De-Skewing LiDAR Scan for Refinement of Local Mapping

Lei He, Zhe Jin * and Zhenhai Gao

State Key Laboratory of Automotive Simulation and Control, Jilin University, Changchun 130022, China; jlu_belei@jlu.edu.cn (L.H.); gaozh@jlu.edu.cn (Z.G.)
* Correspondence: jinzhe18@mails.jlu.edu.cn

Received: 31 January 2020; Accepted: 25 March 2020; Published: 26 March 2020

Abstract: Simultaneous localization and mapping have become a basic requirement for most automatic moving robots. However, the LiDAR scan suffers from skewing caused by high-acceleration motion that reduces the precision in the latter mapping or classification process. In this study, we improve the quality of mapping results through a de-skewing LiDAR scan. By integrating high-sampling frequency IMU (inertial measurement unit) measurements and establishing a motion equation for time, we can get the pose of every point in this scan’s frame. Then, all points in this scan are corrected and transformed into the frame of the first point. We expand the scope of optimization range from the current scan to a local range of point clouds that not only considers the motion of LiDAR but also takes advantage of the neighboring LiDAR scans. Finally, we validate the performance of our algorithm in indoor and outdoor experiments to compare the mapping results before and after de-skewing. Experimental results show that our method smooths the scan skewing on each channel and improves the mapping accuracy.

Keywords: skewing; slam; 3D-LiDAR; point cloud; sensor fusion

1. Introduction

Various kinds of sensors are becoming increasingly important in a self-driving vehicle and multi-sensor fusion plays an important role in mapping or navigation [1]. They help vehicles or robots perceive the environment [2], recognize objects [3], and localize themselves in a global coordinate system [4]. The accuracy and precision of sensor data have a significant effect on the results of simultaneous localization and mapping (SLAM).

In the field of autonomous driving, LiDAR (light detection and ranging) provides a rich, 3D point cloud in real-time by using a laser and detector pair mounted in a compact housing. If there is not any special instruction, LiDAR in the following section refers specifically to the field of autonomous driving. Furthermore, the rotational speed of LiDAR ranges from 5 to 20 Hz and it can create 360° images with reflected points. As laser has strong power and can hardly be affected by light or other electromagnetic waves, LiDAR has been widely used in pose estimation or environmental reconstruction [5]. However, LiDAR also has weaknesses. Different from camera or IMU working at a high frequency, it suffers from sensing accuracy when vehicles or robots moving fast suddenly. In addition, the laser reflection can be influenced by snow or rain which may lead to distant mistakes. These factors often result in a pulsed or serrated point cloud and noise points. Thus, the mapping system may be struggling when processing point clouds, then leading to tough output eventually.

Various optimization approaches are applied on slam, like the graph-based approach [6], Ceres Solver [7], and GTSAM optimization toolbox [8]. By minimizing the accumulated errors, these methods simplify the computational complexity and avoid reprocessing the mass sensor data during execution. Although the optimization model having been improved increasingly, the errors existing in raw LiDAR
point cloud have not been valued greatly. The motion of platform carrying LiDAR could be complex in the real world. Because of the uneven road or radical driving, LiDAR could be shaken suddenly, leading to jagged scan lines in LiDAR measurements. This kind of error brought about by motion can hardly be erased in the mapping process and would even affect semantic segmentation, object detection, filtering dynamic objects and so on. Some of the other methods attempt to reduce this impact by linear interpolation, ICP core, and so on. However, because of the existence of gimbal deadlock, spherical interpolation cannot be achieved using Euler angles and some did not make use of additional sensor data which reduce the processing accuracy.

In this study, we present a point cloud de-skewing algorithm to reduce the sensing error caused by ego-motion when mapping the surrounding area. Our algorithm integrates the high-frequency IMU data between two LiDAR scans to calculate the IMU motion discretely and treats IMU poses as optimization variables, restricted by LiDAR odometry. Then, interpolation is implemented to generate poses corresponding to every point. After transferring point clouds from the LiDAR frame to the IMU frame, the ceres solver is used to optimize the IMU poses until the de-skewing process converges. Compared with optimization in a single LiDAR scan, our algorithm can effectively eliminate the skewing by utilizing adjacent point clouds. Also, quaternion interpolation would not cause the risk of unpredictable problems.

In Section 2, we summarize related works in the fusion of LiDAR and IMU measurements, point cloud registration and other de-skewing methods. In Section 3, we first illustrate LiDAR skewing brought by ego-motion and present an overview of our system. Then, we integrate IMU data and model interpolation, preparing for transformation. Before optimizing in ceres solver, LiDAR odometry is applied as a limitation in the following work. In Section 4, several experiments were performed to choose which ICP method and evaluate our de-skewing effect, including a discussion of them. Finally, we conclude our study in Section 5.

2. Related Works

There are primarily two kinds of approaches to deal with the fusion of LiDAR point cloud and IMU measurements. One of them is loosely coupled fusion that estimates the LiDAR and IMU respectively and then fuses the results. [9] uses a loosely coupled EKF (extended Kalman filter) to fuse IMU and 2D LiDAR but it cannot handle the complex environments. [10] presents a Multi-Sensor-Fusion Extended Kalman Filter to fuse IMU with other sensor data. It sacrifices the accuracy of the process results in exchange for computational efficiency as a result that it does not update the odometry part with IMU measurements. The other category, tightly coupled fusion, combines LiDAR and IMU measurements to estimate a common variable. [11] proposed a method to optimize the IMU measurements with the LiDAR odometry. [12] applies an error-state Kalman filter to tightly couple the IMU and LiDAR, updating corrections with the matched LiDAR heightmap and a prior DEM (digital elevation model). Although the tightly coupled method is more computationally complex than loosely coupled fusion, the former takes advantage of sensor data and gets better results.

3D point cloud registration is the front-end of the mapping pipeline and the registration consequences affect all the following mapping processes [13]. Point cloud registration is usually divided into two steps: coarse registration and fine registration. The first step is implemented to find out an approximate transformation when two point clouds are far from each other. LORAX divides the point cloud into many small pieces and generates descriptors using deep neural network [14]. 4-Points congruent sets [15] and its improved proposals [16–18] could register the point cloud quickly and robustly. The second one is to improve the accuracy after coarse registration, after having an approximate result. Fine registration is mainly processed with ICP (iterative closest point) and variants of ICP. The basic ICP pairs the corresponding points between two point clouds and evaluates the transformation to minimize RMS (root mean square) of all the point pairs. The algorithm needs an initial value and a suitable value could get better convergence. Point cloud segmentation and feature extraction can improve process efficiency and the quality of the registration results. However, this rudimentary method depends on a large amount of calculation and suffers from noise and outlier.
The LiDAR motion during its rotation period twists the point clouds since LiDAR rotates at a low frequency with 360° fields of view [19]. Once high acceleration motion occurs, the point clouds reflected by laser are disordered because of fast-moving. In [1], it preprocesses the LiDAR data to reduce distortion caused by the movement during the scan. It extracts roll, pitch and yaw from the rotation matrix between two scans. Then, the relative rotation of the point in the point cloud is approximated by linear interpolation and the rotation matrix, estimated from feature points, which is not accurate enough for de-skewing. [20] uses ICP core for de-skewing which does not make use of other sensor data. [21] only transfers the point clouds between two coordinate systems and has actually no de-skewing. [22] shows that skewing leads to trajectory drift when mapping the surrounding environment and exhibits visible rotation errors under the complex movement condition. It also attempts to de-skew with the assistance of high sampling IMU data but gives up eventually because of the challenging in calibration between LiDAR and IMU. [23] proposes a pre-processing module for de-skewing detailly. However, it interpolates Euler angles with linear interpolation which suffers from gimbal deadlock. [24] only conducted distortion correction at the odometry part which may cause a risk of cumulative error for later process.

3. Methods

3.1. The Impact of LiDAR Motion on Skewing

The frequency of LiDAR in the autonomous driving field is mostly under 20 Hz which is not very fast as opposed to cameras. Distance information measured by LiDAR is concerned with the flying time of the laser beam and firing channel. All these firing mechanisms, arranged vertically, rotates around the center of LiDAR and a scan period is the time of one rotation. The faster it rotates, the more information we get in a certain time but also shakes more heavily. In contrast, the slower, the more vulnerable to skewing. Considering the accuracy of measurement and the scan frequency of LiDAR, ranging from 5 Hz to 20 Hz, we set the frequency at 10 Hz.

As we can see from the left part of Figures 1 and 2, the LiDAR scan reflects the real wall correctly when LiDAR is stationary. Three scan points and the LiDAR itself constitute an isosceles triangle when LiDAR does not move or rotate. In order to find out the exact effects of LiDAR motion on scan skewing, we separate the motion into uniform linear motion and uniform circular motion for convenience and compare the point clouds with that under stationary condition. The middle part of two figures shows the LiDAR scan in two motion conditions and the right part of them shows the point clouds in two circumstances respectively. As we can see, linear motion mainly changes the relative rotation of the point in the point cloud is approximated by linear interpolation and the rotation matrix, estimated from feature points, which is not accurate enough for de-skewing. [20] uses ICP core for de-skewing which does not make use of other sensor data. [21] only transfers the point clouds between two coordinate systems and has actually no de-skewing. [22] shows that skewing leads to trajectory drift when mapping the surrounding environment and exhibits visible rotation errors under the complex movement condition. It also attempts to de-skew with the assistance of high sampling IMU data but gives up eventually because of the challenging in calibration between LiDAR and IMU. [23] proposes a pre-processing module for de-skewing detailly. However, it interpolates Euler angles with linear interpolation which suffers from gimbal deadlock. [24] only conducted distortion correction at the odometry part which may cause a risk of cumulative error for later process.

Figure 1. The effect of ego-motion on scan skewing.
3.2. Overview of De-Skewing System

Figure 3 describes the workflow of our de-skewing algorithm. We fuse both point cloud and IMU data for the de-skewing process. Every process period contains two LiDAR point clouds and a series of IMU data during the scan period, as shown in Figure 4. The last LiDAR scan $L_i$ in Figure 3, corresponds to the first point cloud in the processing period of Figure 4 and the current LiDAR scan $L_j$ matches with the second one in Figure 4. The de-skewing process performs as follows: (1) Update IMU state iteratively when new data comes; (2) after receiving the LiDAR scan $L_j$, integrate IMU data as $T$ in Figure 3 to be used for transformation (Section 3.3); (3) at the time when the other point cloud $L_i$ received, estimate the LiDAR odometry through registration (Section 3.4); (4) transform points from the LiDAR frame to the IMU frame and then, to the frame of the first point in the current point cloud as $L'_i$ in Figure 3 (Section 3.3); (5) setting the ceres solver to minimize the skewing problem with the assistance of a point cloud map (Section 3.5). The LiDAR odometry is utilized to reduce divergence results; (6) the current process is finished when reaching the number of iteration or the amount of change in the cost function is smaller than a threshold.

![Figure 2. Effect of ego-rotation on scan skewing.](image)

![Figure 3. De-skewing system framework.](image)

![Figure 4. Sensor data in a processing period.](image)
3.3. IMU Odometry and Coordinate Transformation

Measured data from IMU are in the form of the Euler angle and are transferred to quaternion when updating the IMU odometry. We denote the three-dimensional coordinates of IMU measurements as \( p_i \in \mathbb{R}^3 \), where \( i \) represents the time sequence of the measurement stream. Similarly, quaternion \( q_i \in \mathbb{R}^4 \), corresponding to the rotation matrix \( R_i \), are orientation and velocity parts of IMU data. Conveniently, we take quaternion form of rotation for IMU odometry and the conversion from Euler rotation to \( q_i \) and then to \( R_i \) is shown as Equations (1) and (2)

\[
q_i = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) - \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) + \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) - \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \end{bmatrix},
\]

(1)

\[
R_i = \begin{bmatrix} 1 & -2q_i^2 & 2q_iq_j & 2q_iq_k \\ 2q_iq_j & 1 - 2q_i^2 & 2q_iq_k - 2q_j^2 \\ 2q_iq_k & 2q_iq_j + 2q_k^2 & 1 - 2q_i^2 - 2q_j^2 \\ 2q_iq_k - 2q_jq_k & 2q_jq_i + 2q_k^2 & 2q_iq_j - 2q_k^2 \end{bmatrix},
\]

(2)

where \( \varphi, \theta, \psi \) are the rotation angles around \( x, y, z \) axes and \( q_w, q_x, q_y, q_z \) are the four components of the quaternion \( q_i \), corresponding to \( w, x, y, z \) in Equation (1).

Every time when new IMU data arrives, we update the IMU state by discrete evolution [8] as Equations (3)–(5),

\[
p_j = p_i + \sum_{k=i}^{j-1} \left[ v_k \Delta t + \frac{1}{2} g^W \Delta t^2 + \frac{1}{2} R_k(\omega_k - b_{\omega k}) \Delta t^2 \right],
\]

(3)

\[
v_j = v_i + g^W \Delta t^2 + \sum_{k=i}^{j-1} R_k(\omega_k - b_{\omega k}) \Delta t,
\]

(4)

\[
q_j = q_i \otimes \prod_{k=i}^{j-1} \delta q_k = q_i \otimes \prod_{k=i}^{j-1} \left[ \frac{1}{2} \Delta t (\omega_k - b_{\omega k}) \right],
\]

(5)

in which \( \Delta t \) is the time between two consecutive IMU data and \( g^W \) is the vector form of gravity in the world frame. \( \omega_k \) and \( \omega_k \) are linear acceleration and angular acceleration of raw IMU measurements. \( b_{\omega k} \) and \( b_{\omega k} \) are gyroscope bias and acceleration bias respectively.

All the IMU data are integrated during the point cloud scan period, from \( k = i \) to \( k = j \). Then, each IMU input corresponds with its relative pose in the LiDAR scan frame, which is optimized.

Although poses are integrated between every two consecutive IMU data, they do not correspond with points in the point cloud. In order to estimate the pose of point, interpolation is applied. The more interpolation variables are used, the more accurate the interpolation result is. However, this could result in a large amount of calculation and the interpolation results may become overfitting when the number of interpolation variables exceeds a certain threshold. As a result, the four closest poses, calculated by IMU data, are chosen for interpolation which balances the accuracy and efficiency, as shown in Figure 5. We represent location with a three-dimensional vector and rotation with quaternion. Each pose contains both location and rotation. Since quaternion has additional constraints, we interpolate location and rotation respectively.
Figure 5. The pose of every point.

### 3.3.1. Location Modeling

We use Lagrangian interpolation to establish location function on time. It is used to fit the IMU motion trajectory by discrete measurements. Similar to the two-dimensional coordinate system, we replace the variable with a three-dimensional pose vector as Equations (6) and (7)

\[
\begin{align*}
L_{\text{location}}(t) &= \sum_{j=0}^{n} p_j \ell_j(t), \\
\ell_j(t) &= \prod_{i=0, i \neq j}^{n} \frac{t-t_i}{t_j-t_i},
\end{align*}
\]

where \( t \) and \( p_j \) are time and three-dimensional vectors of IMU pose respectively and the number \( n \) is four as we choose the four closest IMU pose.

### 3.3.2. Rotation Modeling

Because of the existence of gimbal deadlock, spherical interpolation cannot be achieved in Euler angles. Quaternion interpolation is not the same with any linear interpolation. Linear interpolation can hardly separate rotation evenly, as shown in Figure 6. The chord length is divided equally into four but the arc length is not separated evenly.

We use unit quaternion to represent rotation in this study, which signifies a four-dimensional hypersphere. Glenn Davis proposes slerp [25] (spherical linear interpolation) to interpolate between two quaternions, as Equation (8)

\[
r = \text{slerp}(p_i, p_{i+1}, t) = \frac{\sin(1-t)\theta}{\sin\theta} p_i + \frac{\sin t\theta}{\sin\theta} p_{i+1},
\]

where \( p_i \) and \( p_{i+1} \) are known IMU quaternion updating iteratively, \( r \) is the quaternion, the point rotation in the current scan, to be solved, and \( t \) is the interpolation variable, which ranges from 0 to 1, as shown in Figure 7.

However, when \( r \) comes close to \( p_i \) or \( p_{i+1} \), slerp result is not accurate enough because we have not utilized other neighborhood rotation messages. To improve this, Dam and Erik B propose squad [26] (spherical and quadrangle) based on slerp,

\[
\text{squad}(p_i, p_{i+1}, s_i, s_{i+1}, t) = \text{slerp( slerp}(p_i, p_{i+1}, t), \\
\text{slerp}(s_i, s_{i+1}, t), 2t(1-t)),
\]

where auxiliary quaternion \( s_i \) is taken as the temporary controller,

\[
s_i = \exp\left( -\frac{\log(p_{i+1}p_i^{-1}) + \log(p_{i-1}p_i^{-1})}{4} \right) p_i
\]

and the interpolation variable \( t \), as (8), the same as slerp, shown in Figure 7,

\[
t = \frac{t_f - t_{p_i}}{t_{p_{i+1}} - t_{p_i}},
\]
$t_r$ is the timestamp of point in the current point cloud timeline, $t_{p_1}$, $t_{p_{i+1}}$ are timestamps of IMU pose, the closest between $t_r$, integrated from IMU data.

After estimating the quaternion of every point, we transfer them to the rotation matrix by Equations (1) and (2) since they are four-dimensional vector and cannot directly participate in the vector multiplication.

3.3.3. Coordinate Transformation

As a result that LiDAR and IMU both have their own frame when collecting data, which is not convenient for further process, it is necessary to transform the point cloud from LiDAR frame to IMU frame by Equation (12), as shown in Figure 8.

$$
\begin{align*}
    x_b &= R_b^o x_l + T_b^o \\
    \begin{bmatrix} x_b \\ 0 \\ 1 \end{bmatrix} &= \begin{bmatrix} R_b^o & T_b^o \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_l \\ 1 \end{bmatrix} \\
    \begin{bmatrix} x_b \\ 0 \\ 1 \end{bmatrix} &= T_{ca} \begin{bmatrix} x_l \\ 1 \end{bmatrix}
\end{align*}
$$

(12)

Figure 6. Linear orientation interpolation on arc length when interpolation over 1/4 intervals.

Figure 7. Spherical linear interpolation.

The subscripts of $x_b$ and $x_l$ ($x \in \mathbb{R}^3$) represent the point data in IMU and LiDAR frame respectively. $R$ is the rotation part of this transformation and the superscript and subscript denote the transform direction from the LiDAR frame to IMU.

Then, we transform all points of the current point cloud to the frame of the first point. Since location and rotation models are established already, we only have to determine the interpolate parameter $t$ which concerns the time of IMU poses and LiDAR points.

We estimate the time of LiDAR points first. [27] describes the time management in each LiDAR scan in detail. Briefly, one data point period consists of three steps, laser firing, laser recharging, and data packing. Since the timestamp of every scan is the time of the first data point in the package,
as shown in Figure 4, the time of each point in this scan $t_p$ could be calculated from the offset to the timestamp of the current scan as Equations (13) and (14)

$$t_p = t_s + t_{off},$$

$$t_{off} = 50(\mu s) \times (n_{channel} - 1) + 3(\mu s) \times (n_{data} - 1),$$

where $t_s$ and $t_{off}$ is the timestamp of the current scan and time offset relative to the first point. The time offset $t_{off}$ consists of measurement time and intervals between firing. Since all sixteen lasers being fired and recharged every 50 $\mu$s and the cycle time between firing being 3 $\mu$s, the measurement part and cycles part could be deduced from the index of each channel.

However, this $t_p$ is the relative time in this scan. For further calculation, whether transferring $t_p$ to ROS frame or transferring IMU time back to this relative frame works.

![Figure 8. $T_{ex}$ is the pose of inertial measurement unit (IMU) in the LiDAR frame, known from prior calibration.](image)

### 3.4. Point Cloud Registration and LiDAR Odometry

Before optimizing in ceres solver, a limit shall be set in case of divergence, as shown in Figure 9. ICP algorithm is used to evaluate the rigid body transformation between two sets of point clouds. They are overlapped after one of them transforms into another through the transformation relationship solved by ICP. Pairs of corresponding points are chosen iteratively to calculate their distance which is used to refine the transformation by minimizing the sum of them. Because of the different definition of sampling, matching, weighting, rejecting, and error metric, this algorithm has various variants and are widely improved.

![Figure 9. The error range in ceres solver.](image)

We use the PCL (point cloud library) in this study which is a large-scale, open project for 2D/3D image and point cloud processing. It contains numerous state-of-the-art algorithms and is convenient for adjustment parameters. Although the ICP algorithm has many variants, only some of them are available in PCL. Considering two point clouds are adjacent to two frames of LiDAR measurements, the initial matrix guess could be replaced with the pose integrated from IMU data.
3.5. Ceres Solver Optimization

For now, we have LiDAR odometry between two adjacent scans, point cloud to be de-skewed and discrete poses integrating from IMU measurements. However, poses interpolated from IMU may not be accurate enough for de-skewing since IMU data suffer from noise and bias. We try to optimize IMU poses through ceres solver to minimize the influence of measurement errors.

The optimization progress contains two tasks, one of which is to minimize the thickness of flat areas in a point cloud map and the plane function needs to be estimated first. Different from the idealized data set, the corresponding points of two consequent scans hardly exist, leading to the results of point cloud registration far from ground-truth. However, our approach only deals with points on a plane that are selected by the curvature and distance changes and have no requirement for strict correspondence. Every time when a new scan comes, it is transformed to a submap that only contains the point clouds near the LiDAR in a certain distance to reduce the computational complexity. Then all points in the submap are classified into different planes. In each plane, the plane function is fit by PCA and the IMU poses are adjusted to minimize the height difference in the plane area.

Plane fitting using PCA tries to find the normal vector \( \vec{n} \) and the center of plane \( q \),

\[
\begin{align*}
n^T (p_i - q) &= 0, \quad p_i \in \mathcal{L}_p, \\
A(q) &= [\cdots, p_i - q, \cdots].
\end{align*}
\]

where \( p_i \) is the point in the plane area \( \mathcal{L}_p \) and \( n^T n = 1 \). While in real data, (15) can hardly be satisfied and thus it is replaced by minimizing the cost function below,

\[
\begin{align*}
\text{cost}(n, q) &= \min_{n^T} \sum_{p_i \in \mathcal{L}_p} |n^T (p_i - q)|^2 \\
&= n^T A(q) A(q)^T n, \\
\end{align*}
\]

where \( A(q) \triangleq [\cdots, p_i - q, \cdots] \). The problem can be transferred as follows,

\[
\begin{align*}
n^* &= \arg\min_{n^T} B(q^*) n \\
\text{s.t. } n^T n &= 1,
\end{align*}
\]

After factorizing the \( B(q^*) \) into the singular value decomposition, \( B(q^*) = U \Sigma V^T \), the normal vector \( \vec{n} \) is the last column of \( U \) and we get the point-normal equation for a plane,

\[
\begin{align*}
n_x (x - q_x^*) + n_y (y - q_y^*) + n_z (z - q_z^*) &= 0, \\
A &\triangleq \begin{bmatrix} n_x & n_y & n_z \end{bmatrix},
\end{align*}
\]

where \( n_x, n_y, n_z \) are components of \( \vec{n} \) in three axes, similarly as \( q_x^*, q_y^*, q_z^* \). This can be simplified as,

\[
\begin{align*}
Ax + By + Cz + D &= 0, \\
A, B, C &\text{ are equal to } n_x, n_y, n_z, \quad D = -(n_x q_x^* + n_y q_y^* + n_z q_z^*),
\end{align*}
\]

and the residual for this part can be represented as,

\[
\begin{align*}
r_L &= \frac{1}{2} \sum_{p_i \in \mathcal{L}_p} ||Ap_x + Bp_y + Cp_z + D||^2 ,
\end{align*}
\]

The other part of optimization is to minimize the degree of dispersion of points on the same scan channel. As a result that the distance between each measured point and LiDAR is different, points of each channel are not part of a common plane but a tapered surface. Thus, we think of angular deviation as an optimization variable. However, this single parameter cannot reflect the degree of dispersion objectively since the laser path length of different points is various, as shown in Figure 10. Although point 2 has a bigger angle gap to the theoretical angle than point 1, point 1 is far away from
the theoretical line which is a severe error. The angular parameter is replaced by the distance to the tapered surface of the corresponding channel,

\[ r_{\Delta \theta} = \frac{1}{2} \sum_{p \in L_c} \left\| \arctan \left( \frac{p_2}{\sqrt{p_2^2 + p_2^2}} \right) - q_i \right\|^2 , \]  

(21)

\( l_p \) is the laser path length and \( q_i \) represents the theoretical angle on the channel \( i \). The channel information is unreachable in the usual PCL point cloud type and maybe is not published in some LiDAR driving program. Changing point type when programming or estimating the channel number through a vertical angle both can complete the missing information.

![Figure 10. Impact of angle and laser path length on the degree of dispersion.](image)

In order to avoid divergence result, we take trust region methods as the solving approach and reject the current optimization circle when the final IMU pose exceeds a certain range, defined by the odometry of LiDAR and IMU. The Levenberg-Marquardt algorithm is the most popular algorithm for solving non-linear least squares problems and was also the first trust-region algorithm to be developed. The LM method is an improvement based on the Gauss-Newton algorithm. By adjusting the parameters, the optimization can be freely switched between the gradient descent method and the Gauss-Newton method, and the fast convergence can be ensured while ensuring the drop. A common form of LM method is shown as,

\[ (J^T J + \lambda I)\delta = -J^T e, \]  

(22)

where \( J \) is the gradient of the raw function relating to the independent variable, \( I \) is the identity matrix, \( \delta \) is the amount of change in the independent variable, \( e \) is the error of raw function. The damping factor \( \lambda \) (non-negative) is adjusted at each iteration. A smaller value can be utilized to bring the algorithm closer to the Gauss-Newton algorithm or \( \lambda \) can be increased to give a step closer to the gradient-descent direction.

Because of the existence of measurement error and algorithm defect, the transformation evaluated from LiDAR odometry and IMU integration is different and we define the pose error between them in the first optimization circle as the error range, as the light blue area in Figure 9. In the following optimization, a result is discarded when the distance between transformation calculated from IMU poses and LiDAR odometry is larger than the error range.

### 4. Results and Discussion

In this section, we performed and analyzed several experiments. The first part, ICP registration comparison tried to find the best ICP method for indoor and outdoor experiments. Then, the latter two parts were conducted to evaluate our algorithm.

We received LiDAR point cloud and IMU measurements on the ROS (robot operating system) under the Linux system. The results of point clouds were presented in the PCL viewer, a third-party plugin. MATLAB was used to process data comparing with the ground-truth. Before the experiment, external calibration between LiDAR and IMU is evaluated. Similar to [11,28], we combined LiDAR
odometry and IMU motions to solve the external parameters first which were optimized in ceres solver later.

4.1. ICP Registration Comparison

We compared the different configurations of the ICP register to find out the best parameter for the following experimental data set. Codes on the PCL document were taken as the basic ICP method, which was the first, blue one on the legend in Figure 11. Then, we added the IMU pose as the initial guess of the ICP algorithm. Reciprocal correspondences were added to the third method and the final one combined both the initial guess and reciprocal correspondences. We tested both indoor and outdoor data set and these methods showed little difference in the outdoor part. Thus, we focus on the indoor part.

![Figure 11. Comparison of different ICP registrations.](image)

We ignored the first and last 10% data to avoid the effect of severe motion. The first part of Figure 11 showed the ICP fitness score of four separate ICP configurations. The smaller the score, the better the result showed. The third method gets an extremely higher score sometimes. When it came to extreme points, this method scored the highest in all of them mostly. But other parts were similar to each other and can hardly tell the difference between them. Then, we set the first method, the basic ICP algorithm, as a reference and drew the score error curves between the reference and the rest three methods respectively, as shown in the middle part of Figure 11. As we can see, the pink one scored much higher than those two with IMU input as an initial guess. Thus, the ICP method with reciprocal correspondences was not the best choice. However, as a result that the curve plotted first could cover the latter one, it can hardly tell which method was better for the rest two of them. Then, we change the plot sequence as shown in the final part of Figure 11. Combining the last two parts of Figure 11, we find that both two methods performed similarly most of the time. While at some extreme points, the curve in purple had a lower score than that in orange. Above all, reciprocal
correspondences did not improve the ICP performance a lot and the second method, ICP with IMU input, would become our registration program in the following experiments.

4.2. Indoor Experiments

In this section, we moved the pair of LiDAR and IMU around by hand in a room to construct the 3D information of the surrounding space, as shown in Figure 12. We held LiADR and IMU in the air first and kept them horizon, just like they were mounted on a vehicle. Then, we moved them in a circular route, the radius of which was about 20 cm. To model vibration in the real world, the LiDAR-IMU pair was rotated in various directions and artificial jitter was added occasionally to simulate sudden shaking when our vehicle went through a pit.

Figure 12. Scanned room in the indoor experiment.

Figure 13 showed the mapping results of our room before and after de-skewing. Because of our room being in a mess, we chose a relatively clear wall for comparison, as the red rectangle in the left part of Figure 13. Then, we enlarged this area and demonstrated two pictures, representing raw point cloud and de-skewed point cloud respectively. As we can see from two red arrows, the point cloud of a wall after de-skewing rowed more orderly and arranged flatter than that before de-skewing. The “thickness” of the wall was also reduced. However, because of our mapping algorithm’s precision, there were still some messy points around.

Figure 14 and Table 1 showed exactly the error of the indoor experiment. We calculated the average angle errors in one scan changing with time and the RMSE of each channel. As we can see in Figure 14, the curve of raw data changed rapidly sometimes because of unstable motion. The sawtooth wave of raw data in blue resulted from our regularly rotating the LiDAR-IMU pair at random speed. After de-skewing, the scan error curve became smoother without massive pulse shapes. Overall, the orang curve stabilized around 0.8° which was lower than that of raw data. The ground truth of each channel’s vertical angle could be found in [27]. Sixteen scans are equally distributed in the vertical area from $-15.0^\circ$ to $+15.0^\circ$. Table 1 showed the comparison of channel accuracy before and after de-skewing. RMSE in nearly all 16 channels showed a better performance after de-skewing. The data after de-skewing had an average 13.7% decrease. The gap between two sorts of numbers seemed smaller around 7 and 8 channels. However, it turned larger when it came to 0 and 15 channels.
3. Outdoor Experiments

The last two outdoor experiments were conducted on the school campus. The former one aimed to find out whether our approach would reduce the effect of the deceleration zone and the latter experiment was taken place on the uneven road.

Our experimental platform was an autonomous vehicle, as shown in Figure 15a. 16-line 3D LiDAR and IMU were mounted on the top of the vehicle. The rotation frequency of LiDAR had three different

---

**Table 1.** Evaluation of channel accuracy in the indoor experiment.

| Channel | RMSE (No De-Skewing) | RMSE (After De-Skewing) | Channel | RMSE (No De-Skewing) | RMSE (After De-Skewing) |
|---------|----------------------|-------------------------|---------|----------------------|-------------------------|
| 0       | 1.7895               | 1.5708                  | 8       | 0.1405               | 0.0996                  |
| 1       | 1.5165               | 1.3302                  | 9       | 0.3232               | 0.2780                  |
| 2       | 1.2540               | 1.1161                  | 10      | 0.5124               | 0.4482                  |
| 3       | 1.0288               | 0.9084                  | 11      | 0.7154               | 0.6290                  |
| 4       | 0.7912               | 0.6937                  | 12      | 0.9312               | 0.8226                  |
| 5       | 0.5679               | 0.4946                  | 13      | 1.3025               | 1.1915                  |
| 6       | 0.3453               | 0.2949                  | 14      | 1.5685               | 1.4473                  |
| 7       | 0.1411               | 0.1008                  | 15      | 2.5504               | 2.3487                  |

---
modes: 5 Hz, 10 Hz, and 20 Hz. A lower frequency achieved a higher angular resolution while a higher frequency could get more real-time information. Considering the accuracy and timeliness, our LiDAR rotation frequency was set to 10 Hz. Different from LiDAR, IMU updates measurements in a much higher frequency, up to 1000 Hz. However, a higher frequency might lead to uncertain errors of communication between IMU and a computer. After comparing communicating stability under various frequencies, we finally set our IMU frequency at 100 Hz. In order to reduce the relative movement, the LiDAR and IMU were placed closely and fixed together, as shown in Figure 15b. Also, a high precision RTK-GPS, mounted at the trunk, was used to provide the ground-truth of poses in the outdoor experiment. During the outdoor experiment, the estimated trajectories are aligned with the ground truth in [29].

The first experiment is performed in front of a building to show the performance of our method when the vehicle going through a deceleration zone, as shown in Figure 16. We drove the vehicle along the red route to rebuild the environment information. We started at the center of the crossroad and drove at a speed of 15 km/h under those trees. After passing through the deceleration zone, the yellow rectangle in Figure 16, we decelerated till our vehicle stopped.

The environment rebuilt by our algorithm is shown in Figure 17. For visual comparison, we choose an area with two parallel walls, as the red rectangle area shown in Figure 17. We focused on the point clouds in this area to evaluate the performance of our method because flat walls were the best, obvious area for comparison and these two walls were the nearest and clearest. Also, these two walls had different height. The above one in the red rectangle area was taller than the below one so that we could see both of them at the same time. To show the red rectangle area more clearly, we amplified this area and compared two point clouds before and after de-skewing, as shown in Figure 18. The left picture was the raw point cloud and the right one was after de-skewing. After comparison, the point cloud of
two walls on the right side of Figure 18 was thinner, denser than that on the left side. Also, two walls looked brighter after de-skewing which meant that points were gathered more closely.

![Figure 17](image1.png)

**Figure 17.** Rebuilt environment in which white points are laser reflection points. The red rectangle area is going to be compared later in Figure 18.

![Figure 18](image2.png)

**Figure 18.** Clouds represent the red comparison area in detail in Figure 17. Two walls were pointed out by red arrows. (a) Represents the raw point cloud; (b) is processed with our de-skewing approach.

We input a high accuracy GPS signal as the ground truth to evaluate the trajectory errors. Figure 19 showed the trajectory errors in three dimensions. Whether with or without de-skewing, pose errors of two kinds were highly related. Both two curves of x-error and y-error in the first half shared little difference. However, when it came to the second half, the gap between them became wider. The trajectory in x and y dimensions of de-skewed point clouds was closer to ground truth and showed less drift. Although z-error in both conditions has similar extremum, the de-skewed curve was more stable than the raw curve.

The results of the accuracy evaluation are shown in Tables 2 and 3. Pose evaluation showed that all six dimensions of the pose after de-skewing got a lower RMSE, compared with that without de-skewing, except roll dimension. Detailly, RMSE in x dimension decreased by about 41% and in y, z dimension, their RMSE both remained nearly half of them after de-skewing. Although the roll dimension had a little increase, three angular dimensions showed little difference, compared with the three locational dimensions. In channel evaluation, we added the maximum degree error as another evaluation index which represented the difference between the theoretical angle and the measured angle in each channel. The results in Table 3 showed that all 16 channels achieved better performance whether in RMSE or MAX after de-skewing. While in RMSE, channel 0 and 1 got worse consequences than channel 14 and 15. It was because, with the channel index increasing from 0 to 15, the corresponding vertical angle decreased from $+15.0^\circ$ to $-15.0^\circ$. As a result, most lasers in channel 0 and 1 were fired upwards which might be easily out of the effective range of our LiDAR scan. Thus, these over ranged points showed worse performance than those fired downwards.
We input a high accuracy GPS signal as the ground truth to evaluate the trajectory errors. Figure 19 showed the trajectory errors in three dimensions. Whether with or without de-skewing, pose errors of two kinds were highly related. Both two curves of x-error and y-error in the first half shared little difference. However, when it came to the second half, the gap between them became wider. The trajectory in x and y dimensions of de-skewed point clouds was closer to ground truth and showed less drift. Although z-error in both conditions has similar extremum, the de-skewed curve was more stable than the raw curve.

The results of the accuracy evaluation are shown in Tables 2 and 3. Pose evaluation showed that all six dimensions of the pose after de-skewing got a lower RMSE, compared with that without de-skewing, except roll dimension. Detailly, RMSE in x dimension decreased by about 41% and in y, z dimension, their RMSE both remained nearly half of them after de-skewing. Although the roll dimension had a little increase, three angular dimensions showed little difference, compared with the three locational dimensions. In channel evaluation, we added the maximum degree error as another evaluation index which represented the difference between the theoretical angle and the measured angle in each channel. The results in Table 3 showed that all 16 channels achieved better performance whether in RMSE or MAX after de-skewing. While in RMSE, channel 0 and 1 got worse consequences than channel 14 and 15. It was because, with the channel index increasing from 0 to 15, the corresponding vertical angle decreased from $+15.0^\circ$ to $-15.0^\circ$. As a result, most lasers in channel 0 and 1 were fired upwards which might be easily out of the effective range of our LiDAR scan. Thus, these over ranged points showed worse performance than those fired downwards.

The second part of the outdoor experiment was performed in a larger scan area. Because of the long-term use, this section of the road was quite uneven and had many cracks or even some gravels and pits on the surface. We drove along the pink route in a counterclockwise direction at a speed of 25 km/h. Similarly, we calculated the channel accuracy in this part. A significant improvement was seen after de-skewing in both RMSE and MAX, as shown in Table 4. RMSE showed an average of ten times decrease after de-skewing. Because of the tough road conditions, data of RMSE and MAX before de-skewing were

---

**Figure 19.** Errors in the outdoor test.

**Table 2.** Pose accuracy in the outdoor experiment.

| Pose | RMSE (No De-Skewing) | RMSE (After De-Skewing) |
|------|----------------------|------------------------|
| X    | 1.5750               | 0.9257                 |
| Y    | 1.9159               | 0.9645                 |
| Z    | 0.1436               | 0.0741                 |
| Roll | 0.0096               | 0.0103                 |
| Pitch| 0.1010               | 0.0895                 |
| Yaw  | 0.0935               | 0.0792                 |

**Table 3.** Channel accuracy in the outdoor experiment.

| Channel | RMSE | MAX (deg) |
|---------|------|-----------|
|         | No De-Skewing | After De-Skewing | No De-Skewing | After De-Skewing |
| 0       | 0.5201       | 0.3499      | 2.8023       | 0.358           |
| 1       | 0.1748       | 0.1591      | 0.3624       | 0.1643          |
| 2       | 0.1089       | 0.0959      | 0.2811       | 0.0984          |
| 3       | 0.0745       | 0.0658      | 0.1996       | 0.0688          |
| 4       | 0.0448       | 0.0374      | 0.1374       | 0.0395          |
| 5       | 0.0446       | 0.0398      | 0.1092       | 0.0441          |
| 6       | 0.0484       | 0.0458      | 0.089        | 0.0479          |
| 7       | 0.0086       | 0.0073      | 0.0274       | 0.0084          |
| 8       | 0.0099       | 0.0104      | 0.0115       | 0.011           |
| 9       | 0.0174       | 0.0130      | 0.0661       | 0.0145          |
| 10      | 0.0372       | 0.0308      | 0.122        | 0.037           |
| 11      | 0.0587       | 0.0500      | 0.1747       | 0.0573          |
| 12      | 0.0827       | 0.0720      | 0.2288       | 0.0840          |
| 13      | 0.1272       | 0.1146      | 0.2961       | 0.1315          |
| 14      | 0.1322       | 0.1166      | 0.3241       | 0.1391          |
| 15      | 0.1415       | 0.1233      | 0.3637       | 0.1451          |
much larger than that of the former part in the outdoor experiment, while our method still achieved a similar error level. The mapping result is presented in Figure 20. We chose two large wall areas for evaluation: L and R. We took a look at L first. Before de-skewing, some shadow points of the wall were highlighted by two red arrows in L1. When one wheel of our vehicle went into a pit, the sudden vibration transferred to LiDAR, leading to these lines. While after de-skewing, L2 showed few lines in the same place. Then, the R area was another building. As shown in R1, three red arrows point out disorganized points behind the wall caused by many cracks in front of this building. After de-skewing, this wall became more clear.

Table 4. Channel accuracy in the outdoor experiment.

| Channel | RMSE No De-Skewing | After De-Skewing | MAX (deg) No De-Skewing | After De-Skewing |
|---------|--------------------|------------------|-------------------------|------------------|
| 0       | 1.3610             | 0.1255           | 6                       | 0.1299           |
| 1       | 1.1805             | 0.1088           | 5.2                     | 0.1112           |
| 2       | 0.9990             | 0.0921           | 4.4                     | 0.0948           |
| 3       | 0.8176             | 0.0753           | 3.6                     | 0.0775           |
| 4       | 0.6377             | 0.0586           | 2.8                     | 0.0643           |
| 5       | 0.4590             | 0.0419           | 2                       | 0.0513           |
| 6       | 0.2838             | 0.0255           | 1.2                     | 0.0299           |
| 7       | 0.1020             | 0.0093           | 0.4324                  | 0.0134           |
| 8       | 0.1413             | 0.0112           | 0.8889                  | 0.0169           |
| 9       | 0.4551             | 0.0348           | 2.1818                  | 0.048            |
| 10      | 0.5982             | 0.0516           | 3.2                     | 0.0667           |
| 11      | 0.8019             | 0.0718           | 2.9474                  | 0.0913           |
| 12      | 1.0547             | 0.0940           | 4.1143                  | 0.1137           |
| 13      | 1.2791             | 0.1179           | 5.5                     | 0.147            |
| 14      | 1.5278             | 0.1449           | 6.1176                  | 0.2023           |
| 15      | 1.8948             | 0.1764           | 7.5                     | 0.2572           |

Figure 20. Outdoor experiment on a larger scale. The pink curve was the driving route. Different colors represent different reflection intensities of points. M: the scanned area, with two comparison areas L and R in red rectangles; L1: enlarged view of L without de-skewing; L2: enlarged view of L with de-skewing; R1: enlarged view of R without de-skewing; R2: enlarged view of R with de-skewing.
4.4. Discussion

The main contribution of this study is to develop a de-skewing system by fusing de-skewing into the mapping process and estimating the pose of points with quaternion interpolation. Before the formal experiment, ICP configuration was first confirmed to adapt to different experimental environments. We conclude that with IMU input as an initial guess, ICP achieved the best performance. Both indoor and outdoor experiments showed that the mapping errors affected by skewing were greatly reduced. In the indoor experiment, point clouds became more ordered and the average angle errors were more stable and less after de-skewing. Also, in the outdoor experiment, the first part demonstrated clearer point clouds of walls after de-skewing. Two trajectory errors, x and y, showed a high relationship when changing with time. However, after going through the deceleration zone, at about 30 s, the curve with de-skewing showed less error. In addition, the trajectory error in the z dimension changed a lot whose maximum was over 0.2 m, while our vehicle was driving on the flat road. That would result from our unsatisfactory mapping system, showing weakness in estimating the z dimension. In the second part, flat walls became clearer and showed a little shadow of walls caused by skewing, compared with those without de-skewing. Although maximum channel errors were much larger than those in the former part, they finally reduced to a similar degree after de-skewing.

5. Conclusion

This study presents a LiDAR de-skewing method which takes advantage of IMU measurements to de-skew LiDAR point clouds. We integrate the high-frequency IMU data between two LiDAR scans to calculate the IMU motion discretely and optimize both the point cloud and IMU data in ceres solver. Our method mainly made two contributions: (1) After updating the IMU state by discrete evolution, we apply Lagrangian interpolation and spherical and quadrangle to modeling the pose function in a certain period. Thus, we avoid the problem of gimbal deadlock and achieve better efficiency. (2) In the optimization process, the scan to be optimized is de-skewed in a submap rather than in a single point cloud. The extra information in the submap helps improve the accuracy of de-skewing results. All potentially available point clouds have maximized their effectiveness.

Our method was evaluated in various experiments. First of all, the best parameter of the ICP method was selected by conducting an extra experiment and we found that reciprocal correspondences did not improve the ICP performance a lot. However, the second method, ICP with IMU input, performed the best among them. Afterward, both the results of indoor and outdoor experiments show that our algorithm improves the quality of point clouds and smooths the trajectory errors at the time of mapping. Skewing point clouds resulting from a sudden rapid movement are greatly corrected after optimization, especially those highlighted walls. Compared with original data, RMSE and MAX in each channel decreased by 13.7% and 71.0% respectively. Also, RMES in trajectory evaluation dropped over 30%.

Further work on our study can focus on limiting the processing time to improve efficiency. As a result that one LiDAR scan contains lots of points, the optimization period takes considerable computational time to adjust the transformation of each point. This reflects the low efficiency of our algorithm in a certain respect. Furthermore, in the first part of our outdoor experiment, trajectory error in the z dimension performs badly because of our immature mapping method. Thus, the accuracy of our mapping process can be improved greatly.

Author Contributions: Conceptualization, L.H. and Z.J.; methodology, Z.J.; writing—original draft preparation, Z.J.; writing—review and editing, L.H. and Z.G. All authors have read and agree to the published version of the manuscript.

Funding: This research was funded by the complex road environment system perception and target tracking technology, grant number 2017YFB0102601.

Conflicts of Interest: The authors declare no conflict of interest.
References

1. Gao, H.; Cheng, B.; Wang, J.; Li, K.; Zhao, J.; Li, D. Object classification using CNN-based fusion of vision and LiDAR in autonomous vehicle environment. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4224–4231. [CrossRef]

2. Xiao, W.; Vallet, B.; Paparoditis, N. Change detection in 3d point clouds acquired by a mobile mapping system. *ISPRS Annals of Photogrammetry. Remote Sens. Spat. Inf. Sci.* **2013**, *1*, 331–336.

3. Ali, W.; Abdelkarim, S.; Zahran, M.H.; Zidan, M.; Sallab, A.E. YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.

4. Dubé, R.; Cramariuc, A.; Dugas, D.; Nieto, J.; Siegwart, R.; Cadena, C. SegMap: 3D segment mapping using data-driven descriptors. *arXiv* 2018, arXiv:1804.09557.

5. Cole, D.M.; Newman, P.M. Using laser range data for 3D SLAM in outdoor environments. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006.

6. Grisetti, G.; Kümmerle, R.; Stachniss, C.; Frese, U.; Hertzberg, C. Hierarchical optimization on manifolds for online 2D and 3D mapping. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; IEEE: Piscataway, NJ, USA; pp. 273–278.

7. Agarwal, S.; Mierle, K. Ceres Solver. Available online: http://ceres-solver.org (accessed on 22 August 2018).

8. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Trans. Robot.* **2016**, *33*, 1–21. [CrossRef]

9. Tang, J.; Chen, Y.; Niu, X.; Wang, L.; Chen, L.; Liu, J.; Shi, C.; Hyypää, J. LiDAR scan matching aided inertial navigation system in GNSS-denied environments. *Sensors* **2015**, *15*, 16710–16728. [CrossRef] [PubMed]

10. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to mav navigation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; IEEE: Piscataway, NJ, USA; pp. 3923–3929.

11. Soloviev, A.; Bates, D.; Van Graas, F. Tight coupling of laser scanner and inertial measurements for a fully autonomous relative navigation solution. *Navigation* **2007**, *54*, 189–205. [CrossRef]

12. Hemann, G.; Singh, S.; Kaess, M. Long-range GPS-denied aerial inertial navigation with LiDAR localization. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; IEEE: Piscataway, NJ, USA; pp. 1659–1666.

13. Pomerleau, F.; Colas, F.; Siegwart, R. A review of point cloud registration algorithms for mobile robotics. *Found. Trends® Robot.* **2015**, *4*, 1–104. [CrossRef]

14. Elbaz, G.; Avraham, T.; Fischer, A. 3D point cloud registration for localization using a deep neural network auto-encoder. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4631–4640.

15. Aiger, D.; Mitra, N.J.; Cohen-Or, D. 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph. (TOG)* **2008**, *27*, 85. [CrossRef]

16. Mellado, N.; Aiger, D.; Mitra, N.J. Super 4pcs fast global pointcloud registration via smart indexing. *Comput. Graph. Forum* **2014**, *33*, 205–215. [CrossRef]

17. Theiler, P.W.; Wegner, J.D.; Schindler, K. Keypoint-based 4-Points Congruent Sets—Automated marker-less registration of laser scans. *ISPRS J. Photogramm. Remote Sens.* **2014**, *96*, 149–163. [CrossRef]

18. Mohamad, M.; Ahmed, M.T.; Rappaport, D.; Greenspan, M. Super generalized 4pcs for 3d registration. In Proceedings of the 2015 International Conference on 3D Vision, Lyon, France, 19–22 October 2015; IEEE: Piscataway, NJ, USA; pp. 598–606.

19. Vlaminck, M.; Luong, H.Q.; Goeman, W.; Veelaert, P.; Philips, W. Towards online mobile mapping using inhomogeneous LiDAR data. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 19–22 June 2016; IEEE: Piscataway, NJ, USA; pp. 845–850.

20. Moosmann, F.; Stiller, C. Velodyne slam. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (iv), Baden-Baden, Germany, 5–9 June 2011; IEEE: Piscataway, NJ, USA; pp. 393–398.

21. Kim, C.; Jo, K.; Cho, S.; Sunwoo, M. Optimal smoothing based mapping process of road surface marking in urban canyon environment. In Proceedings of the 2017 14th Workshop on Positioning, Navigation and Communications (WPNC), Bremen, Germany, 25–26 October 2017; IEEE: Piscataway, NJ, USA; pp. 1–6.
22. Al-Nuaimi, A.; Lopes, W.; Zeller, P.; Garcea, A.; Lopes, C.; Steinbach, E. Analyzing LiDAR scan skewing and its impact on scan matching. In Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Madrid, Spain, 4–7 October 2016; IEEE: Piscataway, NJ, USA; pp. 1–8.

23. Zhao, S.; Fang, Z.; Li, H.; Scherer, S. A Robust Laser-Inertial Odometry and Mapping Method for Large-Scale Highway Environments. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019; pp. 1285–1292. [CrossRef]

24. Droeschel, D.; Behnke, S. LiDAR-based online mapping. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 20–25 May 2018; IEEE: Piscataway, NJ, USA; pp. 1–9.

25. Shoemake, K. Animating rotation with quaternion curves. *ACM SIGGRAPH Comput. Graph.* 1985, 19, 245–254. [CrossRef]

26. Dam, E.B.; Koch, M.; Lillholm, M. *Quaternions, Interpolation and Animation*; Datalogisk Institut, Københavns Universitet: Copenhagen, Denmark, 1998.

27. Robosense. RS-LiDAR-16 UserGuide. Available online: https://www.generationrobots.com/media/RS-LiDAR-16%20datasheet%20Car%20Mounted%28English%29.pdf (accessed on 26 March 2020).

28. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* 2018, 34, 1004–1020. [CrossRef]

29. Umeyama, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* 1991, 4, 376–380. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).