applied sciences

MDPI

Article

Semantic Lidar-Inertial SLAM for Dynamic Scenes

Zean Bu, Changku Sun and Peng Wang *

State Key Lab of Precision Measuring Technology and Instruments, Tianjin University, Weijin Road, Tianjin 300072, China
* Correspondence: wang_peng@tju.edu.cn

Featured Application: This work is used for estimating ego-motion and mapping in dynamic scene.

Abstract: Over the past few years, many impressive lidar-inertial SLAM systems have been developed and perform well under static scenes. However, most tasks are under dynamic environments in real life, and the determination of a method to improve accuracy and robustness poses a challenge. In this paper, we propose a semantic lidar-inertial SLAM approach with the combination of a point cloud semantic segmentation network and lidar-inertial SLAM LIO mapping for dynamic scenes. We import an attention mechanism to the PointConv network to build an attention weight function to improve the capacity to predict details. The semantic segmentation results of the point clouds from lidar enable us to obtain point-wise labels for each lidar frame. After filtering the dynamic objects, the refined global map of the lidar-inertial SLAM sytem is clearer, and the estimated trajectory can achieve a higher precision. We conduct experiments on an UrbanNav dataset, whose challenging highway sequences have a large number of moving cars and pedestrians. The results demonstrate that, compared with other SLAM systems, the accuracy of trajectory can be improved to different degrees.

Keywords: lidar-inertial SLAM; semantic SLAM; semantic segmentation; dynamic scene

check for updates

Citation: Bu, Z.; Sun, C.; Wang, P. Semantic Lidar-Inertial SLAM for Dynamic Scenes. *Appl. Sci.* **2022**, *12*, 10497. https://doi.org/10.3390/app122010497

Academic Editors: Hang Guo, Marcin Uradzinski and You Li

Received: 5 July 2022
Accepted: 12 October 2022
Published: 18 October 2022

1. Introduction

Simultaneous localization and mapping (SLAM) [1,2] is of crucial significance for mobile robot navigation, micro aerial vehicles (MAVs), virtual reality (VR), and augmented reality (AR) since accurate pose estimation is fundamental for machine operation. In recent years, studies have proliferated regarding the theme of the fusion of inertial measurement units (IMU) with a single perceptual sensor. When the sensor is a lidar sensor, the system is called a lidar-inertial SLAM system [3]. Compared with other SLAM systems, the lidar-inertial SLAM system can acquire accurate ego-motion estimation and dense point cloud maps since the lidar sensor is invariant to changes in illumination and can provide distance measurements of the surrounding environments.

Over the past few years, many impressive lidar-inertial SLAM systems [4–9] have been developed and continue to perform well, such as LIO-mapping [6], LIO-SAM [8], SuMa [9], and so on. However, traditional lidar-inertial SLAM systems operate on the assumption that the scene is static. Most scenes in real-life, especially outdoor scenes, consist of dynamic environments. The feature points of the dynamic object are unstable, which affects the accuracy of localization and mapping. Furthermore, the typical lidar-inertial SLAM builds a map only with geometric information (points, lines, and planes), and lacks the semantic information of the surrounding scene that is needed by the robot to complete some advanced tasks.

In this paper, we focus on avoiding the moving objects in dynamic scenes by combining point cloud semantic segmentation with lidar-inertial SLAM. We propose a point cloud semantic segmentation network that imports an attention mechanism to the PointConv [10] network to improve the capacity to predict details. We define a graph constructed from reference points and their neighbors and input the relative coordinates of the neighbors combined with their features. We build attention weight functions to dynamically distribute the weights of neighbors. After filtering the points belonging to moving objects through a semantic segmentation network, stable static feature points are extracted, and a static local map is built. Then, joint non-linear optimization is adopted within a sliding window to obtain the final state of the system. We conduct experiments on the SemanticKITTI dataset [11] and UrbanNav dataset [12]. The results demonstrate that, compared with other SLAM systems, the accuracy of the estimated trajectory can be improved to different degrees compared with other methods, and the global map of our system is refined, becoming clearer.

2. Related Works

2.1. Lidar-Inertial SLAM

Lidar-inertial SLAM can be categorized into two methods: loosely-coupled methods and tightly-coupled methods. LOAM [4] is a classical loosely coupled algorithm that optimizes a large number of variables simultaneously in two frequencies. The high frequency part estimates the velocity of the lidar with a low fidelity. The low frequency part completes the point cloud matching and registration. LeGO-LOAM [5] optimizes based on LOAM. It is lighter in weight and leverages the presence of a ground plane in its segmentation and optimization steps. Tightly coupled algorithms are entering the mainstream for their high accuracy and strong robustness. LIO mapping [6] proposes a rotation-constrained refinement algorithm to align the lidar poses with a global map and performs well with an acceptable drift after long-term experiments. LINS [7] designs an iterated error-state Kalman filter (ESKF) to correct the estimated state recursively by generating new feature correspondences in each iteration. LIO-SAM [8] formulates lidar-inertial SLAM atop a factor graph, allowing for a multitude of relative and absolute measurements, including loop closures, to be incorporated from different sources as factors into the system.

2.2. Semantic SLAM

With the development of deep learning, many works have introduced semantic segmentation into visual-based or lidar-based SLAM systems. DS-SLAM [13] combines a semantic segmentation network with a moving consistency check method to reduce the impact of dynamic objects; thus, the localization accuracy is improved in dynamic environments. Berta et al. present a visual SLAM system (DynaSLAM) [14] that is built on ORB-SLAM2 [15]. It adds the capabilities of dynamic object detection and background inpainting, and DynaSLAM is robust in dynamic scenarios for monocular, stereo, and RGB-D configurations. SuMa++ [16] is an extension of SuMa [9] that consists of integrating semantic information to facilitate the mapping process. SA-LOAM [17] is a semantic-aided LiDAR SLAM with loop closure based on LOAM, which leverages semantics in odometry as well as loop closure detection. Wang et al. propose lidar-based SLAM under semantic constraints in dynamic environments [18]. They used a spatial attention network (SANet) to achieve the semantic segmentation of point clouds.

3. Our Method

In this paper, we define the transformation from frame a to frame b as $\mathbf{T}_a^b \in SE(3)$. The transformation contains translation vector $\boldsymbol{p}_a^b \in \mathbb{R}^3$ and rotation matrix $\mathbf{R}_a^b \in SO(3)$. In addition, the quaternion q_a^b is another expression of rotation for calculation. Feature points $\mathbf{F}_a$ in frame a can be transformed into frame b as feature points $\mathbf{F}_a^b$.

3.1. System Overview

The framework of the proposed semantic lidar-inertial SLAM system is illustrated in Figure 1. Our system is based on LIO-mapping system. IMU measurements $I_{i,j}$ are applied to predict the state of body $\tilde{\mathbf{T}}_{Lj'}^{Lj}$, which is utilized to de-skew the raw point clouds S_j of lidar data. At the same time, we use IMU pre-integration to obtain pre-integration terms $\Delta\boldsymbol{p}_{ij}$,

$\Delta \boldsymbol{v}_{ij}$, and Δq_{ij} for joint non-linear optimization. Then, de-skewed lidar frame $\overline{S}_j$ is input into the semantic segmentation network, and we can obtain static point cloud $\widetilde{S}_j$ after filtering out moving objects. After that, feature points $\mathbf{F}_{L\gamma}^{Lp}$ are extracted from several consecutive frames and they are utilized to build local map $\mathbf{M}_{Lo,i}^{Lp}$. At last, we use joint non-linear optimization to obtain the final estimation of the states $\mathbf{X}$ within a local sliding window.

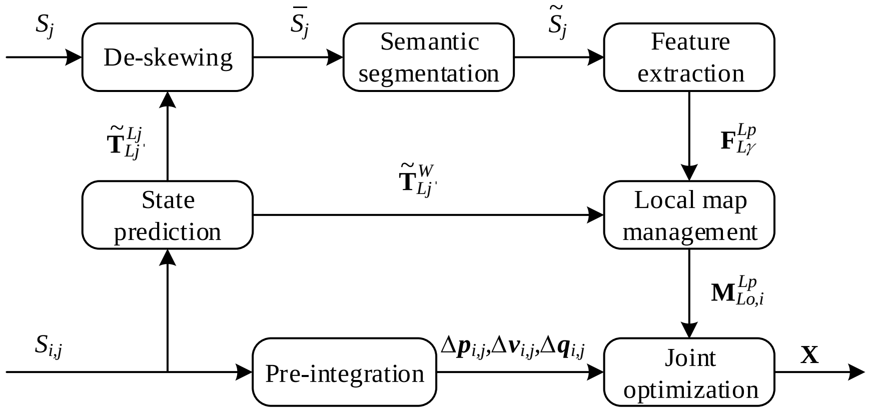

Figure 1. Overall framework of the proposed semantic lidar-inertial SLAM.

3.2. State Prediction and Pre-Integration

There are several IMU frames between two consecutive lidar frames. Within a short time, the state of body can be updated by:

$$\begin{aligned}
\boldsymbol{p}_j &= \boldsymbol{p}_i + \sum_{k=i}^{j-1}\left[\boldsymbol{v}_k \Delta t + \tfrac{1}{2}\boldsymbol{g}^W \Delta t^2 + \tfrac{1}{2}\mathbf{R}_k\left(\hat{\boldsymbol{a}}_k - \boldsymbol{b}_{a_k}\right)\Delta t^2\right] \\
\boldsymbol{v}_j &= \boldsymbol{v}_i + \boldsymbol{g}^W \Delta t_{ij}{}^2 + \sum_{k=i}^{j-1}\mathbf{R}_k\left(\hat{\boldsymbol{a}}_k - \boldsymbol{b}_{a_k}\right)\Delta t \\
q_j &= q_i \otimes \prod_{k=i}^{j-1}\delta q_k = q_i \otimes \prod_{k=i}^{j-1}\begin{bmatrix} \tfrac{1}{2}\Delta t\left(\hat{\boldsymbol{\omega}}_k - \boldsymbol{b}_{g_k}\right) \\ 1 \end{bmatrix}
\end{aligned} \tag{1}$$

where $\boldsymbol{p}$, $\boldsymbol{v}$, and q are the position, velocity, and orientation of the IMU body. $\boldsymbol{b}_a$ and $\boldsymbol{b}_g$ denote acceleration bias and gyroscope bias, respectively. $\hat{\boldsymbol{a}}_k$ and $\hat{\boldsymbol{\omega}}_k$ are the raw measurements of IMU at timestamp k.

The raw lidar data S_j are skewed because of the lidar sensor's motion during scanning. So, we de-skew each lidar frame S_j by linear interpolation of $\widetilde{\mathbf{T}}_{Lj'}^{Lj}$ and obtain de-skewed point cloud $\overline{S}_j$.

At the same time, the raw IMU outputs $\hat{\boldsymbol{a}}_k$ and $\hat{\boldsymbol{\omega}}_k$ can be converted to pre-integration measurements ($\Delta \boldsymbol{p}_{ij}$, $\Delta \boldsymbol{v}_{ij}$, and Δq_{ij}) by:

$$\begin{aligned}
\Delta \boldsymbol{p}_{ij} &= \mathbf{R}_i^T\left(\boldsymbol{p}_j - \boldsymbol{p}_i - \boldsymbol{v}_i \Delta t_{ij} - \tfrac{1}{2}\boldsymbol{g}^W \Delta t_{ij}^2\right) = \sum_{k=i}^{j-1}\left[\boldsymbol{v}_{ik}\Delta t + \tfrac{1}{2}\mathbf{R}(\Delta q_{ik})\left(\hat{\boldsymbol{a}}_k - \boldsymbol{b}_{a_k}\right)\Delta t^2\right] \\
\Delta \boldsymbol{v}_{ij} &= \mathbf{R}_i^T\left(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}^W \Delta t_{ij}\right) = \sum_{k=i}^{j-1}\left[\mathbf{R}(\Delta q_{ik})\left(\hat{\boldsymbol{a}}_k - \boldsymbol{b}_{a_k}\right)\Delta t\right] \\
\Delta q_{ij} &= q_i^{-1} \otimes q_j = \prod_{k=i}^{j-1}\delta q_k = \prod_{k=i}^{j-1}\begin{bmatrix} \tfrac{1}{2}\Delta t\left(\hat{\boldsymbol{\omega}}_k - \boldsymbol{b}_{g_k}\right) \\ 1 \end{bmatrix}
\end{aligned} \tag{2}$$

The pre-integration measurements are used for joint optimization.

3.3. Semantic Segmentation Network

We input $\overline{S}_j$ into sematic segmentation network to classify it point by point. The semantic segmentation network in our SLAM system is based on PointConv network. The network defines a 3D convolution operation called PointConv:

$$PointConv(S,W,F)_{xyz} = \iiint_{(\delta_x,\delta_y,\delta_z)\in G} S(\delta_x,\delta_y,\delta_z)W(\delta_x,\delta_y,\delta_z)F(x+\delta_x,y+\delta_y,z+\delta_z)d\delta_x\delta_y\delta_z \tag{3}$$

where $p = (x,y,z)$ is coordinate of reference point. $(\delta_x,\delta_y,\delta_z)$ is the relative coordinate of any neighbor point from p. $S(\delta_x,\delta_y,\delta_z)$ is the inverse density of point $(\delta_x,\delta_y,\delta_z)$. $W(\delta_x,\delta_y,\delta_z)$ is the weight function. In addition, $F(x+\delta_x,y+\delta_y,z+\delta_z)$ is the feature of point $(\delta_x,\delta_y,\delta_z)$. The main idea of PointConv is using the relative coordinates of K neighbor points to simulate weight function by multi-layer perceptron (MLP). However, it does not take the features of these neighbor points into consideration.

We import an attention mechanism to PointConv and build a graph $G = (V,E)$ as shown in Figure 2. Given a local point set $P = \{p_0,p_1,\ldots,p_K\}$, ($f = \{f_0,f_1,\ldots,f_K\}$ represents the feature of each point), $V \in 0,1,2,\ldots,K$ and $E \in |V|\times|V|$ denote the vertices and edges of the graph, respectively. For each point i, the weight between point i and its neighbor point j can be computed by multilayer perceptron:

$$\widetilde{a}_{ij} = MLP([\Delta p_{ij}||\Delta f_{ij}]) \tag{4}$$

where $\Delta p_{ij} = p_j - p_i$, $\Delta f_{ij} = MLP_g(f_j) - MLP_g(f_i)$, and MLP_g is also a multilayer perceptron, whose function is mapping a feature from one dimension to another. $||$ denotes the concatenation operation.

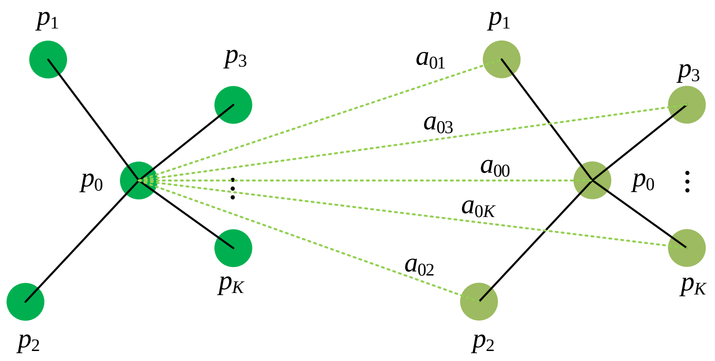

Figure 2. Illustration of attention mechanism on a reference point. The weight function is a weighted combination of the neighbors.

Therefore, attention weight of the n-th feature channel between point i and its neighbor point j can be computed by:

$$a_{ij,n} = \frac{\exp(\widetilde{a}_{ij,n})}{\sum_{l=0}^{K}\exp(\widetilde{a}_{il,n})} \tag{5}$$

The attention feature of point i can be described as:

$$f_i{}' = \sum_{j=0}^{K} a_{ij} * M_g(f_j) + b_i \tag{6}$$

where a_{ij}, M_g, and b_i are learnable. $*$ denotes the element-wise production.

So, for a given local point set $P = \{p_0,p_1,p_2,\ldots,p_K\}$, we can obtain the attention features through Equation (4). Let C_{in} and C'_{in} be the number of channels of input feature f_i and output feature $f_i{}'$. PointConv only takes the relative coordinate of each point $(p_i - p_0)$ as inputs to approximate weight function. We extend PointConv by concatenating attention feature f_i' with $(p_i - p_0)$, and adopt MLP1 to simulate attention weight function W, as shown in Figure 3. Therefore, the network can focus on the most relevant part of the neighbors to learn features. The entire process of attention PointConv network is shown in Figure 3.

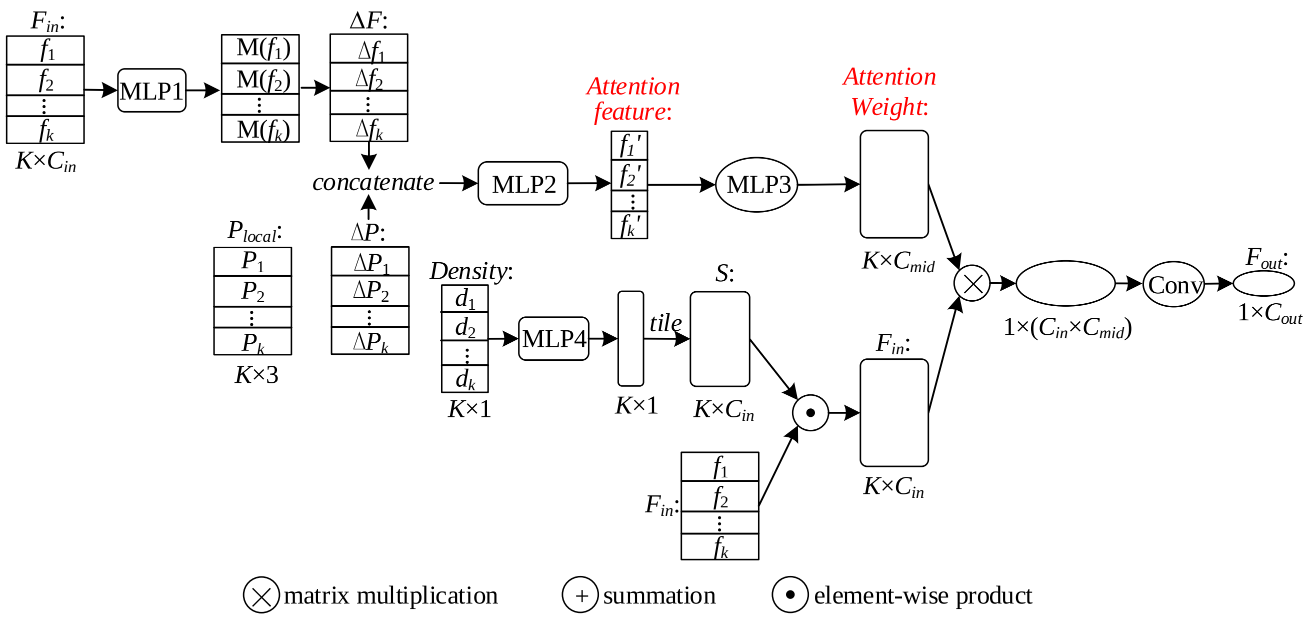

Figure 3. The whole process of attention PointConv network.

3.4. Feature Extraction and Local Map Management

De-skewed point cloud is segmented through attention PointConv network, and the points belonging to dynamic objects are removed. Then, we adopt lidar feature extraction [19] to select planar feature points, which are most likely on a plane.

Since one single lidar frame is too sparse for feature matching, we build a local map that consists of feature points of several lidar frames $\{o, \cdots, p, \cdots, i\}$, where o, p, and i are the first frame, the reference frame, and the last frame in the sliding window, respectively, as shown in Figure 4. For $\gamma \in \{o, \cdots, i\}$, feature points $\mathbf{F}_{L\gamma}$ are projected to reference frame p as $\mathbf{F}_{L\gamma}^{Lp}$ and all projected features in the sliding window build a local map $\mathbf{M}_{Lo,i}^{Lp}$. The frames before p are already optimized. For each point $\boldsymbol{x}^{Lp} \in \mathbf{F}_{L\alpha}^{Lp}$ ($\alpha \in \{p+1, \cdots, i, j\}$), we adopt KNN algorithm to find its K nearest points $\pi(\boldsymbol{x}^{Lp})$ in $\mathbf{M}_{Lo,i}^{Lp}$. Since these K points are most likely on the same plane, they have the plane constraint:

$$\boldsymbol{n}^T\boldsymbol{x} + d = 0,\ \boldsymbol{x} \in \pi\left(\boldsymbol{x}^{Lp}\right) \tag{7}$$

where we denote $m = [\boldsymbol{x}, \boldsymbol{n}, d] \in \mathbf{m}_{L\alpha}$. In this way, we formulate a relative constraint between the reference frame (n and d are defined in frame p) and the following frames ($\boldsymbol{x}$ is defined in reference α).

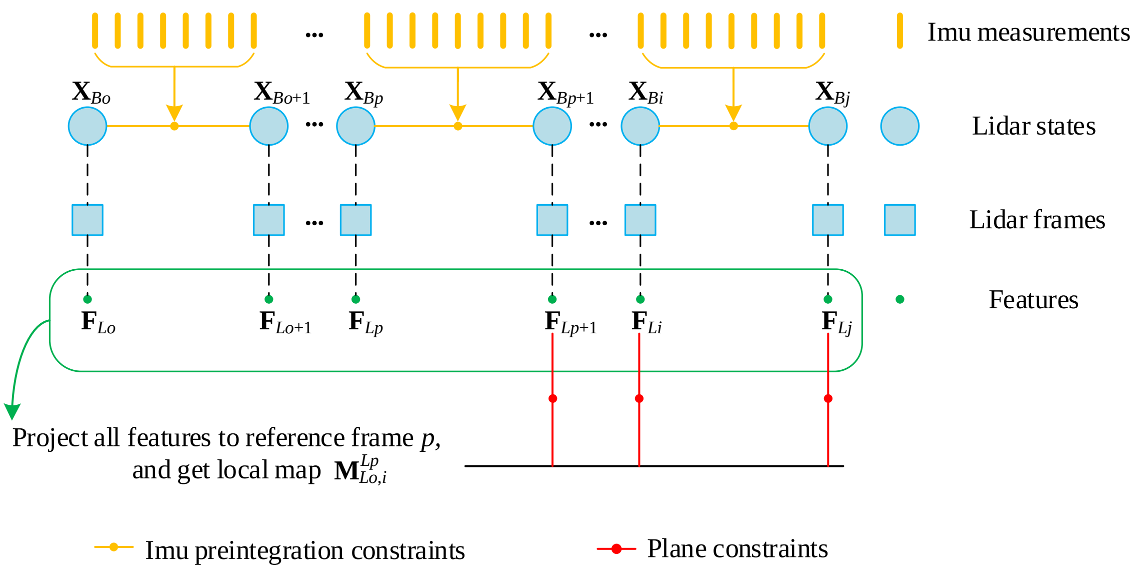

Figure 4. Illustration of sliding window and local map.

3.5. Joint Optimization

Equation (7) provides the constraints between the state of reference frame and the states of following frames. The transformation from the frames to be optimized ($\alpha \in \{p+1, \cdots, i, j\}$) to the reference frame p is computed by the chain rule:

$$\mathbf{T}_{L\alpha}^{Lp} = \mathbf{T}_L^B \mathbf{T}_{B\alpha}^W \mathbf{T}_{Bp}^{W\ -1} \mathbf{T}_L^{B-1} = \begin{bmatrix} \mathbf{R}_{L\alpha}^{Lp} & \boldsymbol{p}_{L\alpha}^{Lp} \\ 0 & 1 \end{bmatrix} \tag{8}$$

where $\mathbf{T}_L^B$ denotes the extrinsic parameters from lidar to IMU. $\mathbf{T}_{B\alpha}^W$ and $\mathbf{T}_{Bp}^W$ are the states of the body in frames α and p.

As the window slides, the reference frame also changes. For each $\alpha \in \{p+1, \cdots, i, j\}$ and $m = [\boldsymbol{x}, \boldsymbol{n}, d] \in \mathbf{m}_{L\alpha}$, the residual of Equation (7) can be represented by the distance from point to plane:

$$r_1\left(m, \mathbf{T}_{Lp}^W, \mathbf{T}_{L\alpha}^W, \mathbf{T}_L^B\right) = \boldsymbol{n}^T\left(\mathbf{R}_{L\alpha}^{Lp}\boldsymbol{x} + \boldsymbol{p}_{L\alpha}^{Lp}\right) + d \tag{9}$$

To obtain the states from frame p to frame j, we adopt a joint non-linear optimization and marginalization strategy. As shown in Figure 4, the number of states to be estimated is fixed in the window from $\mathbf{X}_{Bp}^W$ to $\mathbf{X}_{Bj}^W$. If new constraints appear, the window will include new states and marginalize old states. The entirety of the states in the window can be defined as:

$$\mathbf{X} = \left\{\mathbf{X}_{Bp}^W, \cdots, \mathbf{X}_{Bj}^W, \mathbf{T}_L^B\right\} \tag{10}$$

State $\mathbf{X}$ can be estimated by minimizing the whole cost function:

$$\min_{\mathbf{X}} \frac{1}{2}\left\{\sum_{\substack{m \in \mathbf{m}_{L\alpha} \\ \alpha \in \{p+1, \cdots, j\}}} \|r_1(m, \mathbf{X})\|^2 + \|r_2(\mathbf{X})\|^2 + \sum_{\beta \in \{p, \cdots, j-1\}} \left\|r_3\left(z_{\beta+1}^{\beta}, \mathbf{X}\right)\right\|^2\right\} \tag{11}$$

where $r_1(m, \mathbf{X})$, $r_2(\mathbf{X})$, and $r_3\left(z_{\beta+1}^{\beta}, \mathbf{X}\right)$ represent the residual for point-to-plane constraints, marginalization, and IMU pre-integration constraints, respectively. The Gauss–Newton method is used to solve the function above. $\|r_2(\mathbf{X})\|^2$ is calculated by Schur complement, which is described in [6], and $r_3\left(z_{\beta+1}^{\beta}, \mathbf{X}\right)$ is computed from the states and IMU pre-integration in [20].

4. Experiments

4.1. Semantic Segmentation Network Training

We use the SemanticKITTI dataset to train our attention PointConv network. The semanticKITTI dataset is a large-scale dataset based on the KITTI Vision Benchmark. It provides dense annotations for each individual scan of sequences from 00–10, which enables the usage of multiple sequential scans for semantic segmentation. Labels for the test set (sequences 11–21) are not provided. Figure 5 shows a single lidar frame with semantic annotations of the SemanticKITTI dataset, and it contains 20 classes including classes distinguishing non-moving and moving objects.

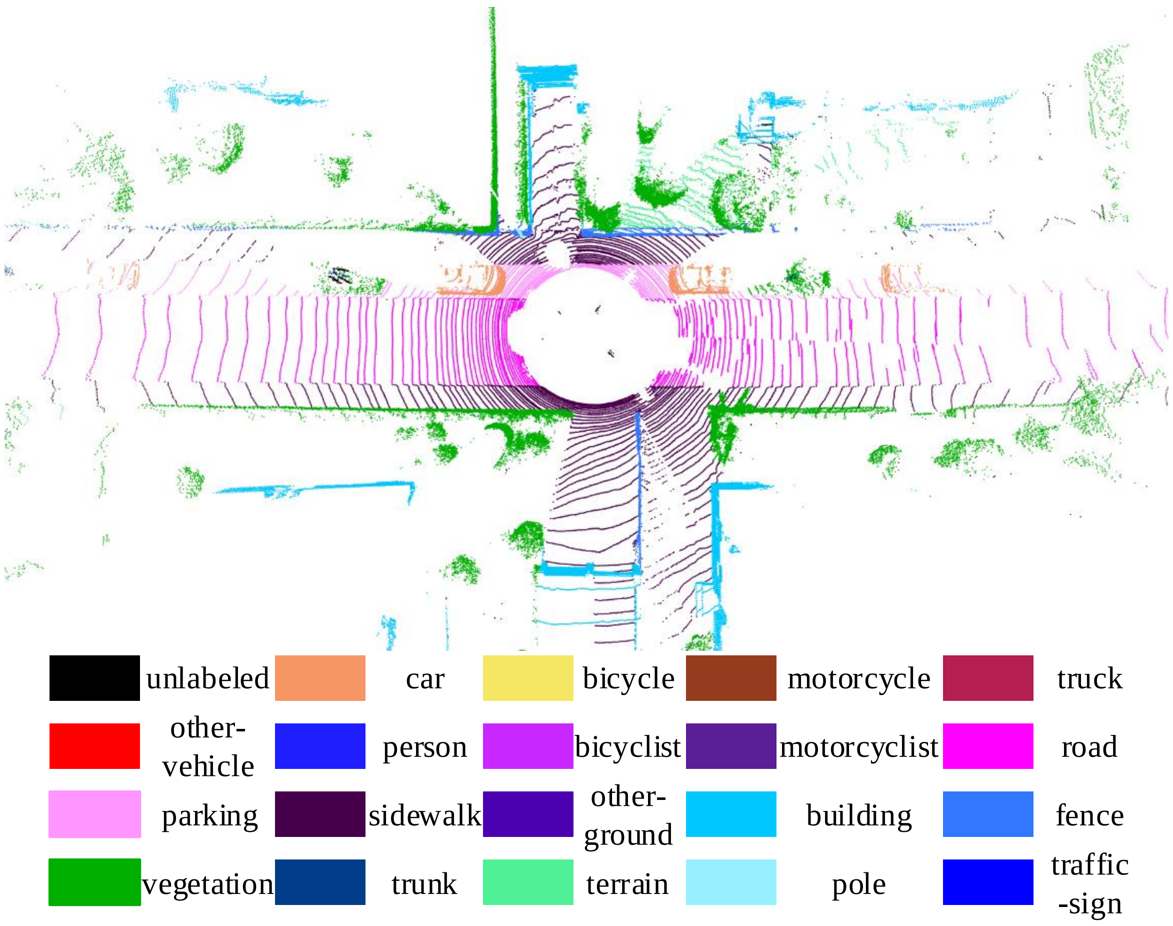

Figure 5. A single scan with labels of SemanticKITTI dataset.

We train our attention PointConv network with all the scans of sequences 00–10. In addition, in the next section, we will show the semantic segmentation results of our trained network.

4.2. Dynamic Object Removal

We use UrbanNav datasets to test the performance of the dynamic object removal. UrbanNav datasets are collected by mounting the data collection platform on a Honda Fit. The platform is equipped with Velodyne HDL-32E lidar, an Xsens Mti-10 IMU, a monocular camera, and a SPAN-CPT (for ground truth). Two lidar scan samples are shown in Figure 6a. Each lidar frame is input to our attention PointConv network to obtain point-wise labels, as shown in Figure 6b. Then, we filter out the points belonging to dynamic labels and obtain static point clouds, as in Figure 6c. From the visualization of the point cloud samples, we can see that the annotations of the points are basically correct, and the removal of the dynamic objects is effective.

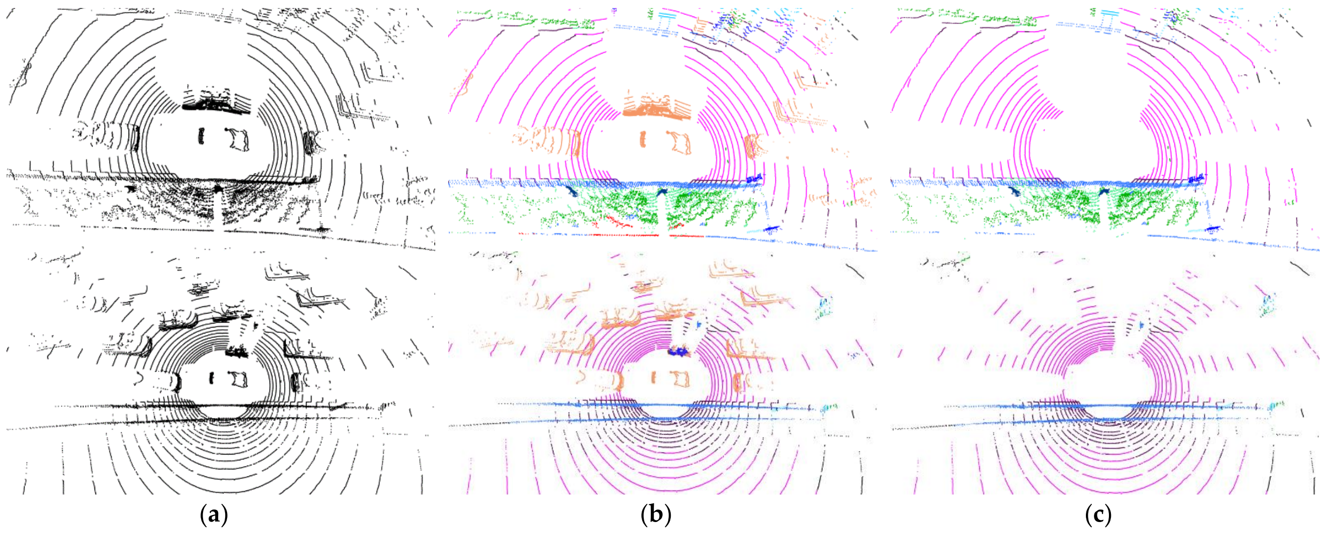

Figure 6. (**a**) the original point clouds; (**b**) semantic segmentation results with point-wise labels; (**c**) point clouds after filtering out dynamic points.

4.3. Pose Estimation Comparison

In this section, we compare the accuracy and robustness of the proposed method with the current state-of-the-art lidar-inertial SLAM methods, including LIO-mapping, LINS, and LIO-SAM. LINS is a filter-based tightly coupled method, and LIO-SAM is a tightly coupled method based on nonlinear optimization.

We also use UrbanNav datasets to compare the performance of different methods. Specifically, we used the Urban2019 dataset and Urban2020 dataset. The former was gathered in a typical urban canyon of Hong Kong featuring high-rising buildings and numerous dynamic objects in the evening. The latter was gathered in a low-urbanization area in Hong Kong with some dynamic objects present during in the day, as shown in Figure 7.

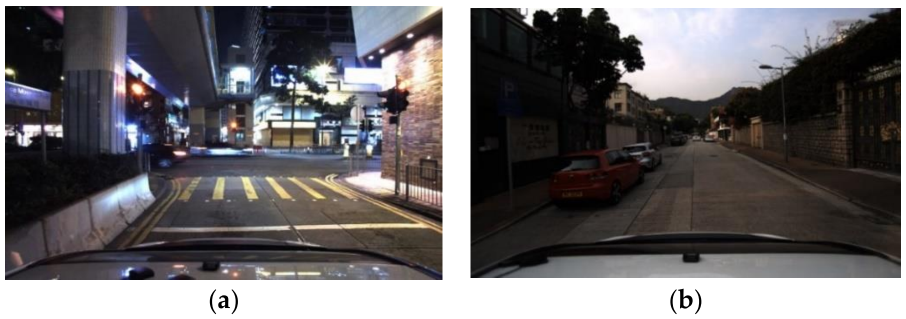

Figure 7. Sample images of datasets: (**a**) Urban2019 dataset; (**b**) Urban2020 dataset.

In this experiment, the datasets start and end at the origin. The GPS measurements of the datasets serve as the ground truth. The estimated trajectories of various methods applied to the Urban2019 dataset are shown in Figure 8. LIO-mapping performs well at the beginning of the dataset, but it drifts later in the trajectory and the rotation of LIO-mapping deviates at the curve. This is mainly caused by false feature points belonging to dynamic objects. The trajectories of LINS and LIO-SAM have a similar degree of misalignment, and they both have a slight deviation from the start point to the end point. The overall trajectory of the proposed method performs well.

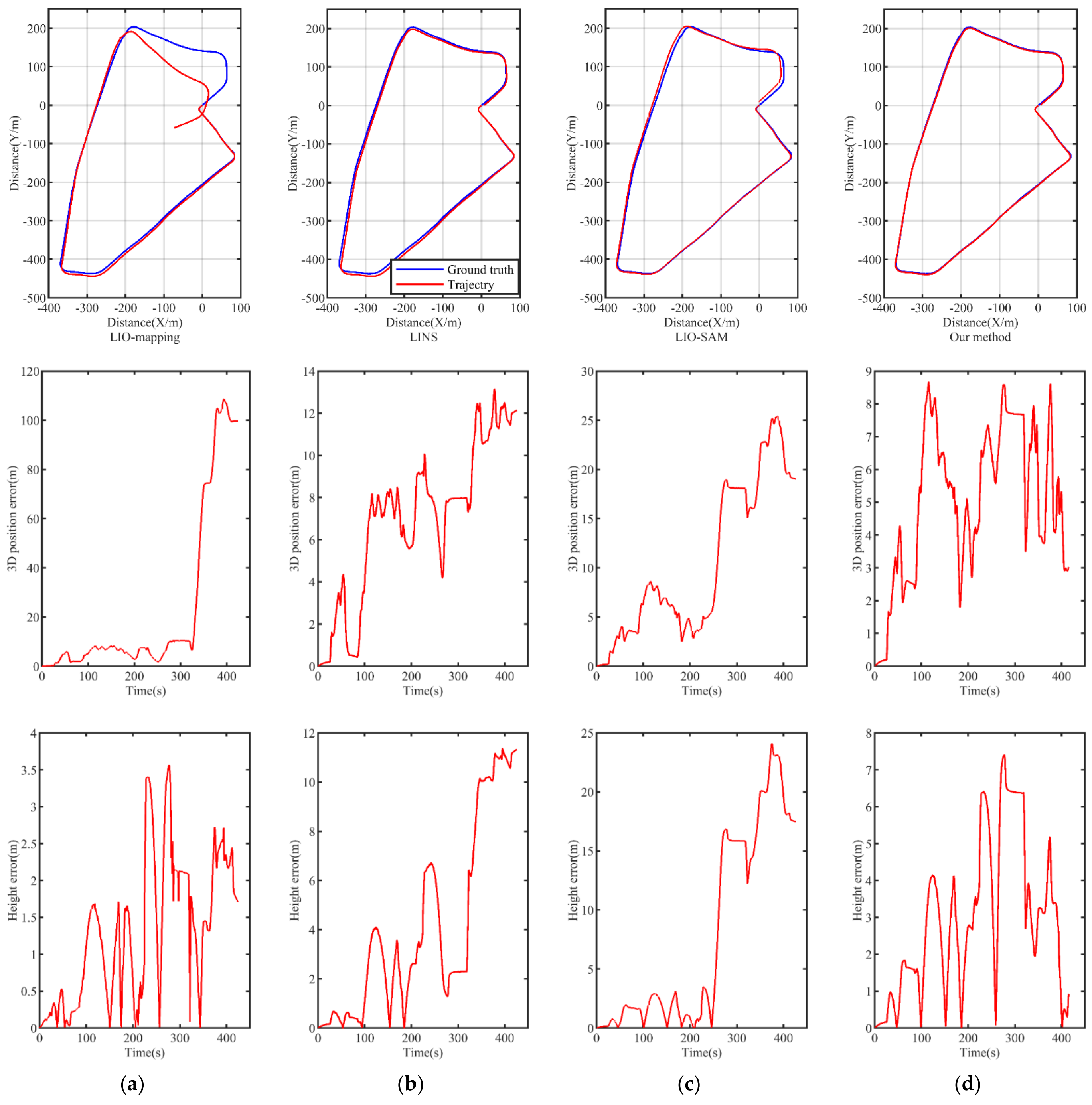

Figure 8. Estimated trajectories, 3D position errors, and height errors of different methods applied to Urban2019 dataset: (**a**) LIO−mapping; (**b**) LINS; (**c**) LIO−SAM; (**d**) our method.

Furthermore, we display the 3D position errors and height errors of the different methods with respect to time in Figure 8. The 3D position errors and height errors of the three other methods increase dramatically at one certain point. The 3D position errors of the LIO-mapping approach reach up to more than 100 m, which means that the algorithm has totally failed. By contrast, the 3D position error and height error of our method fluctuate within a certain section since errors accumulate more slowly after moving dynamic objects.

The estimated trajectories of the Urban2020 dataset are demonstrated in Figure 9. As shown, all the methods perform better than they did for the Urban2019 dataset, as there are fewer dynamic objects on the road and the distance is shorter. Even so, the proposed method performs better than the LIO-mapping approach with respect to the trajectory. In addition, the 3D position errors and height errors are also lower.

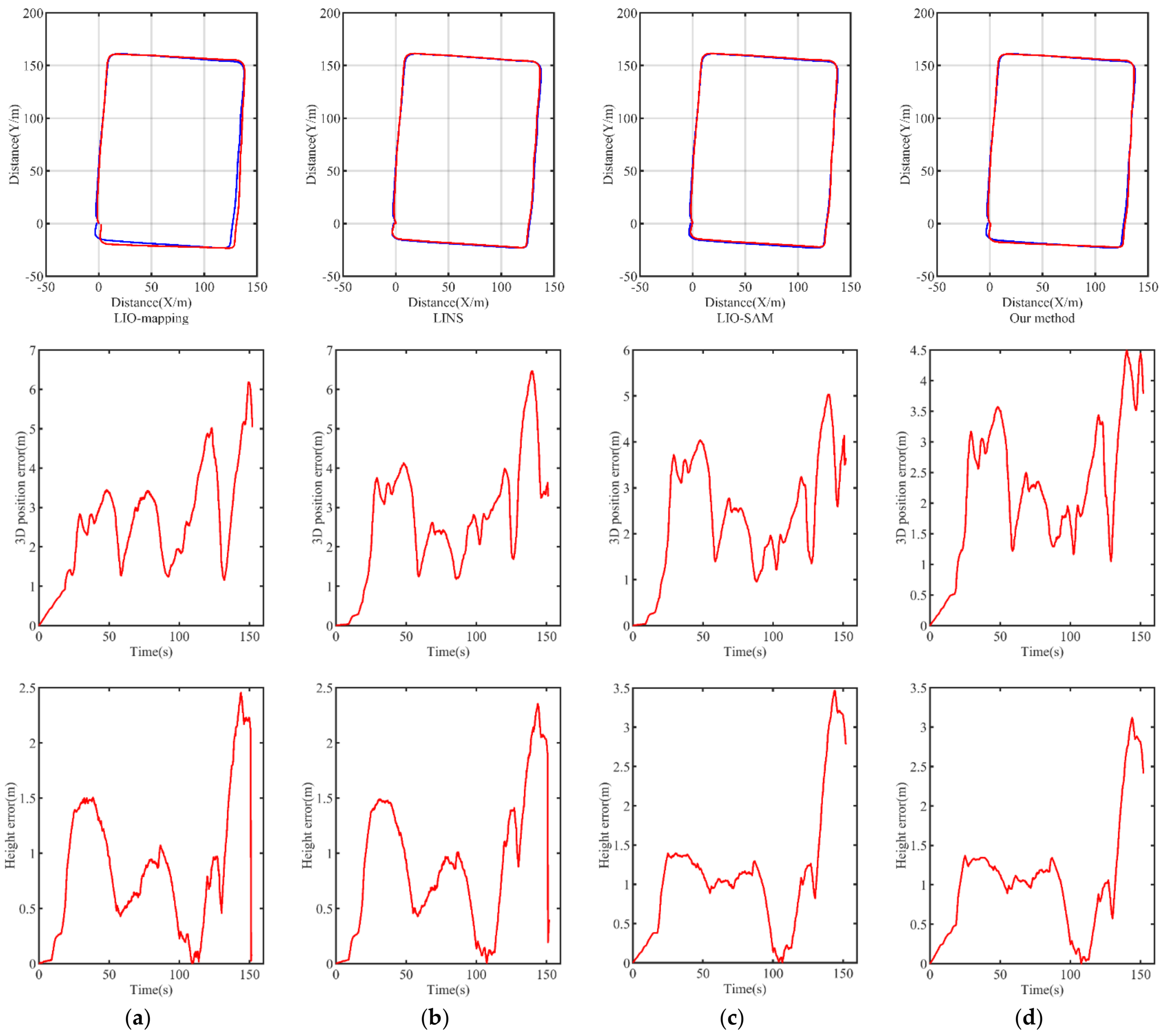

Figure 9. Estimated trajectories, 3D position errors, and height errors of different methods in Urban2020 dataset: (**a**) LIO−mapping; (**b**) LINS; (**c**) LIO−SAM; (**d**) our method.

Table 1 displays the end-to-end translation errors, end-to-end rotation errors, and RMSE errors with regard to the ground truth. Our method achieves the lowest RMSE error on both datasets. Furthermore, our method is superior to LIO-mapping with respect to end-to-end translation and rotation errors.

Table 1. The end-to-end translation errors, end-to-end rotation errors, and RMSE errors of different methods on both datasets.

Dataset	Error Type	LIO-Mapping	LINS	LIO-SAM	Our Method
Urban2019	Translation (m)	95.473	13.258	19.274	**5.673**
	Rotation (degree)	22.829	24.704	16.942	**14.554**
	RMSE (m)	43.018	7.863	13.069	**5.543**
Urban2020	Translation (m)	5.030	**1.276**	3.513	3.713
	Rotation (degree)	14.626	25.092	**6.556**	7.670
	RMSE (m)	3.107	3.039	2.672	**2.625**

4.4. Runtime Performance Evaluation

In this section, we will show the average runtime of each module of our algorithm. The algorithm was tested with a 2.6-GHz i7-9720H CPU and an Nvidia GeForce GTX 1660 Ti GPU in Ubuntu 18.04. We downsampled the point cloud of each lidar frame by using a voxelgrid filter whose leaf size is 0.5 m. The average runtime of each module is presented in Table 2, and all the modules run on different threads.

Table 2. Average runtime of each module (ms).

Dataset	Prediction	Semantic Segmentation	Odometry	Laser Mapping
Urban2019	0.00928	106.8	126.5	201.9
Urban2020	0.01198	122.6	127.9	252.8

4.5. Global Map Comparison

Finally, we test the performance of the global map built using our method. The global maps of the proposed method applied to the Urban2019 dataset and Urban2020 dataset are illuminated in Figure 10a,b). It is shown that the global maps of our method are basically correct. To compare the maps built by the LIO-mapping method and our method, we focus on several parts of the map in Figure 11. Figure 11a corresponds to LIO mapping and Figure 11b to our method. The figures demonstrate that our method can eliminate dynamic objects including vehicles of all kinds, which are meaningless for a global map. Furthermore, Figure 11a shows many obvious trails of the speeding vehicles on the roads, while the same roads in Figure 11b are clean, because the points belonging to moving cars have been filtered out in each lidar frame.

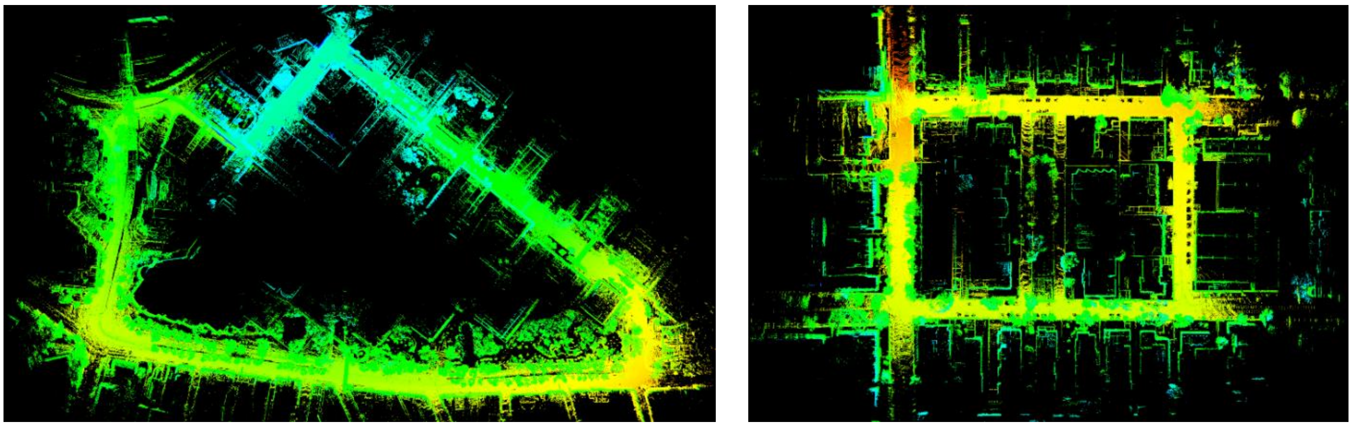
(**a**) (**b**)

Figure 10. Global map built using the proposed method: (**a**) Urban2019 dataset; (**b**) Urban2020 dataset.

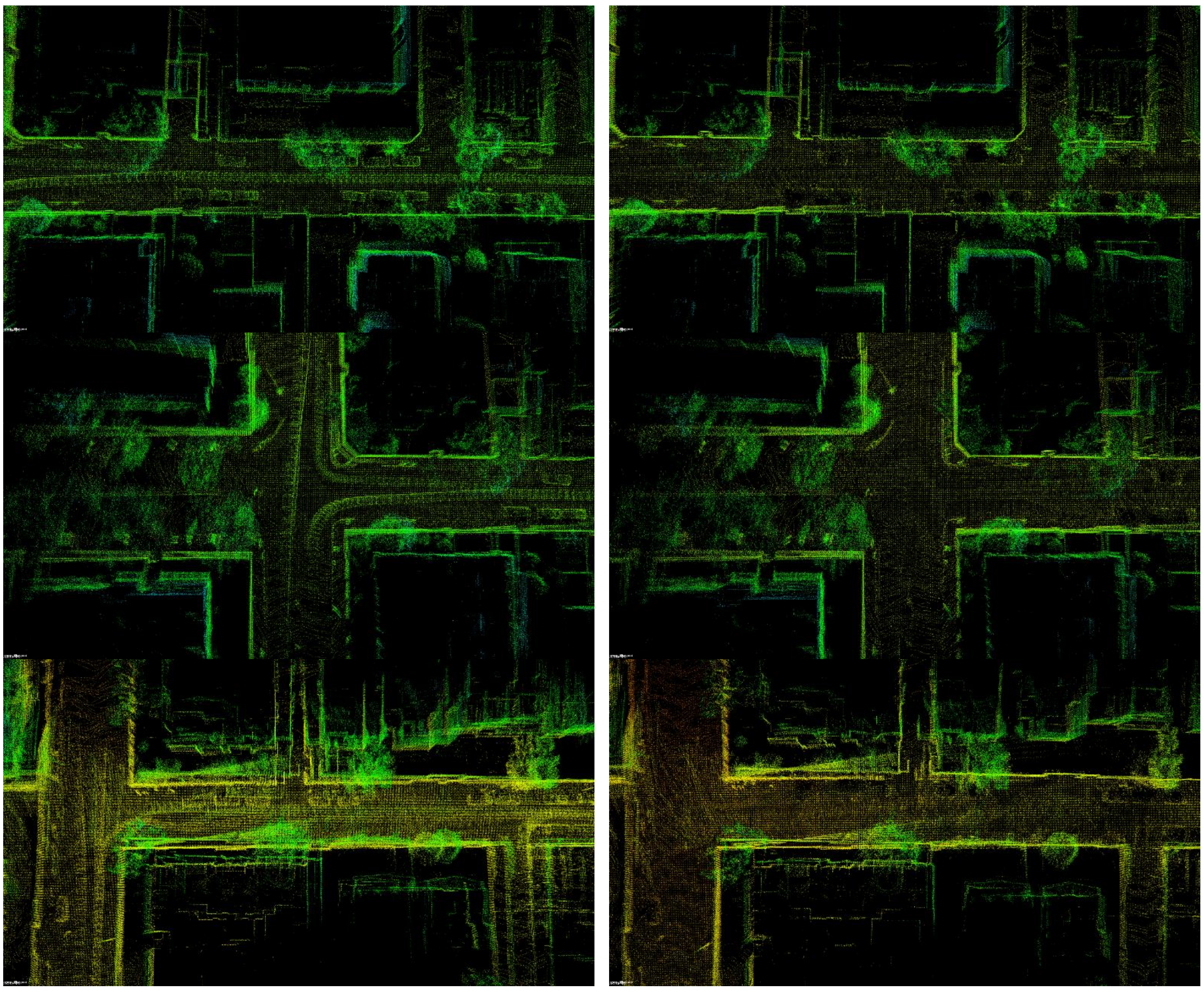
(**a**) (**b**)

Figure 11. Detailed parts of global map created using LIO mapping and our method: (**a**) corresponds to LIO-mapping method; (**b**) corresponds to our method.

5. Conclusions

Most SLAM systems work on the assumption that the environment is static. In this paper, we have proposed a semantic lidar-inertial SLAM system for dynamic scenes. We imported an attention mechanism to the PointConv network to elevate the feature-learning capability of the network. The LIO-mapping system was combined with an attention PointConv semantic segmentation network, and the points belonging to dynamic objects were removed in each lidar frame. The stable static feature points were extracted, and a static local map was built to perform ego-motion estimation. The experiments demonstrate that our method can decrease the errors of the estimated trajectories and build a clear global map of dynamic scenes.

Author Contributions: Conceptualization, Z.B.; methodology, Z.B.; software, Z.B.; validation, Z.B.; formal analysis, Z.B.; writing—original draft preparation, Z.B.; writing—review and editing, C.S. and P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset analyzed during the current study was derived from SemanticKITTI (http://www.semantic-kitti.org/) (accessed on 1 January 2022) and UrbanNavDataset (https://github.com/weisongwen/UrbanNavDataset) (accessed on 1 January 2022).

Acknowledgments: The authors thank the authors of the dataset for making it available online. Furthermore, they would like to thank the anonymous reviewers for their contribution towards enhancing this paper.

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

References

1. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
2. Grisetti, G.; Kümmerle, R.; Stachniss, C.; Burgard, W. A Tutorial on Graph-Based SLAM. *IEEE Intell. Transp. Syst. Mag.* **2010**, *2*, 31–43. [CrossRef]
3. Ji, K.; Chen, H.; Di, H.; Gong, J.; Xiong, G.; Qi, J.; Yi, T. CPFG-SLAM: A robust simultaneous localization and mapping based onLIDAR in off-road environment. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018.
4. Zhang, J.; Singh, S. Loam: Lidar odometry and mapping in real-time. In Proceedings of the 2014 Robotics: Science and Systems, Berkeley, CA, USA, 12–16 July 2014.
5. Shan, T.; Englot, B. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
6. Ye, H.; Chen, Y.; Liu, M. Tightly Coupled 3D Lidar Inertial Odometry and Mapping. In Proceedings of the 2019 IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019.
7. Qin, C.; Ye, H.; Pranata, C.E.; Han, J.; Zhang, S.; Liu, M. LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May 2020–31 August 2020.
8. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021.
9. Behley, J.; Stachniss, C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In Proceedings of the 2018 Robotics: Science and Systems, Pittsburgh, PA, USA, 26–30 June 2018.
10. Wu, W.; Qi, Z.; Li, F. PointConv: Deep Convolutional Networks on 3D Point Clouds. In Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, NY, USA, 15–20 June 2019.
11. Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; Gall, J. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. *arXiv* **2020**, arXiv:1904.01416.
12. Hsu, L.-T.; Kubo, N.; Wen, W.; Chen, W.; Liu, Z.; Suzuki, T.; Meguro, J. UrbanNav: An Open-Sourced Multisensory Dataset for Benchmarking Positioning Algorithms Designed for Urban Areas. In Proceedings of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+), St. Louis, MO, USA, 20–24 September, 2021.
13. Yu, C.; Liu, Z.; Liu, X.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
14. Bescos, B.; Fcil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [CrossRef]
15. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
16. Chen, X.; Milioto, A.; Palazzolo, E.; Giguere, P.; Behley, J.; Stachniss, C. SuMa++: Efficient LiDAR-based semantic SLAM. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019.
17. Li, L.; Kong, X.; Zhao, X.; Li, W.; Wen, F.; Zhang, H.; Liu, Y. SA-LOAM: Semantic-aided LiDAR SLAM with Loop Closure. *arXiv* **2021**, arXiv:2106.11516.
18. Wang, W.; You, X.; Zhang, X.; Chen, L.; Zhang, L.; Liu, X. LiDAR-Based SLAM under Semantic Constraints in Dynamic Environments. *Remote Sens.* **2021**, *13*, 3651. [CrossRef]
19. Bosse, M.; Zlot, R. Continuous 3d scan-matching with a spinning 2d laser. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009.
20. 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]