sensors

MDPI

Article

Method for Estimating Road Impulse Ahead of Vehicles in Urban Environment with Microelectromechanical System Three-Dimensional Sensor

Shijie Zhao, Minghao Wang, Pengyu Wang *, Yang Wang and Konghui Guo

State Key Laboratory of Automotive Simulation and Control, Jilin University, Changchun 130025, China; zhaosj17@mails.jlu.edu.cn (S.Z.); wangmh20@mails.jlu.edu.cn (M.W.); yangwang@jlu.edu.cn (Y.W.); guokh@jlu.edu.cn (K.G.)
* Correspondence: wangpy@jlu.edu.cn

Abstract: Most automated vehicles (AVs) are equipped with abundant sensors, which enable AVs to improve ride comfort by sensing road elevation, such as speed bumps. This paper proposes a method for estimating the road impulse features ahead of vehicles in urban environments with microelectromechanical system (MEMS) light detection and ranging (LiDAR). The proposed method deploys a real-time estimation of the vehicle pose to solve the problem of sparse sampling of the LiDAR. Considering the LiDAR error model, the proposed method builds the grid height measurement model by maximum likelihood estimation. Moreover, it incorporates height measurements with the LiDAR error model by the Kalman filter and introduces motion uncertainty to form an elevation weight method by confidence eclipse. In addition, a gate strategy based on the Mahalanobis distance is integrated to handle the sharp changes in elevation. The proposed method is tested in the urban environment. The results demonstrate the effectiveness of our method.

Keywords: road impulse features; LiDAR error model; pose estimation; MEMS LiDAR

Citation: Zhao, S.; Wang, M.; Wang, P.; Wang, Y.; Guo, K. Method for Estimating Road Impulse Ahead of Vehicles in Urban Environment with Microelectromechanical System Three-Dimensional Sensor. *Sensors* **2024**, *24*, 1192. https://doi.org/10.3390/s24041192

Academic Editor: Jong-Jae Lee

Received: 27 October 2023
Revised: 25 January 2024
Accepted: 1 February 2024
Published: 11 February 2024

1. Introduction

Safety, reliability, comfort, and economy have become the key factors for the development of high-quality automated vehicles (AVs). The main source of vertical input for an AV is road unevenness [1]. The unevenness of the road surface directly affects the ride comfort, smoothness, and operation stability. Due to the diversity of functional requirements and the complexity of the driving environment, AVs often have many heterogeneous sensors [2], such as light detection and ranging (LiDAR), camera, inertial measurement unit, etc. The road impulse features, such as speed bumps in front of the vehicle, can be sensed by relying on these powerful sensors. The semi-active or active suspension of the vehicle can be adjusted in advance before the vehicle reaches the target area. This fundamentally solves the problem of time delay in traditional control methods and improves the ride comfort of the car.

AVs require knowledge of the surroundings to safely and efficiently interact with the environment. The LiDAR is one of the most popular sensors in the field of autonomous driving due to its long range and high accuracy in 3D measurement. Existing approaches were mainly developed for mechanical LiDAR sensors, which collect the surrounding information by spinning a high-frequency laser array [3]. However, due to its high cost and weight, the mechanical LiDAR is difficult to implement on AVs as a mass production solution. In the last few years, the solid-state LiDAR has gained more interest due to its cost-effectiveness and lightweight. The microelectromechanical system (MEMS) solid-state LiDAR is a system that is built entirely on a silicon chip with no moving parts involved [4]. Therefore, it has advantages in size and weight compared to the mechanical LiDAR. Moreover, the MEMS solid-state LiDAR is resistant to vibrations by removing

the rotating mechanical structure, which enhances its reliability and durability [3]. To illustrate the difference between the two LiDARs, we used Velodyne HDL-64E and RoboSense RS-LiDAR-M1, for example. The specifications can be found in Table 1.

Table 1. The difference between mechanical and MEMS solid-state LiDAR.

	Type	Frequency	FoV	Horizontal Resolution	Vertical Resolution	Detection Range	Accuracy	Weight
Velodyne HDL-64E	Mechanical	10 Hz	360° × 26.8°	0.08–0.35°	0.4°	0.5–100 m	2 cm	15.1 kg
RoboSense RS-LiDAR-M1	MEMS Solid-state	10 Hz	120° × 25°	0.2°	0.2°	0.7–200 m	2 cm	0.8 kg

Perceiving the road impulse information is a vital capacity for semi-active or active suspension preview control. This task is usually accomplished by building a map of the road using acquired sensor data. Several mapping methods have been proposed to build a dense map based on trajectory estimation, such as an occupancy grid map and elevation map [5]. An elevation map replaces the binary information in the occupied grid map with elevation so that it is widely applied in outdoor environments. Generally, a global elevation map is built offline when the trajectory is estimated by simultaneous localization and mapping (SLAM). However, in scenarios where no prior map is available, the trajectory is estimated and corrected by SLAM online [6]. It means that we have to not only save all sensor data but also overcome the sensitivity to environmental conditions. For suspension preview control, we focused on the mapping accuracy of observed regions in front of the AVs rather than global mapping accuracy. Moreover, this mapping method can better achieve real-time map updates. Zhao et al. [7] and Wang et al. [8] deployed the local mapping framework to extract the preview elevation of roads based on LiDAR, IMU, and GPS, which achieved real-time and accurate estimation to some extent. Although the aforementioned methods show a good performance, the relatively low vehicle speed allows sufficient time for estimation. Therefore, we constructed a real-time local road elevation estimation. The main idea is to implement a local map through grid height based on the sensor error model and motion uncertainty. The proposed method leverages lightweight pose estimation and grid height models to reduce computational costs while considering uncertainty in the map update to improve accuracy. Our contributions are summarized as follows: (1) a real-time estimation algorithm of the vehicle pose by a MEMS LiDAR with a small FOV; (2) a grid height model based on the LiDAR error model to aggregate multiple points within a grid; (3) by deriving error propagation to align the maps of consecutive frames, lightweight local map updates are achieved; and (4) a gate strategy based on the Mahalanobis distance to deal with the sharp changes in elevation.

This paper is organized as follows: Section 2 reviews the related works on existing LiDAR elevation estimation approaches. Section 3 describes the details of the proposed real-time local road elevation estimation method. Section 4 shows experimental results and comparisons with existing works, followed by a conclusion in Section 5.

2. Related Work

A vehicle's surroundings can be geometrically modeled by constructing representations of the underlying terrain surface using range sensor data. These range sensors produce sparse point cloud representations, which must then be converted into continuous or piecewise structures to use [9]. Many methods have been proposed to describe the terrain under the vehicle. The occupancy grid map [10] proposed by Elfes is the most common one. The occupancy grid map describes the environment as some regular grids of a specific size. And the value of each grid represents the probability that the grid is occupied. The above concept of a 2D occupancy grid map is intuitively extended to a 3D occupancy grid map. Kudriashov et al. [11] proposed a method for constructing a 3D occupancy grid map of unknown terrain by LiDAR. The construction of the 3D map and the pose estimation of the system are carried out simultaneously using the extended Kalman filter and other

probabilistic methods. Three-dimensional occupancy grid maps provide abundant information and less ambiguity [12]. Another idea is to store information about 3D voxels in the octree, such as OctoMap [13]. The voxel size is a design parameter that influences memory usage, computation time, and, more importantly, the accuracy of the map. However, the voxels are distributed discretely, and we lose information about details inside each voxel. The space inside each cell is better represented in the normal distributions transform occupancy map [14]. And it was successfully applied on a large scale [15] and for highly dynamic environments [14]. Another 3D geometric description of the terrain is signed distance fields (SDF) or truncated signed distance Fields (TSDF). These methods map the distance to the nearest occupied cell rather than storing the entire volumetric representation of the environment [16]. Canelhas et al. utilize SDF to represent the alignment error and estimate the motion of the camera [17]. Sommer et al. [18] proposed gradient-SDF to describe 3D geometry that combines the advantages of implicit and explicit representations. Grinvald et al. [19] extended the TSDF model using information about object categories. Lang et al. [20] proposed a monocular SLAM system with direct TSDF mapping based on a sparse voxelized recurrent network. The direct TSDF mapping achieves simultaneous estimation of pose and map using features.

However, the 3D occupancy grid map will cause extremely high computational costs. It fails to strike a balance between resolution and computation time, which makes the method unavailable in real-time scenarios. As a compromise, the 2.5D occupancy grid map method has been proposed in recent years. Each of the grid values is redefined as the height of the grid. Fankhauser et al. [21,22] proposed a 2.5D grid map-building method considering the uncertainty of robot motion. Souza et al. [23] described the occupancy-elevation grid mapping method, where each cell represents the probability, height, and variance of occupancy. Zhou et al. [24] employed the preemptive RANSAC algorithm to extract planes from the terrain height information within the voxel grid. It enables the estimation of parameters such as height and depth in structured environments. Although the above methods achieved real-time mapping in some applications, highly accurate estimation is essential to accomplish reliable suspension preview control. Several works have introduced neural networks [25–30] to generate dense elevation maps. The neural networks effectively eliminate the noises and generate rational features in the occluded regions, which enhances the robustness and accuracy of methods in different environments. But, it is hard to implement real-time suspension preview control.

3. Method

The framework of our proposed method is illustrated in Figure 1. The arrows in the Figure 1 represent the data flow between different parts. In the feature extraction part, we selected feature points that can describe all MEMS LiDAR points. This function reduces the amount of data and speeds up processing. And in the pose estimation part, we utilized scan-to-map to estimate the vehicle pose. In addition, in the map construction part, all MEMS LiDAR points are mapped to the corresponding grid. When multiple points were projected into one grid, we applied the maximum-likelihood estimation to fuse the elevation at that grid. In the map update part, the consecutive frame data are accumulated by pose estimation to deal with the problem of LiDAR sparse sampling points. And the elevation map algorithm that incorporates range measurements with a sensor error model by Kalman filter was used.

3.1. Vehicle Pose Estimate

3.1.1. Feature Extraction

The pose estimation is the bridge between consecutive frame measurements. Therefore, the proposed method develops feature extraction and matching rules based on the characteristics of the MEMS solid-state LiDAR. The resolution of the MEMS solid-state LiDAR is higher than the mechanical LiDAR. As mentioned above, RS-LiDAR-M1 scans $125 \times 126 \times 5$ raw points in each frame, which is a heavy burden for the pose estimator.

It is not only slow but also susceptible to noise to match the raw point clouds directly. Therefore, we down-sampled the raw point cloud through a voxel grid filter [31]. And we further processed the point cloud data in the feature space. Inspired by LOAM [32], we extracted features from the down-sampled point cloud data to transform the point cloud into features with obvious physical significance to improve the quality of registration. At the same time, feature extraction achieves mapping from high-dimensional space to low-dimensional space, making the matching speed meet real-time requirements.

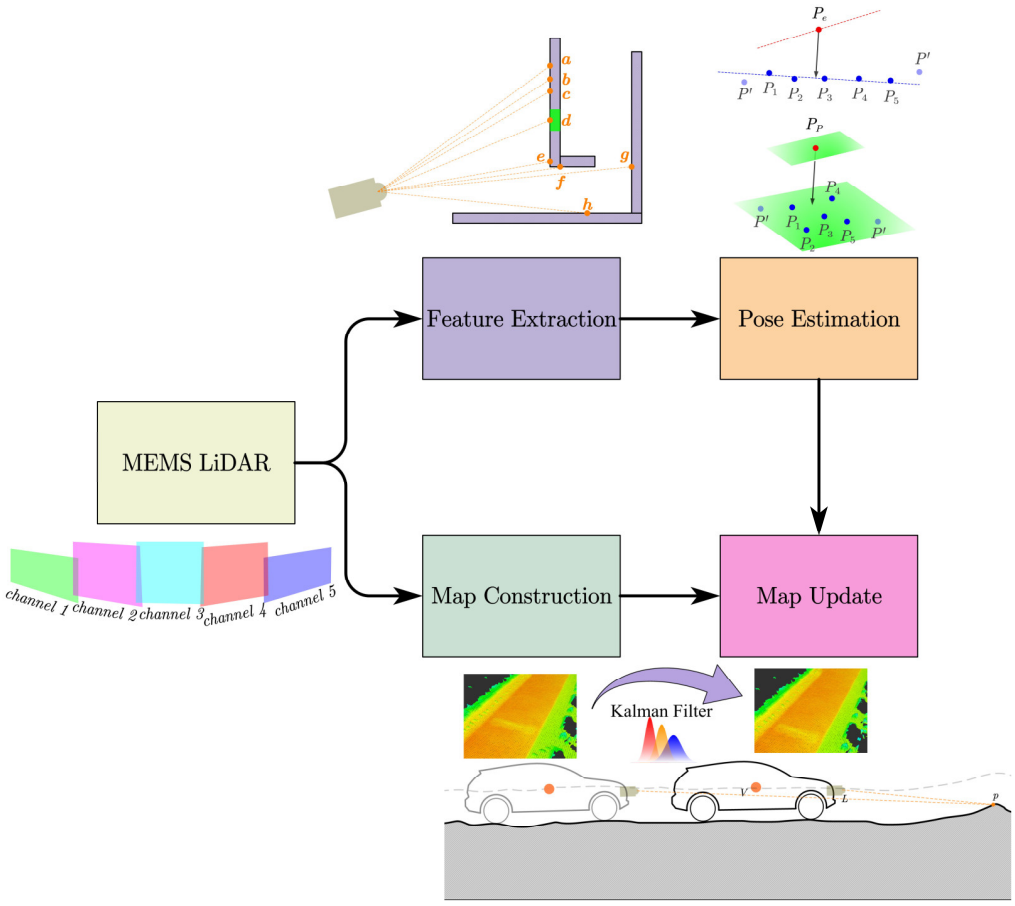


Figure 1. Framework of the proposed method.

Specifically, plane features and edge features were extracted from point clouds. And we divided them according to the local smoothness. The local smoothness f is described as

$$f = \frac{1}{\left|\mathcal{S}\right| \cdot \| P^L_{(k,i)} \|} \| \sum_{j \in \mathcal{S}, j \neq i} \left(P^L_{(k,i)} - P^L_{(k,j)} \right) \| \tag{1}$$

where $P^L_{(k,i)}$ represents the coordinate of the point i in the kth frame in the LiDAR coordinate system. And $P^L_{(k,j)}$ represents the coordinate of the point j in the kth frame in the LiDAR coordinate system. $\mathcal{S}$ represents the set of continuous points from the same row in each frame of the LiDAR. And half of the points in $\mathcal{S}$ are on either side of $P^L_{(k,i)}$. In this paper, we set the size of $\mathcal{S}$ to 10, namely $|\mathcal{S}|$ is 10. A larger local smoothness f means the surrounding plane is curved, while a smaller local smoothness f means the surrounding plane is smooth.

In practice, due to the staggered scanning channels of RS-LiDAR-M1, the data of the five channels are processed separately. In addition, unlike the mechanical LiDAR, the RS LiDAR-M1 uses a spiral scanning trajectory for each channel. The scanning trajectory at the edge of the channel has overlap and large curvature, which is marked with yellow

ellipses in Figure 2. And it does not conform to the assumption of smooth changes in adjacent points. Therefore, it is necessary to remove the spiral parts at the edge of the channel. The comparison before and after removal is shown in Figure 2. We have adopted different colors for each channel, which makes it easy to observe the reduction of the overlap between channels after removal.

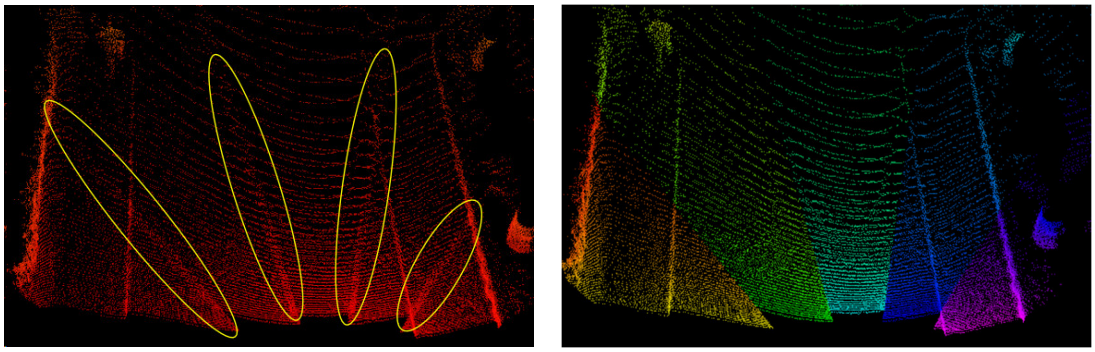


Figure 2. Illustration of comparison before and after removal.

In addition, in response to the small horizontal field FOV of RS LiDAR-M1, a reflection intensity smoothness f_I is defined to alleviate its impact on feature point selection. The f_I is described as

$$f_I = \frac{1}{|\mathcal{S}|}\left|\sum_{j\in\mathcal{S},j\neq i}\left(I(P^L_{(k,i)}) - I(P^L_{(k,j)})\right)\right| \tag{2}$$

where $I(P^L_{(k,i)})$ stands for the reflection intensity of point $P^L_{(k,i)}$, and $I(P^L_{(k,j)})$ stand for the reflection intensity of point $P^L_{(k,j)}$. For any LiDAR point $P = [x, y, z]$ with a reflectivity of RE, the reflection intensity is defined as $I(PC) = \frac{RE}{\sqrt{x^2+y^2+z^2}}$.

The points in each scan channel are sorted based on the f value. Then, we selected the point with the maximum f value as the edge feature and the minimum f value as plane feature. Each scan channel can provide a maximum of 1 edge feature and 2 plane features. As shown in Figure 3, a–c are regarded as plane features, while e and f are regarded as edge features. In practice, we set the plane feature threshold to be less than 0.2 and the edge feature threshold to be greater than 0.5. A point i can be selected as an edge feature or a plane feature only if its f value is larger or smaller than a threshold and the number of selected points does not exceed the maximum.

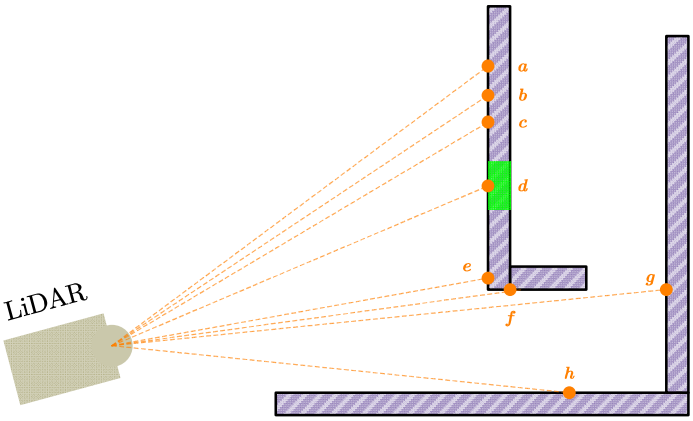


Figure 3. Illustration of the different types of the laser points.

Furthermore, to indicate the impact of the limited FOV of the solid-state LiDAR, we introduced reflectivity as another evaluation metric. Generally speaking, objects of different materials have different reflectivity. If the reflectivity of a point is very different from

the surrounding points, there is a high probability that this point is the edge of the two materials. Therefore, we extended the concept of edge feature points, which regards such points as edge feature points. As shown in Figure 3, due to the green part being another material with different reflectivity, the point d is an edge feature point.

Considering the special measuring function of the MEMS solid-state LiDAR, the point cloud should be processed first. As is shown in Figure 4, the RS-LiDAR-M1 obtains data through five channels simultaneously. In other words, 5 points are received at the same time. There is a certain stagger in the vertical arrangement direction between the channel and the channel. And the FOV of each channel is not exactly the same. Since each channel is spiral scanning, there is a huge curvature at the edge of each channel, which causes a problem in feature extraction. Hence, we eliminated these edge points to increase the pose estimation accuracy. In addition, the LiDAR points that meet the following conditions also need to be eliminated:

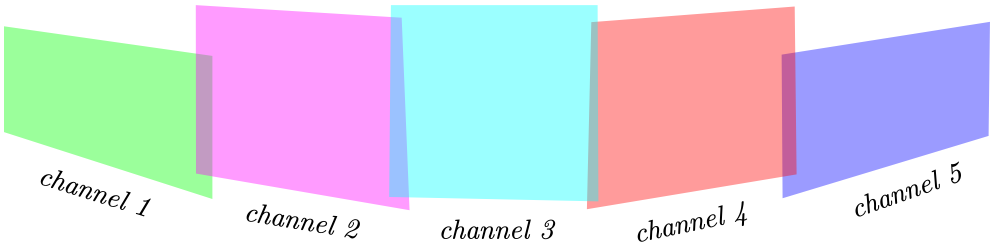


Figure 4. Demonstration of the RS-LiDAR-M1 scanning channel.

Points that are on the boundary of occluded regions, such as point g in Figure 3. If the LiDAR moves a little to another place, these points are unobservable.

Points on local plane surfaces that are roughly parallel to the laser beams, such as point h in Figure 3. These points change dramatically with tiny movements of the LiDAR, which is unreliable.

Points with strange intensity. The intensity describes the strength of the received laser signal. Both too high and too low usually lead to weak confidence and accuracy.

3.1.2. Pose Estimation

Pose estimation is the task of estimating the pose of the current moment relative to the previous moment based on historical scans. Estimation methods include scan-to-scan match and scan-to-map match [33]. A scan-to-scan match only relies on the point cloud data of the previous frame, which obtains a lower computational cost. However, the accuracy of this method is inevitably lower due to less information. Therefore, we implemented a scan-to-map match, which improves accuracy without excessive computational cost consumption.

In order to achieve a balance between performance and efficiency, we introduced a sliding window method to build the local map $M_k = \{P_{k-1}, P_{k-2}, \cdots, P_{k-n}\}$, where n is the number of frames to build the local map. More specifically, the local map is divided into the edge map and the plane map. And the map in every time step is built by the K-D tree [34] to increase search efficiency.

As mentioned above, matching on raw point clouds is less efficient and sensitive to noise. Thus, we leveraged matching edge points and plane points in feature space. To find the nearest edge point from the local map, the edge point p_e was projected to the local map by the following transformation:

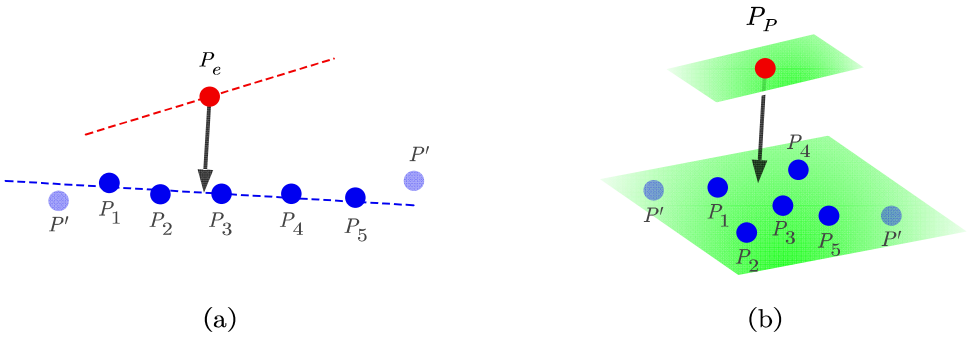
$$\hat{p}_e = T_k \cdot p_e \tag{3}$$

where T_k is the LiDAR pose in the kth frame and needs to be determined by the pose estimation.

In our work, we found five nearest points in the local edge map of each p_e, which is shown in Figure 5a. The arrow represents the corresponding relationship. To ensure that the five points are in a straight line such as P1–P5 located on blue dashed line in Figure 5a,

we computed the mean μ and covariance matrix Σ formed by those five points. If the maximum eigenvalue of the matrix Σ is more than three times larger than the second largest eigenvalue, we believe that those five points are on a straight line. Then, the edge-to-edge residual is computed as the following:

$$r_{e2c} = \frac{|(p_e - p_5) \times (p_e - p_1)|}{p_5 - p_1} \tag{4}$$



(a) (b)

Figure 5. Illustration of the feature point residual. (**a**) Edge-to-edge residual. (**b**) Plane-to-plane residual.

Similar to the edge residual, for each plane point p_p,we searched for the five nearest points in the local plane map such as P1–P5 located on green plane in Figure 5b, which is shown in Figure 5b. To ensure that five points are in the same plane, we also computed the mean μ and covariance matrix Σ formed by those five points. If the smallest eigenvalue of the matrix Σ is more than three times smaller than the second smallest eigenvalue, we believe that those five points are in the same plane. Then, the plane-to-plane residual is computed as the following:

$$r_{p2p} = \frac{(p_p - p_1)^T((p_3 - p_5) \times (p_3 - p_1))}{|(p_3 - p_5) \times (p_3 - p_1)|} \tag{5}$$

Finally, the vehicle pose is estimated by minimizing the edge-to-edge residual and plane-to-plane residual:

$$\underset{T_k}{\operatorname{argmin}} \sum r_{e2c} + \sum r_{p2p} \tag{6}$$

3.2. Road Elevation Map Update by Microelectromechanical System Light Detection and Ranging Measurement

Stable and accurate road elevation estimation plays a crucial role in suspension preview control. This section constructs a vehicle-centric local elevation map. Firstly, an isotropic model is deployed to represent LiDAR error, where the parameters are obtained through experiments. Then, the maximum-likelihood estimation is used to aggregate measurements in the same grid. Secondly, considering the impact of motion uncertainty in map updates, the error propagation of pose estimation is derived. At the same time, the gate strategy based on the Mahalanobis distance is adopted to filter the measurements falling into the grid. And the map is updated using a one-dimensional Kalman filter. Finally, the elevation of each grid is weighted using a confidence ellipse as output for road elevation estimation.

3.2.1. Grid Height Modeling Based on Light Detection and Ranging Error Model

(1) Noise Characterization of the LiDAR

The noise characteristics directly affect the accuracy of the elevation map. The beam model is a commonly used approximation model for LiDAR. However, the complex model results in low efficiency of point cloud data processing. It is difficult to ensure the real-

time requirements of elevation map construction. In reference to [35], the LiDAR error model can be represented as an anisotropic model, as shown in Figure 6. The anisotropic error model is parametrized with a vector representing the beam direction $\vec{b}$, supporting the standard deviation on depth σ_d. The standard deviation of the beam σ_r is supported implicitly by any vector perpendicular to $\vec{b}$. The impact of sunlight, reflection, and large intensity range are ignored. Then, the parameters σ_r and σ_d are defined as follows:

$$\sigma_r = \frac{0.6d + 1.48}{1000} \tag{7}$$

$$\sigma_d = 0.012 \tag{8}$$

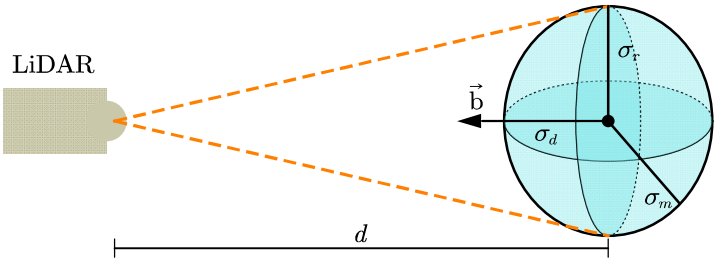


Figure 6. The error model of the LiDAR.

In addition, the error model is further simplified as an isotropic representation with only one standard deviation σ_m and is defined as follows:

$$\sigma_m = \max\{\sigma_r, \sigma_d\} \tag{9}$$

(2) Definition of Grid Height

There are multiple point clouds that fall into one grid, and it is necessary to form a unified description of the grid height through certain processing to improve computational efficiency. After modeling the aforementioned LiDAR error model, each measurement can be described as a Gaussian distribution. Therefore, this section uses maximum likelihood estimation to obtain a unified description of the Gaussian distribution of grid height.

The grid height in the map coordinate system M is defined as a Gaussian distribution $\mathcal{N}\left(p, \sigma_p^2\right)$. And the LiDAR measurements are also approximated as Gaussian distribution $\mathcal{N}\left(p_i, \sigma_{p_i}^2\right)$ by the error model. When many different measurements p_i with known variances $\sigma_{p_i}^2$ fall into the grid, each grid of the elevation map is updated by multiple LiDAR measurements. The maximum-likelihood estimation [36] is adopted in this paper to calculate the grid height.

The probability density function of a Gaussian distribution with mean p and variance $\sigma_{p_i}^2$ is described as the following:

$$f_i\left(p_i \middle| p, \sigma_{p_i}^2\right) = \frac{1}{\sqrt{2\pi\sigma_{p_i}^2}} \exp\left(-\frac{(p_i - p)^2}{2\sigma_{p_i}^2}\right) \tag{10}$$

The likelihood function $L(p)$ is calculated from the product of the probability density function of all n measurements. The equation is as follows:

$$L(p) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{p_i}^2}} \exp\left(-\frac{(p_i - p)^2}{2\sigma_{p_i}^2}\right) \tag{11}$$

To determine the maximum of the likelihood function, the derivative of the likelihood function is taken, and its derivative is set to zero. Taking the logarithm of the likelihood function simplifies the calculation process, and the result remains the same.

$$\frac{\partial}{\partial p} \log L(p) = \sum_{i=1}^{n} \frac{p_i - p}{\sigma_{p_i}^2} \tag{12}$$

Let Equation (13) be equal to 0. Then, solving this equation results in the following:

$$p = \sum_{i=1}^{n} \frac{p_i}{\sigma_{p_i}^2} / \sum_{i=1}^{n} \frac{1}{\sigma_{p_i}^2} \tag{13}$$

$$\sigma_p^2 = \frac{1}{n} \sum_{i=1}^{n} (p_i - p)^2 \tag{14}$$

3.2.2. Map Update from Grid Height

(1) Coordinate Systems

As shown in Figure 7, there are four coordinate frames in this paper: the vehicle coordinate system V, the LiDAR coordinate system L, the map coordinate system M, and the internal coordinate system I. The vehicle coordinate system V is fixed to the vehicle centroid (the orange dot in the vehicle). And the LiDAR coordinate system L is fixed to the LiDAR centroid. There exists a known transformation T_{LV} between the LiDAR coordinate system and the vehicle coordinate system V. With this known transformation, we can convert the LiDAR measurements to the vehicle coordinate system. The map coordinate system M is defined in relation to the vehicle coordinate system V with transformation T_{VM}. The internal system is fixed to the environment. And it is used as a reference for other coordinate systems. The orange arrow indicates the forward direction of the vehicle. The orange dashed lines describe the LiDAR range measurement for point p.

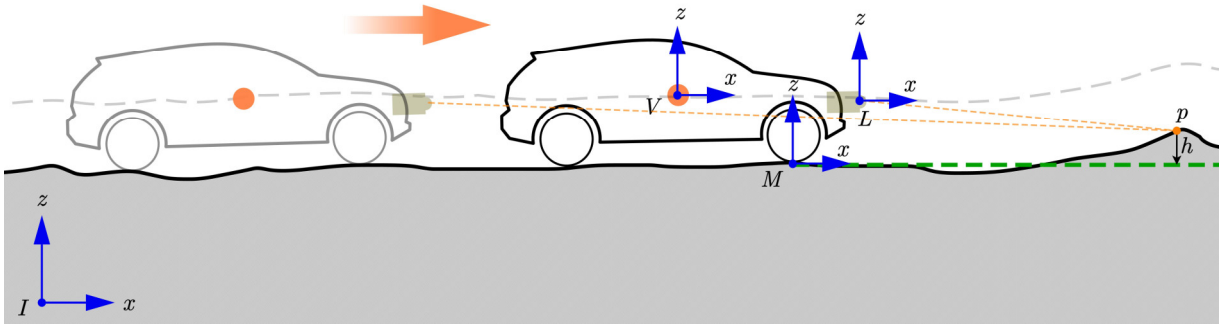


Figure 7. Illustration of the coordinate systems.

(2) Elevation Map Update Based on Motion Uncertainty

Map update mainly consists of two parts. One part is to use the grid height modeled in the previous section to achieve elevation map construction through one-dimensional Kalman filtering. The other part is to derive the error propagation of pose estimation in map update.

The Kalman filter is implemented to achieve elevation map estimation $\left(\hat{h}, \sigma_h^2\right)$ from the grid height measurement $\left(p, \sigma_p^2\right)$. When the LiDAR scans a point p, it needs to be mapped to the map coordinate system. As shown in Figure 7, a single measurement, given as the range ${}_L r_{LP_i}$ in the LiDAR frame, can be transformed into the corresponding measurement p_i with

$$p_i = proj_z(\Phi_{LM}^{-1}({}_L r_{LP_i}) - {}_M r_{LM}) \tag{15}$$

where $proj_z = [0 \quad 0 \quad 1]$ maps the three-dimensional measurement to the scalar measurement p_i.

However, the grid height is established in the map coordinate system, while the LiDAR noise model is established in the LiDAR coordinate system. Therefore, it is necessary to calculate the error propagation introduced by coordinate system conversion. The error propagation for the variance $\sigma_{p_i}^2$ is given as

$$\sigma_{P_i}^2 = J_L \Sigma_L J_L^T + J_\Phi \Sigma_{\Phi_{IL}} J_\Phi^T \tag{16}$$

where J_L and J_Φ represent the Jacobians for the LiDAR measurement and the sensor frame rotation, respectively. J_L and J_Φ are described as

$$J_L = \frac{\partial p}{\partial_L r_{LP}} = proj_z C(\Phi_{LM})^T \tag{17}$$

$$J_\Phi = \frac{\partial p}{\partial \Phi_{LM}} = proj_z C(\Phi_{LM})^T {}_L r_{LP}^\times \tag{18}$$

where $C(\cdot)$ is used to describe the mapping to the corresponding rotation matrix. The superscript $\times$ represents the skew-symmetric matrix of the corresponding vector.

Moreover, $\Sigma_{\Phi_{IL}}$ represents the rotation covariance matrix of the LiDAR. According to the relationship defined by the coordinate system, the z-axis of the elevation map coordinate system and the inertial coordinate system are always aligned. And the LiDAR coordinate system and elevation map coordinate system are fixedly connected to the vehicle body. Therefore, the measurement uncertainty of the z-axis rotation in $\Sigma_{\Phi_{IL}}$ is zero. $\Sigma_{\Phi_{IL}}$ only includes uncertainty in pitch and roll. Σ_L is the covariance matrix of the LiDAR error model. According to the isotropic error model defined above, the equation is as follows:

$$\Sigma_L = \begin{bmatrix} \sigma_m^2 & 0 & 0 \\ 0 & \sigma_m^2 & 0 \\ 0 & 0 & \sigma_m^2 \end{bmatrix} \tag{19}$$

In this paper, we only selected the grid height measurement $\left(p, \sigma_p^2\right)$ as the state. Due to the irregular changes in the height direction, the one-dimensional Kalman filter only includes the update part.

$$K(k) = \sigma_h^{2-}(k) H^T \left(H \sigma_h^{2-}(k) H^T + \sigma_p^2(k) \right)^{-1} \tag{20}$$

$$\hat{h}(k) = \hat{h}^-(k) + K(k) \left(p(k) - H \hat{h}^-(k) \right) \tag{21}$$

$$\sigma_h^2(k) = (I - K(k)H) \sigma_h^{2-}(k) \tag{22}$$

where $\hat{h}^-(k)$ and $\sigma_h^{2-}(k)$ represent the priori estimation of the elevation map and its error covariance at time step k, respectively. $\hat{h}(k-1)$ and $\sigma_h^2(k-1)$ represent the state estimation of the elevation map and its error covariance at time step k, respectively. $p(k)$ is the grid height measurement at time step k. And $\sigma_p^2(k)$ is the measurement noise covariance at time step k. $K(k)$ is the Kalman gain at time step k. $H(k)$ is the measurement matrix at time step k. I represents the identity matrix. H represents the measurement matrix, which is an identity matrix. Consequently, the one-dimensional Kalman filter is rewritten as the following:

$$\hat{h}(k) = \frac{\sigma_p^2(k) \hat{h}^-(k) + \sigma_h^{2-}(k) p(k)}{\sigma_p^2(k) + \sigma_h^{2-}(k)} \tag{23}$$

$$\sigma_h^2(k) = \frac{\sigma_h^{2-}(k) \sigma_p^2(k)}{\sigma_h^{2-}(k) + \sigma_p^2(k)} \tag{24}$$

Due to the assumption that the inertial coordinate system is stationary and the elevation map coordinate system is fixed to the vehicle body, it is necessary to align the map in consecutive frames by pose estimation. As shown in Figure 8, for time k, the map coordinate system (blue coordinate system) is associated with the vehicle coordinate system (red coordinate system) through mapping $\left(r_{\widetilde{V}_k M_k}, \Phi_{\widetilde{V}_k M_k}\right)$. At time $k = 2$, the estimation $\hat{r}_{M_2 P}$ of point P in the map coordinate system M_2 can be represented by the estimation $\hat{r}_{M_1 P}$ of the map coordinate system M_1 at time $k = 1$ (orange dashed line), i.e., the following:

$$\hat{r}_{M_2 P} = -r_{\widetilde{V}_2 M_2} - \hat{r}_{\widetilde{V}_1 \widetilde{V}_2} + r_{\widetilde{V}_1 M_1} + \hat{r}_{M_1 P} \tag{25}$$

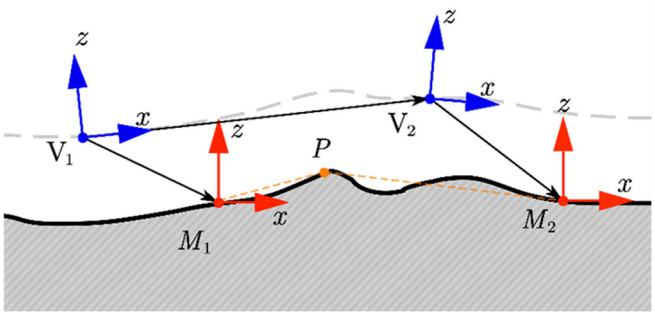


Figure 8. Demonstration of map update based on motion uncertainty.

Rewrite Equation (24) using the reference coordinate system M_2 as

$${}_{M_2} r_{M_2 P} = -r_{\widetilde{V}_2 M_2} - \Phi^{-1}_{\widetilde{V}_2 M_2}\left({}_{\widetilde{V}_2}\hat{r}_{\widetilde{V}_1 \widetilde{V}_2}\right) + \hat{\Phi}^{-1}_{M_1 M_2}\left({}_{M_1} r_{\widetilde{V}_1 M_1} + {}_{M_1}\hat{r}_{M_1 P}\right) \tag{26}$$

By using Equation (25), any estimation in M_1 can be mapped to M_2. And assuming $r_{M_k P} \sim \mathcal{N}\left(\hat{r}_{M_k P}, \Sigma_{P,k}\right)$, the propagation of covariance from $k = 1$ to $k = 2$ can be written as follows:

$$\Sigma_{P,2} = J_P \Sigma_{P,1} J_P^T + J_r \Sigma_r J_r^T + J_\Phi \Sigma_\Phi J_\Phi^T \tag{27}$$

where $\Sigma_{P,1}$ is the covariance matrix at $k = 1$. The covariance matrices Σ_r and Σ_Φ represent the motion uncertainty of the vehicle reference coordinate systems $\widetilde{V}_1$ to $\widetilde{V}_2$. They are both modeled by Gaussian distribution, $r_{\widetilde{V}_1 \widetilde{V}_2} \sim \mathcal{N}\left(\hat{r}_{\widetilde{V}_1 \widetilde{V}_2}, \Sigma_r\right)$ and $\Phi_{\widetilde{V}_1 \widetilde{V}_2} \sim \mathcal{N}\left(\hat{\Phi}_{\widetilde{V}_1 \widetilde{V}_2}, \Sigma_\Phi\right)$. Moreover, the Jacobian matrix corresponding to the three parts is as follows:

$$J_P = \frac{\partial_{M_2} \hat{r}_{M_2 P}}{\partial_{M_1} \hat{r}_{M_1 P}} = C(\hat{\Phi}_{M_1 M_2})^T = I \tag{28}$$

$$J_r = \frac{\partial_{M_2} \hat{r}_{M_2 P}}{\partial_{\widetilde{V}_2} \hat{r}_{\widetilde{V}_1 \widetilde{V}_2}} = -C(\hat{\Phi}_{\widetilde{V}_2 M_2})^T \tag{29}$$

$$J_\Phi = \frac{\partial_{M_2} \hat{r}_{M_2 P}}{\partial \hat{\Phi}_{\widetilde{V}_1 \widetilde{V}_2}} = -\left({}_{M_1} r_{\widetilde{V}_1 M_1} + {}_{M_1}\hat{r}_{M_1 P}\right)^\times C(\hat{\Phi}_{\widetilde{V}_1 M_1})^T \tag{30}$$

(3) Elevation Weighted Method Based on Confidence Eclipse

Motion uncertainty not only affects elevation but also accumulates errors in the xy direction. When calculating the height of a grid, the neighboring grids covered by motion uncertainty also have an impact on it. Therefore, this section uses the confidence ellipse method to search a certain range based on covariance to weigh the elevation value of the grid. Only the grid within the area where the wheel trajectory passes through is height-weighted, which can filter the elevation while ensuring real-time performance. We adopted a 95% confidence ellipse to extract grid elevation, including the acquisition of axis vectors and the calculation of fusion weights.

The major and minor axis of the confidence ellipse with a 95% confidence are $2\sqrt{5.991\lambda_1}$ and $2\sqrt{5.991\lambda_2}$, respectively, where λ_1 and λ_2 are the maximum eigenvalue and minimum eigenvalue of the covariance matrix. If the x-axis and y-axis data have a correlation, then the major and minor axes of the confidence ellipse are not aligned with the coordinate axis. And the angle between them can be determined by the eigenvectors corresponding to the eigenvalues. The angle between the major axis and the x-axis is calculated with the following:

$$\gamma = \arctan\left(\frac{v_1}{v_x}\right) \tag{31}$$

where v_1 represents the eigenvector corresponding to the maximum eigenvalue of the covariance matrix, namely, the eigenvector of the major axis. v_x represents the unit vector of the x-axis.

For a grid, assuming its confidence ellipse covers n adjacent grids, the weight of the surrounding grid i to that grid is ω_i, which is defined as the following:

$$\omega_i = \left(CDF_x(a_x + \frac{b}{2}) - CDF_x(a_x - \frac{b}{2})\right) \cdot \left(CDF_y(a_y + \frac{b}{2}) - CDF_y(a_y - \frac{b}{2})\right) \tag{32}$$

where a_x and a_y represent the distance from grid i to the updating grid in the x and y directions, respectively. CDF_x and CDF_y represent the Gaussian cumulative distribution function in the x and y directions, respectively.

Finally, the fusion elevation of the grid is calculated after obtaining the weights of all adjacent grids within the confidence ellipse with the following formulas:

$$\hat{h} = \frac{\sum_{i=1}^{n} \omega_i \hat{h}_i}{\sum_{i=1}^{n} \omega_i} \tag{33}$$

$$\sigma_h^2 = \frac{\sum_{i=1}^{n} \omega_i (\sigma_{h,i}^2 + \hat{h}_i^2)}{\sum_{i=1}^{n} \omega_i} - \hat{h}^2 \tag{34}$$

(4) The Gate Strategy Based on Mahalanobis Distance

Since there is no clear kinematics relationship for height change, the map cannot realize the sudden change in height immediately when encountering an object whose height direction changes sharply. Therefore, this section proposed a gate strategy to help the map respond quickly.

The gate based on the Mahalanobis distance is defined as the following:

$$Mahalanobis(k) = \left(p(k) - \hat{h}(k-1)\right)^T \left(\frac{\sigma_h^2(k-1) + \sigma_p^2(k)}{2}\right)^{-1} \left(p(k) - \hat{h}(k-1)\right) \tag{35}$$

If the Mahalanobis distance is less than gate size c, it is considered that the measurement falls into the gate, and there is no sharp change in elevation. When encountering objects with sharp increases or decreases in height, such as walls and stone piers, we selected the larger of the measurement and estimated it as output. Based on the above constraints, the gate strategy is described as follows:

$$\hat{h}(k) = \begin{cases} p(k) & p(k) > \hat{h}(k-1) \text{ and } Mahalanobis(k) > c \\ \hat{h}(k-1) & p(k) < \hat{h}(k-1) \text{ and } Mahalanobis(k) > c \\ \frac{\sigma_h^2(k-1)p(k) + \sigma_p^2(k)\hat{h}(k-1)}{\sigma_h^2(k-1) + \sigma_p^2(k)} & else \end{cases} \tag{36}$$

$$\sigma_{\hat{h}}^2(k) = \begin{cases} \sigma_p^2(k) & p(k) > \hat{h}(k-1) \text{ and } Mahalanobis(k) > c \\ \sigma_h^2(k-1) & p(k) < \hat{h}(k-1) \text{ and } Mahalanobis(k) > c \\ \frac{1}{\frac{1}{\sigma_h^2(k-1)} + \frac{1}{\sigma_p^2(k)}} & else \end{cases} \tag{37}$$

4. Experiment Evaluation

The sport utility vehicle demonstrator equipped with a MEMS solid-state LiDAR and a GPS/IMU is shown in Figure 9. The sensor parameters are listed in Table 2. The MEMS solid-state LiDAR is installed in the middle bottom of the front of the vehicle, and the GPS/IMU is installed

in the center of the rear axle in the trunk. The proposed road elevation estimation algorithm is executed on an IPC ADVANTECH MIC-770 with an Intel i7-9700E CPU at 2.6 GHz, 16 GB RAM, and Ubuntu 18.04 OS. And it is implemented as a C++ library with an interface to the Robot Operating System (ROS) [37]. For efficient data handling and operations, the software builds upon the grid map library [38]. Furthermore, we evaluated our proposed method from height accuracy and time efficiency in the real-world experiment.

Figure 9. The sport utility vehicle demonstrator.

Table 2. The detail parameters of the sensors.

Sensor	Parameters
RoboSense RS-LiDAR-M1	Type: MEMS Solid-state Frequency: 10 Hz FoV: 120° × 25° Horizontal Resolution: 0.2° Vertical Resolution: 0.2° Weight: 0.8 kg Detection Range: 0.7–200 m Accuracy: 2 cm
CHC CGI-220	Frequency: 100 Hz Position accuracy: 1 cm + 1 ppm (RTK) Pose accuracy: 0.1° (baseline $\geq$ 2 m)

4.1. Accuracy Analysis

The primary indicator of the mapping algorithm is accuracy since road unevenness is important for ride comfort, which is strongly related to suspension preview control. The evaluation scenario is shown in Figure 10. The car is driven at a speed of 36 km/h. And we put a cuboid with the size of $0.6 \times 0.4 \times 0.05$ m^3 in the scenario. To evaluate the accuracy, the ground truth is divided into two parts. One is the cuboid part, which uses its own size parameters as the ground truth. Another one is the road part, which uses the Hi-Target D8Pro GNSS receiver to survey the road as the ground truth with a position accuracy of 5 mm + 0.5 ppm. The visualization of the road elevation map is shown in Figure 11. In Figure 11, the color of the grid represents the height. And red indicates higher elevation values and green indicates lower elevation values. We manually picked the cuboid from the built map and compared the height with the ground truth, i.e., 0.05 m, to obtain the elevation accuracy. Then, four grid sizes of 0.05, 0.10, 0.15, and 0.20 m are set to evaluate the

accuracy with respect to the resolution. Additionally, the Octomap is employed as a comparison, and the results are shown in Table 3. The error of the cuboid height of the proposed method is no more than 5 mm when the resolution is 0.05 m. This error has almost no impact on a tire wheel due to its characteristics [39]. As a comparison, the elevation accuracy is affected by the resolution using Octomap due to the 3D discretization.

Figure 10. The evaluation scenario.

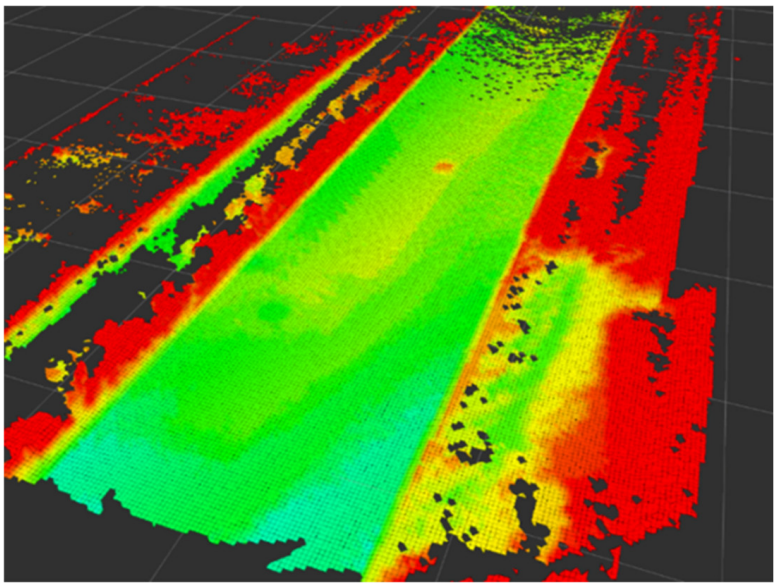


Figure 11. The visualization of road elevation map.

Table 3. The mapping accuracy comparison.

Resolution (m)	**0.01**	**0.2**	**0.05**	**0.10**
Ours (cm)	0.48	0.64	0.83	0.97
Octomap (cm)	1.25	2.24	3.59	4.16

4.2. Time Efficiency

Time efficiency is another important indicator of the mapping algorithm supporting online control. When the system cannot achieve real-time performance, elevation maps can be sparse and do not sufficiently feed enough information to the controller. We first evaluated the computation time of the elevation map on IPC. Then, we compared the time with other comparative methods to show the evolution in a large-scale environment.

To evaluate the computation time, we ran both methods in the above evaluation scenario and recorded the frame rate. In the proposed method, the map size is 15×9 m^2 with 0.05 m resolution. Consequently, our approach achieves almost 25 Hz on IPC and Octomap only 8 Hz. Like most voxel representations, integration of measurements by raycasting and nonrigid transformations is computationally expensive to perform. Therefore, the time required for map generation of Octomap lagged significantly behind that for sensor data acquisition (10 Hz).

Moreover, scalability is also an important metric of time efficiency. To validate the scalability, we compared the running time of the elevation map building of the proposed model with the size of 15×9 m^2, 20×12 m^2, and the resolution of 0.10 m and 0.20 m. As shown in Figure 12, when the size of the elevation map becomes larger, more cells are allocated for map building, resulting in more computation time. Considering that the cells in the map 20×12 m^2 with 0.10 m resolution have seven times more grids than those with 15×9 m^2 with 0.20 m resolution, the computation time only grows less than one point seven times. In addition, even if the mapping environment becomes larger and larger, the proposed algorithm still has good scalability, which is important for real-time suspension control.

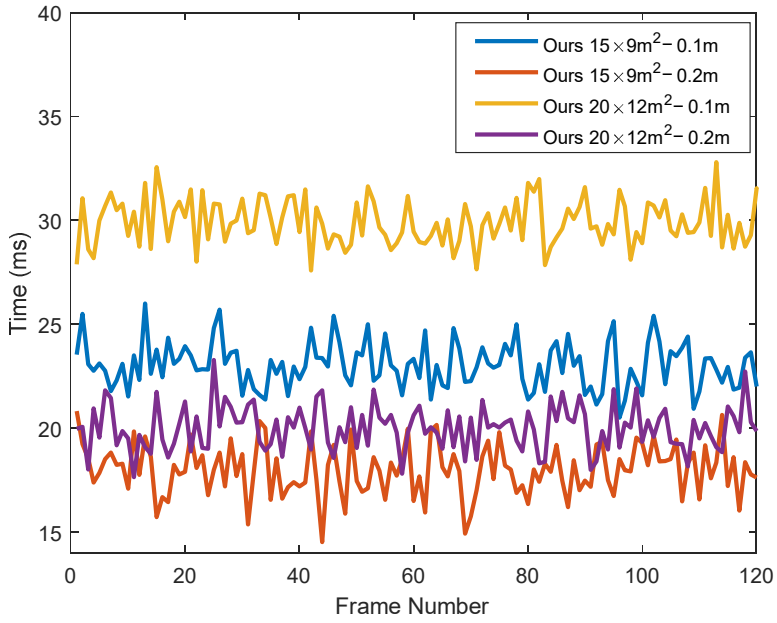


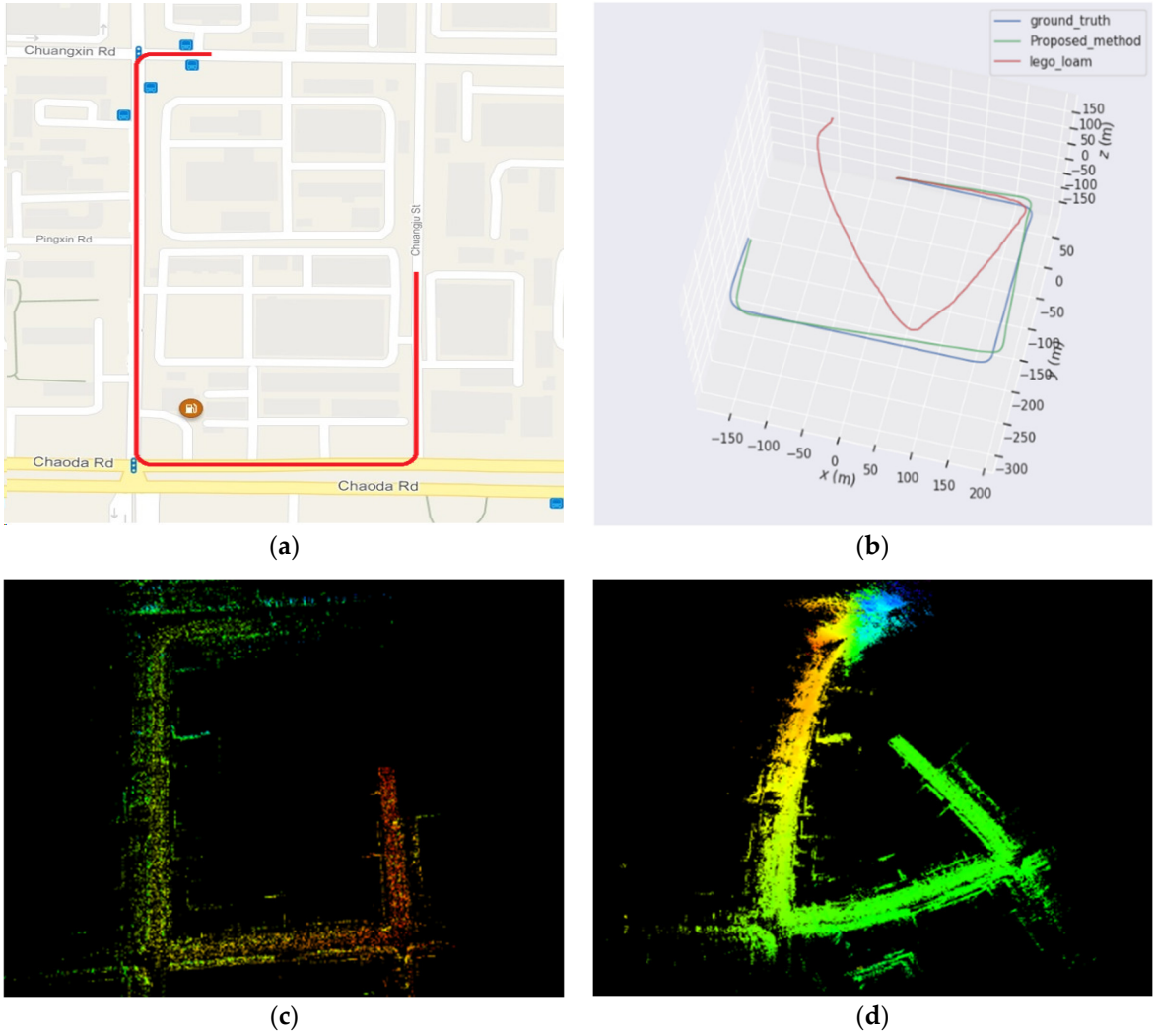
Figure 12. Time scalability of the proposed method.

4.3. Pose Estimation Performance

In the suspension preview control, it is only necessary to extract the elevation from the local map, which has a low requirement for global consistency. However, the generation process of the local map still needs to introduce the vehicle pose. This paper compares our pose estimation algorithm with the open-source algorithm LeGO-LOAM in the urban outdoor scenario that is shown in Figure 13a. The trajectory of the car is the red line in Figure 13a. The car was driven at a speed of 36 km/h. In order to obtain a good GPS signal, the experiment is carried out on an unobstructed road; thus, the feature points may be sparse. In addition, the test is conducted in sunny weather. Moreover, there are some snows on the road that may affect the LiDAR.

The comparison results are shown in Figure 13b, where the trajectories of ground truth, LeGO-LOAM, and our method are plotted in blue, red, and green, respectively. The RTK signal is utilized to represent the ground truth. The proposed method can accurately estimate the trajectory in the urban outdoor scenario. The LeGO-LOAM algorithm can also correctly estimate the vehicle pose when running straight, but the trajectory has a large error during turning. Figure 13c,d are the point

clouds aligned by the pose estimated using our proposed algorithm and LeGO-LOAM. It can be shown that the algorithm proposed in this paper has a high consistency, while the LeGO-LOAM algorithm has a poor effect due to the low accuracy of vehicle rotation estimation.



(**a**) (**b**)

(**c**) (**d**)

Figure 13. Comparison results of pose estimation algorithm. (**a**) Target trajectory overlaid on Google Maps. (**b**) Estimated trajectories. (**c**) The point clouds aligned with ours. (**d**) The point clouds aligned with LeGO-LOAM.

5. Conclusions

The traditional method of utilizing suspension sensors makes it hard to handle the irregular road impulses that occur ahead of vehicles in urban environments, which seriously affects ride comfort. Therefore, we have presented a road impulse estimation method in an urban environment with a MEMS LiDAR that addresses the problem of road sensing for suspension preview control. The main novelty of the proposed method is estimating the elevation in the local coordinate so that the process of incorporating the new measurements into the elevation map is only affected by the noise of the range sensor and the accuracy of the pose estimation. The method includes four key components. First, in order to solve the problem of sparse sampling points of the LiDAR, the proposed algorithm deploys a real-time estimation of the vehicle's 3D pose. Second, taking the sensor error model of LiDAR into account, the proposed algorithm builds the grid height measurement model to aggregate multiple points falling into one grid. Third, it constructs local elevation maps using the Kalman filter and introduces motion uncertainty to accomplish accurate elevation map updates. Lastly, a gate strategy based on the Mahalanobis distance is integrated to deal with the sharp changes in elevation. The proposed method is tested in a real outdoor environment. Finally, the accuracy and efficiency of the proposed method are validated in a real-world experiment, demonstrating its feasibility for

road sensing. In the future, it will entail further considering the impact of dynamic objects in the scenario on elevation estimation. Our subsequent work will leverage deep learning for dynamic object detection. Then, they will be separated from the elevation map to overcome any adverse effects of dynamic objects.

Author Contributions: Conceptualization, S.Z. and M.W.; methodology, S.Z. and M.W.; writing—original draft, S.Z. and M.W.; software, M.W.; validation, S.Z.; funding acquisition, K.G., Y.W. and P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the China Postdoctoral Science Foundation (Grant No. 2022M720433).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset available on request from the authors.

Acknowledgments: All opinions expressed in this paper are solely those of the authors and do not represent those of the sponsors. The authors would like to thank the experienced anonymous reviewers for their constructive and valuable suggestions for improving the overall quality of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Theunissen, J.; Tota, A.; Gruber, P.; Dhaens, M.; Sorniotti, A. Preview-based techniques for vehicle suspension control: A state-of-the-art review. *Annu. Rev. Control* **2021**, *51*, 206–235. [CrossRef]
2. Zhao, S.; Wang, Y.; Wang, P.; Ma, T.; Guo, K. Adaptive Non-Linear Joint Probabilistic Data Association for Vehicle Target Tracking. *IEEE Access* **2021**, *9*, 14138–14147. [CrossRef]
3. Wang, H.; Wang, C.; Xie, L. Lightweight 3-D Localization and Mapping for Solid-State LiDAR. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1801–1807. [CrossRef]
4. Lin, J.; Zhang, F. Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 3126–3131.
5. Pan, Y.; Xu, X.; Ding, X.; Huang, S.; Wang, Y.; Xiong, R. GEM: Online Globally Consistent Dense Elevation Mapping for Unstructured Terrain. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–13. [CrossRef]
6. Zieliński, K.; Belter, D. Keyframe-based Dense Mapping with the Graph of View-Dependent Local Maps. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 10744–10750.
7. Zhao, D.; Wang, L.; Li, Y.; Du, M. Extraction of preview elevation of road based on 3D sensor. *Measurement* **2018**, *127*, 104–114. [CrossRef]
8. Wang, L.; Zhao, D.; Ni, T.; Liu, S. Extraction of Preview Elevation Information Based on Terrain Mapping and Trajectory Prediction in Real-Time. *IEEE Access* **2020**, *8*, 76618–76631. [CrossRef]
9. Ewen, P.; Li, A.; Chen, Y.; Hong, S.; Vasudevan, R. These Maps are Made for Walking: Real-Time Terrain Property Estimation for Mobile Robots. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7083–7090. [CrossRef]
10. Elfes, A. Sonar-based real-world mapping and navigation. *IEEE J. Robot. Autom.* **1987**, *3*, 249–265. [CrossRef]
11. Kudriashov, A.; Buratowski, T.; Garus, J.; Giergiel, M. 3D environment exploration with slam for autonomous mobile robot control. *WSEAS Trans. Syst. Control* **2021**, *16*, 450–456. [CrossRef]
12. Liu, Y.; Emery, R.; Chakrabarti, D.; Burgard, W.; Thrun, S. Using EM to Learn 3D Models with Mobile Robots. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML), Williamstown, MA, USA, 28 June–1 July 2001; pp. 329–336.
13. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
14. Saarinen, J.; Andreasson, H.; Stoyanov, T.; Lilienthal, A.J. 3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments. *Int. J. Robot. Res.* **2013**, *32*, 1627–1644. [CrossRef]
15. Saarinen, J.; Andreasson, H.; Stoyanov, T.; Ala-Luhtala, J. Normal Distributions Transform Occupancy Maps: Application to large-scale online 3D mapping. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 2233–2238.
16. Oleynikova, H.; Millane, A.; Taylor, Z.; Galceran, E.; Nieto, J.I.; Siegwart, R.Y. Signed distance fields: A natural representation for both mapping and planning. In Proceedings of the RSS Workshop: Geometry Beyond-Representations, Physics, Scene Understanding Robot, Ann Arbor, MI, USA, 20–22 June 2016; pp. 1–6.

17. Canelhas, D.R.; Stoyanov, T.; Lilienthal, A.J. SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3671–3676.
18. Sommer, C.; Sang, L.; Schubert, D.; Cremers, D. Gradient-SDF: A Semi-Implicit Surface Representation for 3D Reconstruction. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 24 June 2022; pp. 6270–6279.
19. Grinvald, M.; Furrer, F.; Novkovic, T.; Chung, J.; Cadena, C.; Siegwart, R.; Nieto, J. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3037–3044. [CrossRef]
20. Lang, R.; Fan, Y.; Chang, Q. SVR-Net: A Sparse Voxelized Recurrent Network for Robust Monocular SLAM with Direct TSDF Mapping. *Sensors* **2023**, *23*, 3942. [CrossRef] [PubMed]
21. Fankhauser, P.; Bloesch, M.; Hutter, M. Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3019–3026. [CrossRef]
22. Fankhauser, P.; Bloesch, M.; Gehring, C.; Hutter, M.; Siegwart, R. Robot-centric elevation mapping with uncertainty estimates. In *Mobile Service Robotics*; World Scientific: Singapore, 2014; pp. 433–440.
23. Souza, A.; Gonçalves, L. Occupancy-elevation grid: An alternative approach for robotic mapping and navigation. *Robotica* **2016**, *34*, 2592–2609. [CrossRef]
24. Zhou, H.; Ping, P.; Shi, Q.; Chen, H. An Adaptive Two-Dimensional Voxel Terrain Mapping Method for Structured Environ-ment. *Sensors* **2023**, *23*, 9523. [CrossRef]
25. Katyal, K.; Popek, K.; Paxton, C.; Moore, J.; Hager, G.D. Occupancy map prediction using generative and fully convolutional networks for vehicle navigation. *arXiv* **2018**, arXiv:1803.02007.
26. Sharma, V.D.; Chen, J.; Shrivastava, A.; Tokekar, P. Occupancy map prediction for improved indoor robot navigation. *arXiv* **2022**, arXiv:2203.04177.
27. Hoeller, D.; Rudin, N.; Choy, C.; Anandkumar, A.; Hutter, M. Neural scene representation for locomotion on structured terrain. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8667–8674. [CrossRef]
28. Stölzle, M.; Miki, T.; Gerdes, L.; Azkarate, M.; Hutter, M. Reconstructing occluded elevation information in terrain maps with self-supervised learning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 1697–1704. [CrossRef]
29. Yang, B.; Zhang, Q.; Geng, R.; Wang, L.; Liu, M. Real-Time Neural Dense Elevation Mapping for Urban Terrain with Uncer-tainty Estimations. *IEEE Robot. Autom. Lett.* **2023**, *8*, 696–703. [CrossRef]
30. Barranquero, M.; Olmedo, A.; Gómez, J.; Tayebi, A.; Hellín, C.J.; Saez de Adana, F. Automatic 3D Building Reconstruction from OpenStreetMap and LiDAR Using Convolutional Neural Networks. *Sensors* **2023**, *23*, 2444. [CrossRef]
31. Yang, J.; Wang, C.; Luo, W.; Zhang, Y.; Chang, B.; Wu, M. Research on Point Cloud Registering Method of Tunneling Roadway Based on 3D NDT-ICP Algorithm. *Sensors* **2021**, *21*, 4448. [CrossRef]
32. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in real-time. In Proceedings of the Robotics: Science and Systems Conference (RSS), Berkeley, CA, USA, 12–16 July 2014.
33. Ryu, K.; Dantanarayana, L.; Furukawa, T.; Dissanayake, G. Grid-based scan-to-map matching for accurate 2D map building. *Adv. Robot.* **2016**, *30*, 431–448. [CrossRef]
34. Greenspan, M.; Yurick, M. Approximate k-d tree search for efficient ICP. In Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling, Banff, AB, Canada, 6–10 October 2003; pp. 442–448.
35. Pomerleau, F.; Breitenmoser, A.; Liu, M.; Colas, F.; Siegwart, R. Noise characterization of depth sensors for surface inspections. In the Proceedings of the 2nd International Conference on Applied Robotics for the Power Industry (CARPI), Zurich, Switzerland, 11–13 September 2012; pp. 16–21.
36. Göhrle, C.; Schindler, A.; Wagner, A.; Sawodny, O. Road Profile Estimation and Preview Control for Low-Bandwidth Active Suspension Systems. *IEEE/ASME Trans. Mechatron.* **2014**, *20*, 2299–2310. [CrossRef]
37. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-Source Robot Operating System. In Proceedings of the IEEE ICRA Workshop on Open Source Software, Guiyang, China, 18–20 September 2009; IEEE: Piscataway, NJ, USA, 2009.
38. Fankhauser, P.; Hutter, M. *A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation, Robot Operating System (ROS)—The Complete Reference*; Koubaa, A., Ed.; Springer International Publishing: Cham, Switzerland, 2016; Volume 1.
39. Liu, B.; Zhao, D.; Chang, J.; Yao, S.; Ni, T.; Gong, M. Statistical Terrain Model with Geometric Feature Detection Based on GPU Using LiDAR on Vehicles. *Meas. Sci. Technol.* **2022**, *33*, 095201. [CrossRef]