

It can be seen from the above formula that the flatness depends on the size of the third principal component in the overall composition. Approximately, as the value of the third principal component changes from large to small, the value of the flatness characteristic R increases from small to large. This means that in point cloud space, part of the point cloud space with a larger R value is closer to the plane, and part of the point cloud space with a smaller R value has a steeper change. Similarly, the linearity of point cloud can be described as

$$L = \frac{\lambda_1 - \lambda_2}{\lambda_1} = \frac{e_1 - e_2}{e_1}. \tag{2}$$

This indicates that part point clouds with a larger L value in point cloud space are more concentrated in one direction, and part point clouds with a smaller L value are more divergent in a certain direction. Figure 3 shows the key points selected by R and L as feature description, respectively.

In addition, this paper finds that the introduction of entropy of local feature description can improve the consistency of the selection of key points of adjacent frames, because entropy represents the uncertainty of information, so selecting a larger entropy value can screen out more significant neighborhood features, which is specifically expressed as

$$E_\lambda = -\sum_{i=1}^{3} e_i \ln e_i. \tag{3}$$

According to the above analysis, points with significant features are characterized by low flatness or linearity and high entropy. Therefore, by introducing flatness and linearity into the above equation, the flatness entropy and linearity entropy can be obtained as follows:

$$\begin{aligned} E_r &= \frac{R}{E_\lambda} = \frac{(e_2 - e_3)/e_1}{-\sum_{i=1}^{3} e_i \ln e_i}, \\ E_l &= \frac{L}{E_\lambda} = \frac{(e_1 - e_2)/e_1}{-\sum_{i=1}^{3} e_i \ln e_i}. \end{aligned} \tag{4}$$

Therefore, the feature point selection problem can be described as

$$c_i = \begin{cases} \operatorname{argmin} E_r, \\ \operatorname{argmin} E_l, \end{cases} \quad c_i \in C, \tag{5}$$

where $C = \{c_1, c_2, \cdots c_i\}$ is the key point collection. In order to ensure the real-time performance of the algorithm, a total of 10 points are selected for a sweep, and 5 points are selected by plane entropy value and linear entropy value, respectively. In addition, in order to ensure the uniform distribution of key points, the idea adopted in LOAM is to divide the scanning region of LiDAR and select several feature points in each subregion for subsequent registration. However, in some special scenarios, the distribution of key points is more concentrated.

See Figure 4. Key points in area 2 and area 3 do not cover a large number of scene features, so the accuracy of subsequent registration will be reduced. In this paper, a Soft-NMS-Select method is proposed to solve the above problems. Firstly, a neighborhood radius τ is determined, and the intersection volume between each sphere centered on key points is calculated. Considering that, using NMS directly will cause the algorithm to suppress most significant feature points according to the sphere radius in some extreme scenarios. Meanwhile, low feature points are selected.

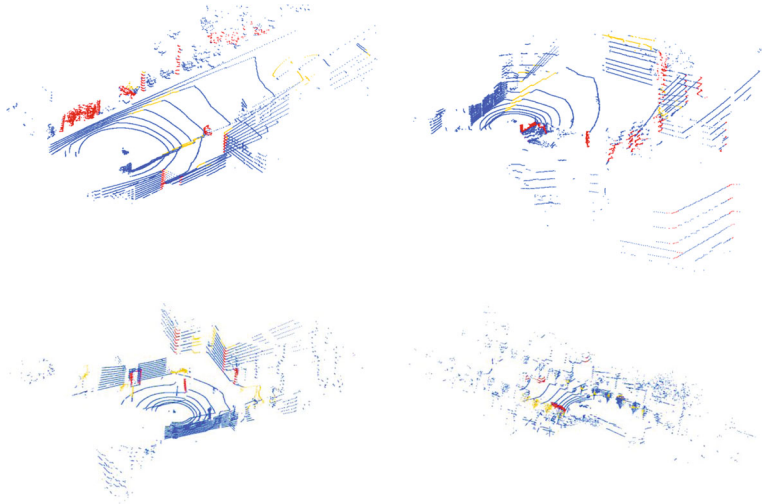


Figure 3: Key point effect diagram. Among them, the red point represents the screening point of plane entropy value, and the yellow point represents the screening point of linear entropy value.

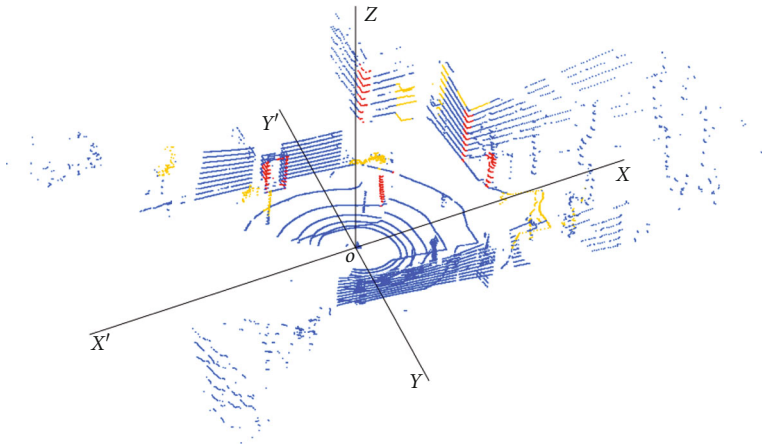


Figure 4: Schematic diagram of scanning area division effect.

Therefore, this paper uses a soft method to suppress nonmaxima to achieve uniform selection. The pseudo-code of Soft-NMS-Select is shown in Algorithm 1.

Using different radii τ, 5 to 10 to 15, 20 key point distribution is shown in Figure 5. It can be seen from the figure that a relatively uniform distribution of key points can be obtained by using $10 \le \tau \le 15$ for urban road scenarios. In addition, it can be seen from the figure that key points are mainly distributed at locations with drastic changes in the scene, such as corners and bulges. Because these locations often cover the uniqueness of the current scene, the key points calculated by using the above feature description can well represent the information of the current sweep and meet the requirements of subsequent point cloud registration.

3.2. Dedistortion Based on LiDAR Odometer. After obtaining the key points of the point cloud, odometer information needs to be calculated according to the registration result, so as to obtain the LiDAR movement posture at each time. The key points at moment t_k and moment t_{k+1} are, respectively, denoted as $Q_{(k,i)} = \{q_{(k,1)}, q_{(k,2)} \cdots, q_{(k,i)}\}$ and $Q_{(k+1,i)} = \{q_{(k+1,1)}, q_{(k+1,2)} \cdots, q_{(k+1,i)}\}$. The motion posture of point cloud $t_k \longrightarrow t_{k+1}$ is denoted as

$$t_k = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} t_{k+1}, \qquad Q_{(k+1,i)} = R \cdot Q_{(k,i)} + T, \tag{6}$$

where R is the 3×3 rotation matrix and T is the 3×1 translation matrix. At this point, the matching key point problem is essentially a point-to-point ICP problem. The point-pair matching is shown in Figure 6.

It can be described as follows:

$$\Delta J = \operatorname{argmin} \frac{1}{2} \sum_{1}^{i} \left\| Q_{(k+1,i)} - \left(R \cdot Q_{(k,i)} + T \right) \right\|^2, \Delta J = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}. \tag{7}$$

Input: key points set (C), radius τ, volume threshold ξ
Output: new key points set (C^{Select})={ $c_1, c_2, \cdots c_{10}$}
Create $C^{Select} \longleftarrow$ {}
$C = C$.sorted(max $\longrightarrow$ min)
while$C$.size > 0 **or**C^{Select} =10 **do**
C^{Select}.append($c^{\max}$)
ifdo
C^{Select}.append(c_i)
C.remove(c_i)
elsedo
ifdo
C^{Select}.append(c_i)
C.remove(c_i)
else do
C^{Select}.append($c_i \cdot (1 - sphere^{volume})$)
C.remove(c_i)
end if
end if
end while
return key points set (C^{Select})

Algorithm 1: Soft-NMS-Select.

Then, singular matrix S is constructed by two decentralized point clouds, respectively, and SVD is performed on S to obtain the current optimal translation matrix T:

$$\begin{aligned} &S = X \cdot Y^T, X = [x_1, x_2, \cdots .x_i] Y = [y_1, y_2, \cdots .y_i]^T, \\ &\text{which } x_i = Q_{(k,i)} - \overline{Q_{(k)}} y_i = Q_{(k+1,i)} - \overline{Q_{(k+1)}}, \\ &S = U \Sigma V^{\mathrm{T}}, \\ &R = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \det\left(VU^T\right) \end{bmatrix} \cdot U^T, \\ &T = \overline{Q_{(k+1)}} - R \cdot \overline{Q_{(k)}}. \end{aligned} \tag{8}$$

The optimal solution can be obtained through continuous iteration $J = (R|T)$.

Generally, the use of laser speedometer for LiDAR is an important premise to the distortion of the uniform model assumption, the assumption of LiDAR between adjacent sweep movements is uniform, and with a moment of motion being equal, using J represented the moment point cloud motion matrix; the movement of the middle time i gesture by linear interpolation can be represented as

$$J_i = \frac{t_i - t_{k+1}}{t_k - t_{k+1}} \cdot J. \tag{9}$$

The above formula can be used to calibrate the point cloud at the intermediate time to the coordinate system at $k+1$ with a uniform velocity model, so as to realize LiDAR motion compensation. In this paper, it is found through experiments that higher compensation accuracy can be obtained by linear interpolation of rotation matrix R and shift matrix T, denoted as $J = (R|T)$. For the translation matrix $T = [t_x, t_y, t_z]$, since the first-order difference can approximately represent the change trend of the function at the current point, the translation change at moments t_{k-1}, t_{k-2} can be used to approximately deduce the translation matrix at t_k, and then, linear interpolation can be used to obtain the translation change at intermediate moments to obtain higher accuracy:

$$T_i = \frac{t_i - t_{k+1}}{t_k - t_{k+1}} \cdot \left(\frac{t_{k-1} - t_{k-2}}{(k-1) - (k-2)} \cdot k \right). \tag{10}$$

As for the rotating attitude, classical Euler angle $(x - y - z)$ is used in this paper, so the Euler angle-rotation matrix transformation needs to be carried out using Rodrigues' formula and represents the Euler angle of LiDAR motion at the moment t_i:

$$\begin{aligned} \theta_i &= \frac{t_i - t_{k+1}}{t_k - t_{k+1}} \cdot ((\theta_{k-1} - \theta_{k-2}) \cdot k), \\ R_i \cdot v &= \left(\cos\theta_i \cdot I + (1 - \cos\theta_i) k k^T + \sin\theta_i \cdot K \right) \cdot v. \end{aligned} \tag{11}$$

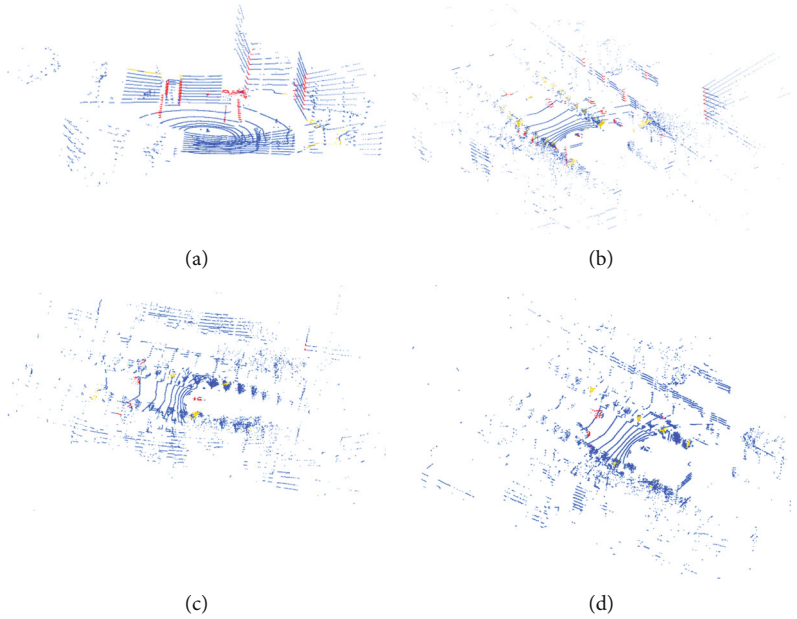
Let X_i represent the point cloud at time i, so the motion posture after two linear interpolation can be described as follows:

$$X_k = R_k \cdot X_{k+1} + T_k. \tag{12}$$

The point cloud effect after two interpolation is shown in Figure 7. Compared with single linear interpolation, the calibration effect of LiDAR carrier under acceleration or deceleration is improved because first-order difference is used to calculate the rate of change.

3.3. Moving Object Compensation Based on Euclidean Clustering. Another innovation of this method is to calculate the movement posture of objects scanned by LiDAR, so as to realize rough compensation. Therefore, after the key points set in the scene are extracted, the point cloud in the current sweep needs to be clustered. Due to the randomness of the scene, it is impossible to predict the number of categories in advance, so this paper selects European clustering to process the point cloud. Euclidean clustering is a clustering algorithm based on Euclidean distance; that is, the Euclidean distance between points is calculated to determine whether some point clouds belong to the same class, and the distance threshold is set to ξ. For the point clouds after clustering, the mean values of point clouds within the class are calculated, respectively, to obtain the location of the center point of the cluster. The pseudo-code of the Euclidean clusters and extraction center is shown in Algorithm 2.

The effect of realizing Euclidean clustering of objects in the scene by using the above methods is shown in Figure 8.



(a) (b) (c) (d)

Figure 5: Filtering effects of key points in scenarios with different thresholds. (a–d) The key point filtering results when τ is 5, 10, 15, and 20. It can be seen from the figure that when τ is 5, the distribution of key points in the scene is still redundant; when $\tau = 20$, the value of key points is too sparse, which is not conducive to subsequent feature point matching.

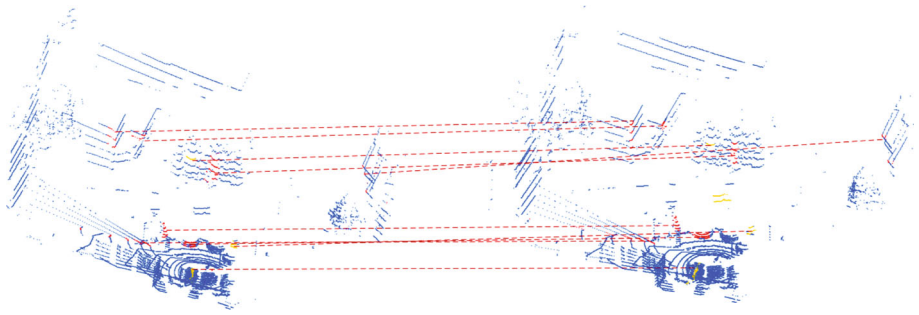
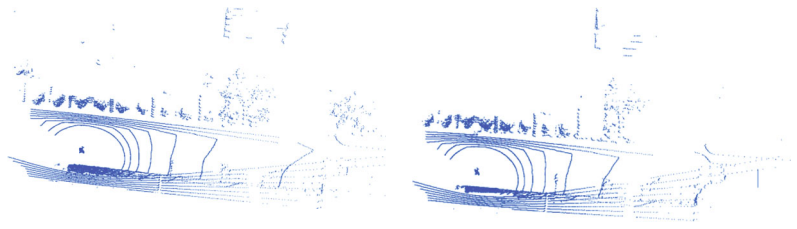


Figure 6: Matching diagram of cloud key points of adjacent frame points.



(a) The original point cloud (b) Postcompensation point cloud

Figure 7: Compensation rendering of LiDAR point cloud.

The coordinate transformation of the central point set to its adjacent sweep inner central point set can be described as follows:

$$P_k = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot P_{k+1}. \tag{13}$$

The actual center set is $R_{k+1} = \{r_1, r_2, r_3 \cdots r_n\}$. For the matching center point pair $p_i \longrightarrow r_i$, is the threshold of minimum matching distance. If $|p_i - r_i| < \varphi$, it is approximately considered that no absolute motion has occurred in the corresponding category. Therefore, no motion state estimation is performed on the point cloud set corresponding to the center point. If $|p_i - r_i| \geq \varphi$, then we calculate the corresponding central point translation matrix $T^{\text{trans}} = [t_x, t_y, t_z]$,

Input: Point cloud(C)={ $c_1, c_2, c_3 \ldots c_n$ } and distance threshold ξ
Output: Point set(P)={... c_i } and center point M =($\bar{x}, \bar{y}, \bar{z}$), which $\bar{x} = 1/i\sum x_i$, $\bar{y} = 1/i\sum y_i$, $\bar{z} = 1/i\sum z_i$
create $P \longleftarrow$ {}
create $K \longleftarrow$ {}
for c in $C$and $c$not in (P) **do**
find k nearest neighbour(K) = $\{k_1, k_2, k_3 \cdots k_n\}$
for k in K **do**
ifand$c$not in (P)**then**
P.append(k)
end if
calculate M= P^{mean}
return set(P), center M

Algorithm 2: Euclidean clusters and extraction center.

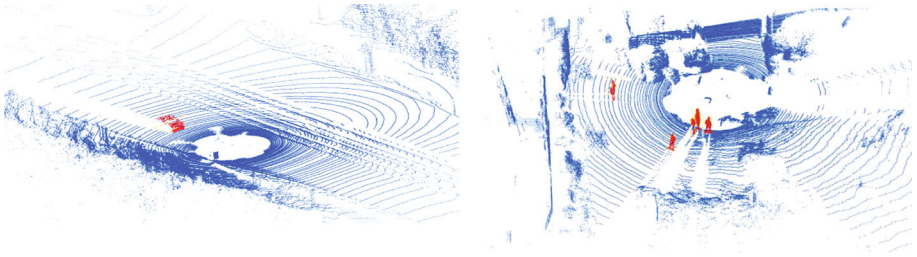
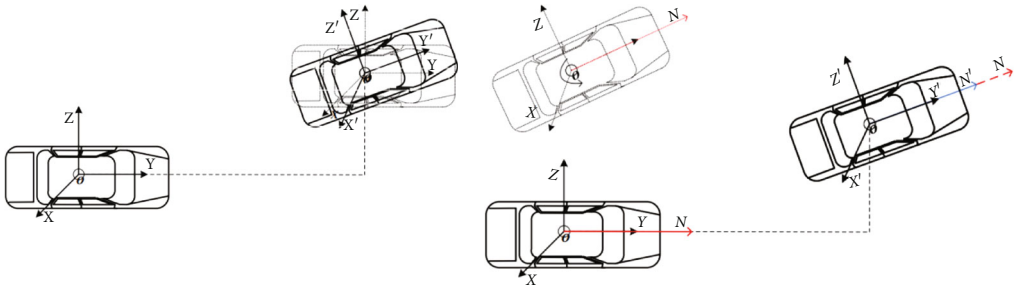
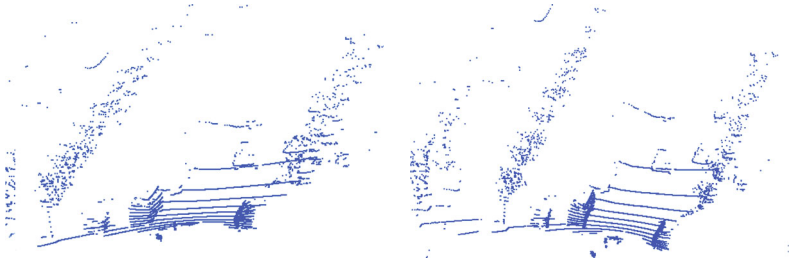


Figure 8: European clustering effect diagram. Red dots represent clustering points.



(a) Translational transform (b) Rotation transformation

Figure 9: Principle diagram of point cloud dedistortion of the moving object. The vector N is the normal vector of the object. (a) shows that the coordinate offset of the center point is used to estimate the translation transformation of the object, and (b) estimates the rotation transformation of the object by the offset of the normal vector before and after the transformation.



(a) The original point cloud (b) Point cloud after compensation

Figure 10: Compensation effect of the moving object. It can be seen that the distribution of point cloud of vehicles in (a) has obvious distortion, while that in (b) has obvious change.

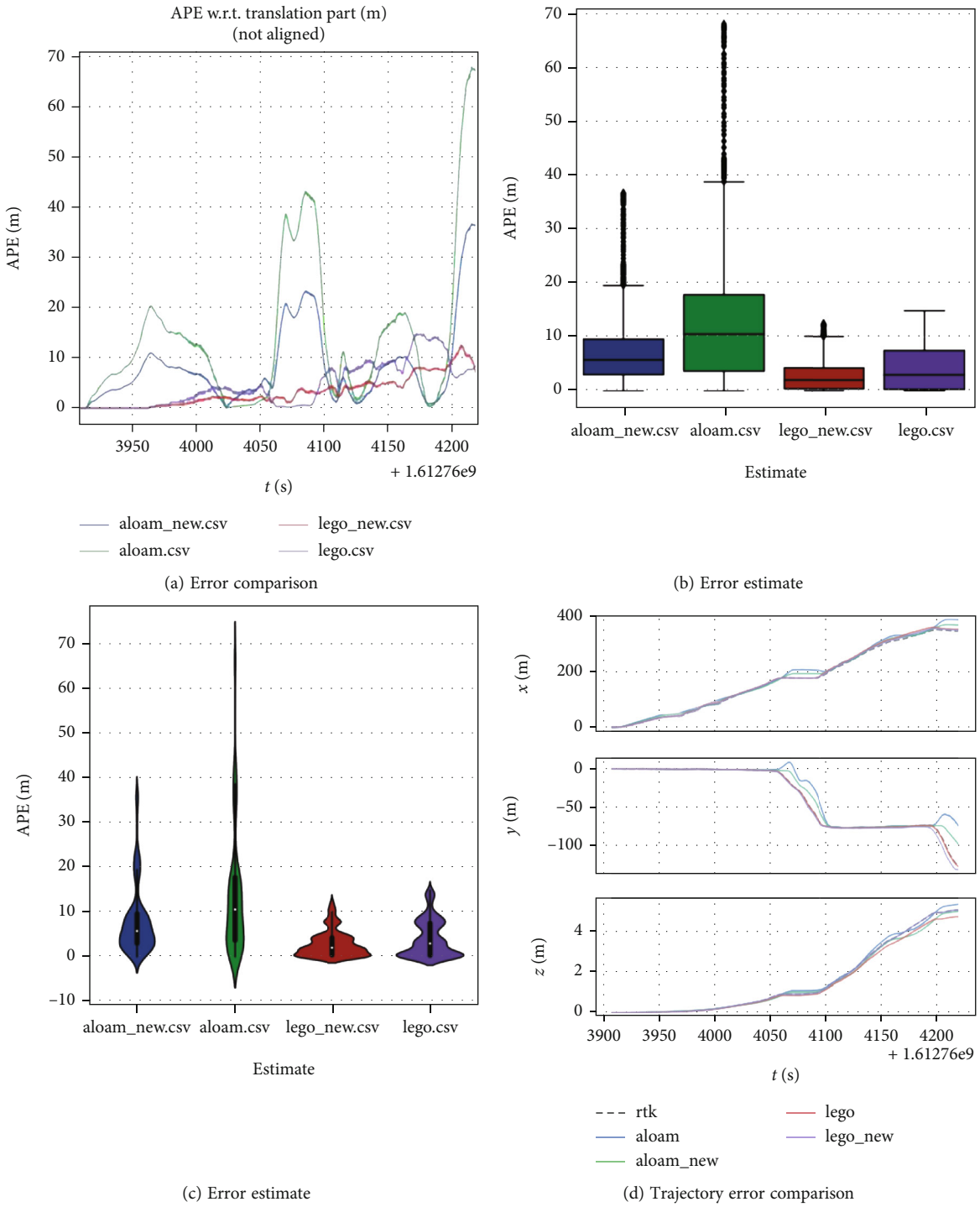
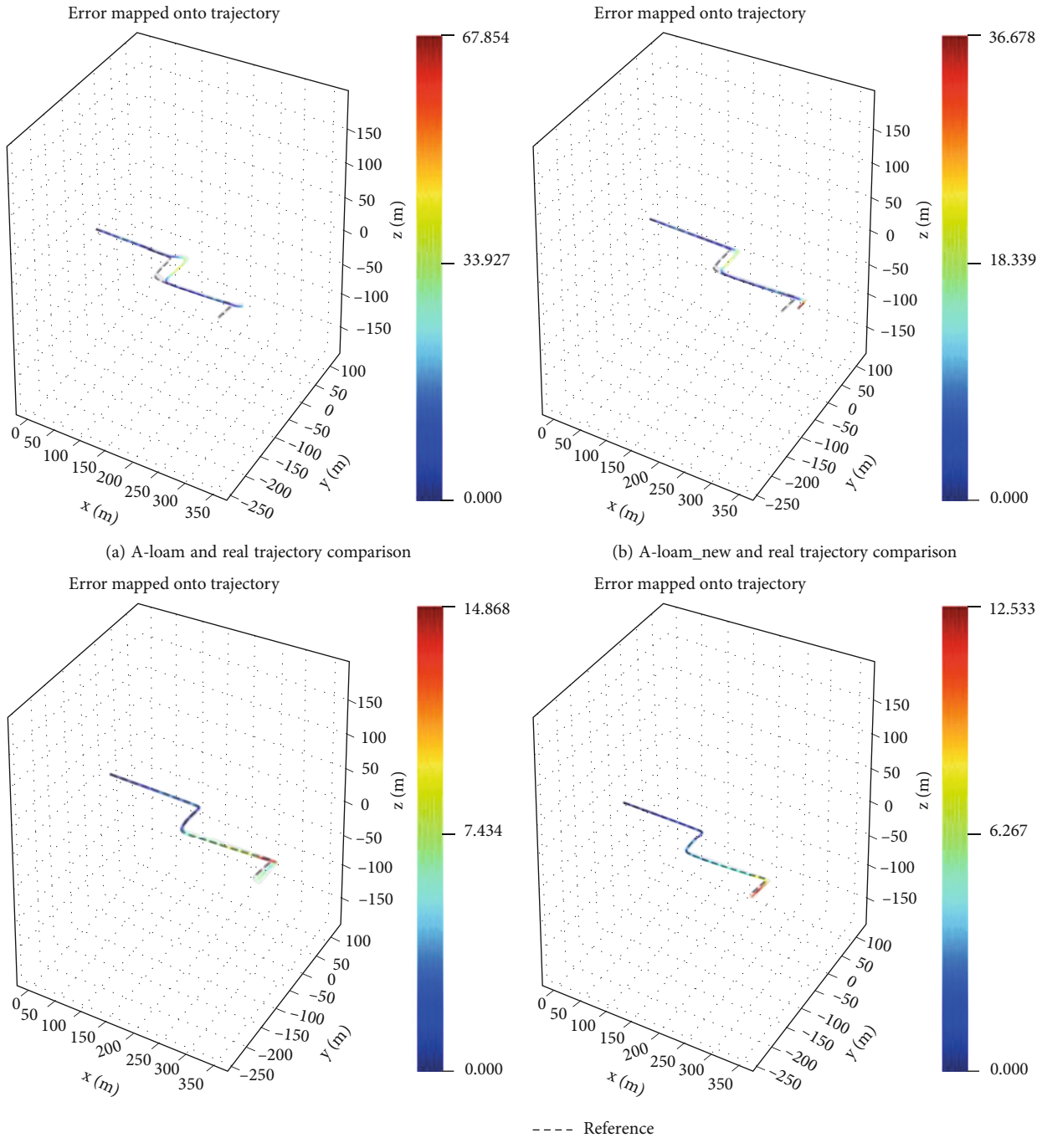


Figure 11: Comparison diagram of quantitative analysis of campus environmental experiment.

and the scanning time of the point convergence corresponding to the central point can be obtained from the LiDAR timestamp. Assuming that the starting time of the scanning of a point convergence is and the scanning time of a point in the middle is t_{k+i}, then the position calibration of the point is the same as the above LiDAR orthodontic principle. For the whole point convergence, it is also necessary to consider its orientation angle θ^{rot}, so it is necessary to calculate the normal vector of the central point neighborhood and find the rotation matrix that makes the normal vector direction consistent. The overall principle is shown in Figure 9.

The effect of compensating the moving objects in the scene by using the above methods is shown in Figure 10.

Since this method needs to solve the motion posture of each point cloud cluster after clustering in the scene, resulting in too much computing overhead, it is necessary to limit



(a) A-loam and real trajectory comparison

(b) A-loam_new and real trajectory comparison

(c) LEGO-loam and real trajectory comparison

(d) LEGO-loam_new and real trajectory comparison

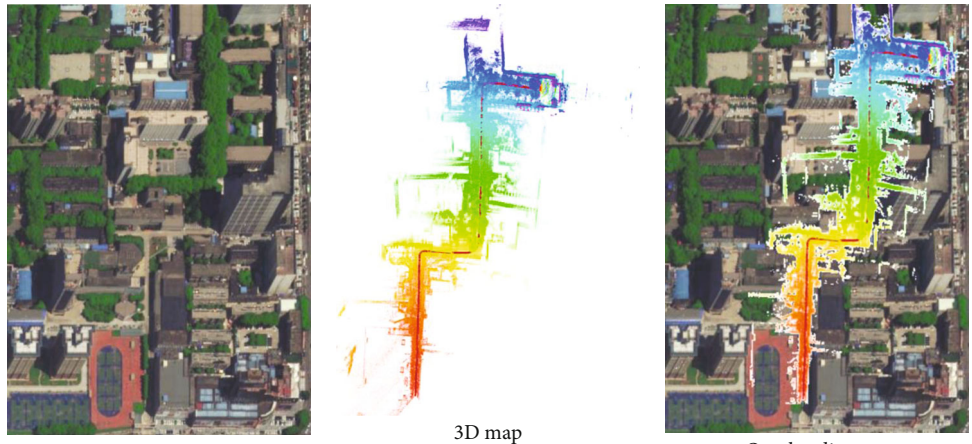
Figure 12: Comparison diagram of the trajectory effect.

the point cloud cluster for different scenes. Since an unmanned driving scene is taken as the benchmark in this paper, the point cloud cluster is set as

$$\begin{aligned} \left|x^{\max} - x^{\min}\right| \le 3 \wedge x^{\max} \le 20, \\ \left|y^{\max} - y^{\min}\right| \le 3 \wedge y^{\max} \le 20, \\ z^{\max} \le 2. \end{aligned} \tag{14}$$

4. Result

In this part, we first use three SLAM algorithms, A-LOAM, LEGO-LOAM, and T-LOAM, as the benchmark for experimental comparison. Then, we use the point cloud dedistortion method proposed in this paper, respectively, for the original versions of the above three algorithms and obtain their trajectories to quantitatively analyze the performance differences between the proposed method and the original



(a) Satellite imagery (b) 3D map (c) Overlay diagram

Figure 13: LEGO-LOAM_new map aligned with Google Earth.

Table 1: Comparison of ablation results.

Method	LiDAR compensation	Motion compensation	APE in cm					
			Max	Mean	Median	RMSE	SSE	Std
A-LOAM	—	—	67.8541	14.4364	10.4834	20.7422	611797	14.8919
	√	—	52.6475	11.2146	8.99951	16.1594	425757	12.2473
	√	√	36.6778	8.12781	5.76563	11.2797	180924	7.82116
LEGO-LOAM	—	—	14.8682	4.08091	2.9907	5.8075	47959.9	4.13198
	√	—	12.6464	3.25944	2.91616	4.45671	31614.5	3.25294
	√	√	12.5331	2.93791	2.85014	4.15562	24556.8	2.93902
T-LOAM	—	—	8.1165	2.1564	2.1168	3.9515	26546.4	2.16445
	√	—	6.5941	1.9594	2.2661	3.5941	21646.1	1.94865
	√	√	5.3911	1.9849	2.6489	3.1646	19816.7	1.89446

method. The experimental site was selected in the campus environment, the PIX unmanned vehicle carrying VLP-16 and RTK was used to obtain external information, and the algorithm was implemented in Ubuntu 18.04 (ROS) under AMD 3950X. All the algorithms are implemented in C++.

In the campus scene experiment, we avoided the driverless car driving through the sparse environment and ensured the presence of a certain number of moving objects (such as pedestrians and bicycles) in the scene. In other words, stable feature points can be obtained for interframe matching, while some moving objects exist, and then, distorted point clouds are generated.

We show the two benchmark algorithms and the absolute pose error (APE) after the improved point cloud compensation stage, respectively, as shown in Figure 11.

Because the reference algorithm lacks the correction of the distortion of moving objects in the scene, it can be found that the orthodontic algorithm proposed in this paper can effectively lower the APE of the reference algorithm and improve its positioning accuracy in any direction by comparing the figures. Meanwhile, we also compared the motion trajectory of the two benchmark algorithms and the improved algorithm reference RTK, as shown in Figure 12. It can be seen intuitively that the two benchmark algorithms can improve the coincidence degree with the real trajectory after using the orthodontic method proposed in this paper.

In addition, the method proposed in this paper can further improve the mapping effect. We used the improved LEGO-LOAM 3D map to overlay with the satellite map, as shown in Figure 13. It can be seen that after the improved motion compensation algorithm, more accurate 3D scene information can be obtained, and the constructed map and the actual map can be accurately matched.

Table 1 shows the ablation experiments of the proposed method. Since the motion compensation of LiDAR is loosely coupled with point cloud orthodontics of moving objects, we compare the performance of one module alone with that of the whole algorithm to illustrate the effectiveness of each module of the algorithm.

Table 1 shows the overall performance improvement effect of the two modules on the algorithm. Due to the large number of moving objects in the scene, the interframe matching is not accurate, so the performance of the original version of A-LOAM is poor. By using the point cloud compensation method proposed in this paper, the attitude of LiDAR itself can be estimated more reliably, and the errors

caused by point cloud distortion of moving objects can be calibrated to a certain extent, which makes the APE smaller, so as to obtain more accurate positioning effect.

5. Limitations

In this experiment, the feature points of the scene are used as the basis for interframe matching, and the movement posture is obtained by calculating the movement of the Euclidean clustering center point and the deviation of the normal vector, and then, the point cloud of the moving object is dedistorted. In practice, in a sparse scene, most of the feature points selected by the Soft-NMS-Select algorithm may be distributed in the point cloud cluster of moving objects. In this case, the experimental effect of this algorithm is worse than the benchmark, because point cloud matching highly depends on the selection of feature points within the scene. If most of the feature points are in motion, the odometer will be extremely inaccurate. In addition, due to the extra computational overhead brought by Euclidean clustering, the algorithm takes more time in most scenarios. However, not all objects involved in Euclidian clustering in the scene are in motion, so the algorithm has some invalid operations.

6. Conclusion

In this paper, we propose a LiDAR motion compensation method. We calculate the flatness and linearity of point cloud, introduce entropy value to screen the feature points with greater degree of information, and then optimize the odometer information of the point cloud by obtaining the prior information of its own motion through quadratic interpolation. Then, Euclidean clustering is performed for the current scanning scene, and cluster center points and normal vectors are calculated to represent moving objects. By eliminating distortion of the point cloud of moving objects, higher quality scanning results can be obtained, and positioning and mapping results can be further improved. We conducted localization and mapping experiments using VLP-16 in a real campus scene and evaluated our method with two benchmark LiDAR SLAM algorithms. The results show that the proposed method can effectively reduce the APE and improve the accuracy of pose estimation.

Data Availability

This article does not cover data research. No data were used to support this study.

Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

The authors would like to thank the National Natural Science Foundation of China (51974229), the Shaanxi Province Science and Technology Department (2021TD-27), and the National Key Research and Development Program, China (2018YFB1703402), for their support in this research.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[2] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[3] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[4] A. Censi, "An ICP variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, pp. 19–25, Pasadena, CA, USA, 2008.

[5] K. L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[6] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robotics: science and systems*, vol. 2, no. 4, p. 435, 2009.

[7] D. Kovalenko, M. Korobkin, and A. Minin, "Sensor aware lidar odometry," in *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–6, Prague, Czech Republic, 2019.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, and A. J. Davison, "Kinectfusion: real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, Basel, Switzerland, 2011.

[9] A. Nüchter, *Parallelization of Scan Matching for Robotic 3D Mapping*, EMCR, 2007.

[10] D. Qiu, S. May, and A. Nüchter, "GPU-accelerated nearest neighbor search for 3D registration," in *International conference on computer vision systems*, pp. 194–203, 2009.

[11] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, and J. Hornegger, "Real-time RGB-D mapping and 3-D modeling on the GPU using the random ball cover data structure," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1161–1167, Barcelona, Spain, 2011.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: speeded up robust features," in *European conference on computer vision*, pp. 404–417, 2006.

[13] H. Dong and T. D. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Field and Service Robotics*, pp. 327–342, Springer, 2014.

[14] S. Anderson and T. D. Barfoot, "RANSAC for motion-distorted 3D visual sensors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2093–2099, Tokyo, Japan, 2013.

[15] C. H. Tong and T. D. Barfoot, "Gaussian process Gauss-Newton for 3D laser-based visual odometry," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5204–5211, Karlsruhe, Germany, 2013.

[16] S. Anderson and T. D. Barfoot, "Towards relative continuous-time SLAM," in *2013 IEEE International Conference on Robotics and Automation*, pp. 1033–1040, Karlsruhe, Germany, 2013.

[17] R. Zlot and M. Bosse, "Efficient large-scale 3D mobile mapping and surface reconstruction of an underground mine," in *Field*

and Service Robotics, pp. 479–493, Springer, Berlin, Heidelberg, 2014.

[18] M. Bosse, R. Zlot, and P. Flick, "Zebedee: design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[19] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2D laser," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4312–4319, Kobe, Japan, 2009.

[20] T. Shan and B. Englot, "Lego-loam: lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, Madrid, Spain, 2018.

[21] P. Zhou, X. Guo, X. Pei, and C. Chen, "T-loam: truncated least squares lidar-only odometry and mapping in real time," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 99, pp. 1–13, 2021.

[22] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, and V. Kumar, "Sloam: semantic lidar odometry and mapping for forest inventory," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1–1, 2020.