applied sciences

MDPI

Article

Real-Time Drivable Region Mapping Using an RGB-D Sensor with Loop Closure Refinement and 3D Semantic Map-Merging

ChangWan Ha [1], DongHyun Yang [1], Gicheol Wang [2], Sung Chang Kim [2] and HyungGi Jo [1,*]

[1] Division of Electronic Engineering, Jeonbuk National University, Jeonju 54896, Republic of Korea; cwha0212@jbnu.ac.kr (C.H.); ekdh7456@jbnu.ac.kr (D.Y.)
[2] Edge Computing Application Service Research Section, Electronics and Telecommunications Research Institute, Gwangju 61012, Republic of Korea; gcwang@etri.re.kr (G.W.); sungchang@etri.re.kr (S.C.K.)
* Correspondence: hygijo@jbnu.ac.kr

Abstract: Drivable region maps, created using a visual sensor, are essential for autonomous navigation because off-the-shelf maps do not reflect contemporary real-world conditions. This study presents a large-scale drivable region mapping system that is capable of capturing large-scale environments in real-time, using a single RGB-D sensor. Whereas existing semantic simultaneous localization and mapping (SLAM) methods consider only accurate pose estimation and the registration of semantic information, when loop closure is detected, contemporaneous large-scale spatial semantic maps are generated by refining 3D point clouds and semantic information. When loop closure occurs, our method finds the corresponding keyframe for each semantically labeled point cloud and transforms the point cloud into adjusted positions. Additionally, a map-merging algorithm for semantic maps is proposed to address large-scale environments. Experiments were conducted on the Complex Urban dataset and our custom dataset, which are publicly available, and real-world datasets using a vehicle-mounted sensor. Our method alleviates the drift errors that frequently occur when the agents navigate in large areas. Compared with satellite images, the resulting semantic maps are well aligned and have proven validity in terms of timeliness and accuracy.

Keywords: visual simultaneous localization and mapping (SLAM); mapping; map-merging; semantic segmentation

check for updates

Citation: Ha, C.; Yang, D.; Wang, G.; Kim, S.; Jo, H. Real-Time Drivable Region Mapping Using an RGB-D Sensor with Loop Closure Refinement and 3D Semantic Map-Merging. *Appl. Sci.* **2024**, *14*, 11613. https://doi.org/10.3390/app142411613

Academic Editor: Pedro Couto

Received: 7 November 2024
Revised: 2 December 2024
Accepted: 9 December 2024
Published: 12 December 2024

Copyright: © 2024 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 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

When autonomous mobile platforms such as autonomous vehicles and mobile robots navigate external environments, the difference in road representation between off-the-shelf maps and actual drivable conditions can lead to potential confusion and accidents.These incidents include scenarios where robotic agents descend stairs, fall off cliffs, or navigate into unexpected construction sites. Therefore, real-time estimates of drivable regions (surfaces where a mobile robot can travel, such as roads and sidewalks) are crucial for safe and reliable autonomous navigation systems. Drivable region mapping has two key requirements. The first is simultaneous localization and mapping (SLAM) [1–3], which allows agents to determine their pose and the structure of the surrounding environment. The second is semantic segmentation [4–6], which involves understanding what constitutes the surrounding environment. Based on these two technologies, semantic mapping not only maps 3D space but also its meaning. The most basic map representation method for expressing 3D space is to use point clouds, and, thus, the semantic point cloud map contains a semantic label for each point. When considering mapping a larger area, two key factors must be considered for large-scale semantic mapping. First, it is essential to create accurate onsite maps in real time. Second, multiple onsite maps must be effectively merged through postprocessing.

For the first key factor, the most important component in creating accurate onsite maps is loop closure detection. Loop closure refers to the identification of a loop when a

specific building or area is circumnavigated and the agent returns to the visited area. When a loop is detected, a globally matched edge is created. This step adjusts the entire recorded trajectory to reduce errors, following the principles of global matching. In addition, the map is transformed according to the trajectory changes. Unsuccessful loop creation or a map in need of modification potentially hinders the collection of large volumes of data for map construction. Moreover, the absence of semantic information, such as roads or sidewalks, can lead to inaccurate path planning. Thus, studies on semantic SLAM [7–9] are also being conducted to address this issue. Semantic SLAM extracts semantic information from images and uses it within SLAM. However, in these methods, loop closure is either not applied, or the segmented information is not refined after loop closure.

For the second key factor, when dealing with large-scale tasks, an additional map-merging process is needed. Map-merging combines two or more submaps into a single global map [10–16]. Using partially constructed maps to generate a large-scale map is highly efficient. Map-merging methods vary significantly depending on the type of maps, such as grid, topological, feature-based, or point cloud maps. This research adopts a point cloud registration method [17–19]. In map-merging via these registration methods, there is a risk of failure in complex environments because it relies solely on the geometric information of the whole point cloud. Additionally, it requires using all the points within the submaps, resulting in higher computational costs.

To resolve the two aforementioned issues, our proposed method considers two stages. The first stage is the onsite drivable region mapping stage, and the second stage is the map-merging stage. In the first stage, drivable region maps are created in real time. Additionally, the map-merging stage allows us to combine multiple segmented drivable region maps into a large-scale map. The main goal of the first stage is to estimate drivable regions using both visual SLAM and semantic segmentation. Our approach addresses the misalignment issue arising from semantic SLAM. The loop closure refinement is implemented by considering the semantic label encoded error function. Consequently, our method prioritizes minimizing the gap between semantic information and world/agent data, including maps and six degrees of freedom (6-DOF) poses. In the second stage, for large-scale environments, the map-merging stage is required to construct a global map, i.e., the multiple drivable region maps are integrated into a single map. During the map-merging stage, our proposed merging method is performed based on overlapping regions specified by semantic information. This approach avoids the drift problem that frequently occurs in large-scale drivable region mapping. In these two stages, a large-scale drivable region map is obtained.

In summary, this study proposes a new method to create a large-scale map of drivable regions through semantic map generation and map-merging algorithms. Initially, input images captured by RGB-D sensors undergo semantic segmentation to distinguish between roads, vehicles, and non-drivable areas. We improved the accuracy of the semantic location information through loop closure refinement and map-merging. Finally, leveraging the camera pose derived from the SLAM system, the drivable regions are visualized in real time. To the best of our knowledge, this study represents the first instance of using map-merging and semantic visual SLAM to construct a large-scale semantic map. The main contributions of our proposed method are as follows:

- Semantic visual SLAM and map-merging methods are proposed to identify drivable areas in large-scale environments. Our method allows for the visualization of drivable areas that traditional visual SLAM cannot depict;
- A loop closure refinement for semantic information is developed to build seamless semantic 3D maps. This approach allows for the simultaneous use of semantic segmentation and visual SLAM results to create a unified semantic map, minimizing the gap between semantic information and world/agent data;
- A 3D point cloud registration-based semantic map-merging is performed not only on 3D point descriptors but also on semantic labels to create a large-scale map. Our proposed method uses fewer points in overlapping areas, thereby accelerating the map-merging process;

- Our proposed system was tested in real-world experiments by attaching a single RGB-D camera to a vehicle. We constructed a large-scale drivable map from multiple segmented datasets, demonstrating higher accuracy than existing methods.

2. Related Work

Visual SLAM [20–25] is a computational method for building or updating a map of an unknown environment and estimating the current pose using visual sensors. Visual SLAM extracts visual features and optimizes poses and map points through correspondence matching by calculating geometric errors. Feature-based SLAM systems, such as VINS-MONO [24] or ORB-SLAM [25], extract key features from images, find correspondences between images, and estimate the camera pose and the 3D information of the environment. However, this process inevitably leads to drift errors over time. To address this issue, most SLAM systems employ pose graph optimization [26]-based loop closure techniques to minimize accumulated errors. Alternatively, Elhashash et al. [27] proposed the use of satellite images as an external data source to correct drift, while Arun et al. [28] utilized Wi-Fi data for error correction. Similarly, previous works [29,30] have proposed SLAM methods that address trajectory drift by leveraging external data rather than relying on loop closure. In contrast, this study adopts an RGB-D camera-based approach with loop closure as a key component. RGB-D cameras are commonly used as visual sensors for 3D mapping because they provide more stable localization in SLAM processes and face fewer challenges with scale ambiguity compared to monocular or stereo cameras.

For exploiting semantic information of the scene, semantic segmentation has been extensively studied in the field of computer vision. Prior research has investigated recognizing drivable regions from segmented images with deep learning-based methods. The results generated from the training of semantic segmentation [31–35] or instance segmentation [36–38], which are well-known tasks for representing semantics, can be utilized for drivable region detection. Recently, Xu et al. [39] employed the semantic segmentation method to determine the real-time drivable region from an image. It applies the concept of a proportional–integral–derivative (PID) controller to a deep learning model and is designed for real-time segmentation.

Gosala et al. [40] proposed a self-supervised learning system trained on monocular images for classification, which performs semantic mapping for roads and objects through bird's-eye-view transformation. Some studies [41–43] have introduced loop closure methods that leverage semantic information. While these approaches demonstrate the potential of semantic data in mapping and localization, they primarily focus on consecutive frame-to-frame semantic mapping. As a result, a gap between the 3D map and the semantic map appears during loop closure since the frame-to-map correspondence has not been considered.

Regarding the second key factor, *map-merging process*, for large-scale drivable mapping, the descriptor of features extracted from point clouds and the registration method that uses it are important. Point feature histogram (PFH) [44] is a method that finds four features using the normal vector of a plane obtained through principal component analysis (PCA) and the point itself by considering one point and its k-neighbors. Fast point feature histogram (FPFH) [45] is an improved algorithm based on PFH. The original PFH has a complexity of $O(nk^2)$ with n points and k neighboring points. FPFH addresses the complexity issue by first calculating the relationships between points and neighbors and then assigning weights, reducing the complexity to $O(nk)$ and enabling faster computations. Radius-based surface descriptor (RSD) [46] focuses on the shape of surfaces among features. It classifies surfaces into types, such as plane, edge, corner, cylinder, and sphere. Signatures of a histogram of orientations (SHOT) [47] collect information based on the direction and curvature of the object's surface in point cloud data, creating a histogram that represents unique features.

For 3D point clouds registration, iterative closest point (ICP) [48] is a well-known algorithm that finds correspondences between target and source point clouds, calculates ro-

tation and translation based on corresponding points, and iteratively refines the alignment. Generalized iterative closest point (G-ICP) [49] is an algorithm that combines ICP and point-to-plane ICP. G-ICP uses point correspondences and also incorporates the normal vectors of planes from the target and source point clouds. However, existing registration methods using point cloud descriptors lack a comprehensive consideration of semantic information for each point or feature descriptors. In this paper, we propose a method that considers semantic information to find overlapped region and effectively merges multiple local semantic maps.

3. Methodology

3.1. Problem Formulation

As shown in Figure 1, our proposed method consists of two steps: drivable region mapping and map-merging.

In the drivable region mapping process, as the RGB-D camera navigates an unknown environment, the series of input images, denoted by $\mathcal{I} = \{I_t\}_{t=1:T}$ from time $t = 1$ to $t = T$, represents the collection of images captured. As I_t is an RGB-D image, it can be converted to point clouds $P_t = \{\mathbf{p}_i\}_{i=1:N}$, where N denotes the number of points. A camera pose that corresponds to P_t is denoted as $\mathbf{x}_t = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & \mathbf{1} \end{bmatrix} \in \mathrm{SE}(3)$, where $\mathbf{R} \in \mathrm{SO}(3)$ represents orientation and $\mathbf{t} \in \mathbb{R}^3$ represents position. A camera pose can also be represented with quaternion vectors as $\mathbf{x}_t = \left(\mathbf{x}_t^p, \mathbf{x}_t^q\right)^T$, where $\mathbf{x}_t^p = (x_t \ y_t \ z_t)$ and $\mathbf{x}_t^q = \left(q_{x,t} \ q_{y,t} \ q_{z,t} \ q_{w,t}\right)$ are the position and quaternion vectors, respectively. The camera trajectory $\mathcal{X} = \{\mathbf{x}_t\}_{t=1:t}$ can be obtained using visual SLAM [25] methods. Semantic segmentation outputs labels $\xi_t \in \{1, \cdots, cl\}$ for each pixel of an image I_t, where cl indicates the number of classes. The main goal of this step is to build fused point clouds $\widetilde{P}_t$, which are composed of P_t and $\xi_t \in \{road, sidewalk\}$. A 3D point $\mathbf{p}_i$ in the camera coordinates is projected into a 2D pixel point $\mathbf{u}_i$ on the image plane by

$$\begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \begin{bmatrix} \pi(\mathbf{p}_i) \\ 1 \end{bmatrix} = \frac{1}{z}\mathbf{K}\mathbf{p}_i \tag{1}$$

where $\mathbf{K}$ is the intrinsic camera matrix and $\pi(\cdot)$ is a projection function. Conversely, 2D point $\mathbf{u}$ and a class label $\widetilde{\xi}$ are converted into 3D point clouds with a depth value of each point [50].

$$\mathbf{p}_i = \pi^{-1}(\mathbf{u}_i) = D(\mathbf{u}_i)\mathbf{K}^{-1}\begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} \tag{2}$$

where $D(\mathbf{u}_i)$ is the depth value of image pixel $\mathbf{u}_i$. Thus, $\widetilde{P}_t = \left\{P_t, \widetilde{\xi}\right\} \in \mathbb{R}^4$ is composed of point clouds and each point's class label. The world coordinate point cloud map $\widetilde{P}_t^W$ can be generated by the camera pose $\mathbf{R}$ and $\mathbf{t}$.

$$\widetilde{P}_t^W = \left\{\mathbf{R}\widetilde{P}_t + \mathbf{t}, \widetilde{\xi}\right\} \tag{3}$$

The 3D feature point clouds map $\mathcal{M}_j$. Here, j indicates the j-th submap index, which is constructed by combining the SLAM results with the semantic point clouds.

$$\mathcal{M}_j \leftarrow \widetilde{P}_t^W \cup \mathcal{M}_j \tag{4}$$

The next step is map-merging. The separately built maps $\mathcal{M}_{j=1:N}$ across various regions are combined into a single map $\widetilde{\mathcal{M}}$. To make $\widetilde{\mathcal{M}}$, the collection of feature points $\mathcal{F}_j = \left\{\mathbf{f}_{k,j}\right\}_{k=1:K}$ is extracted using FPFH [45] from $\mathcal{M}_j$. To integrate maps developed at various times and locations, the 3D point-registration technique was used [51].

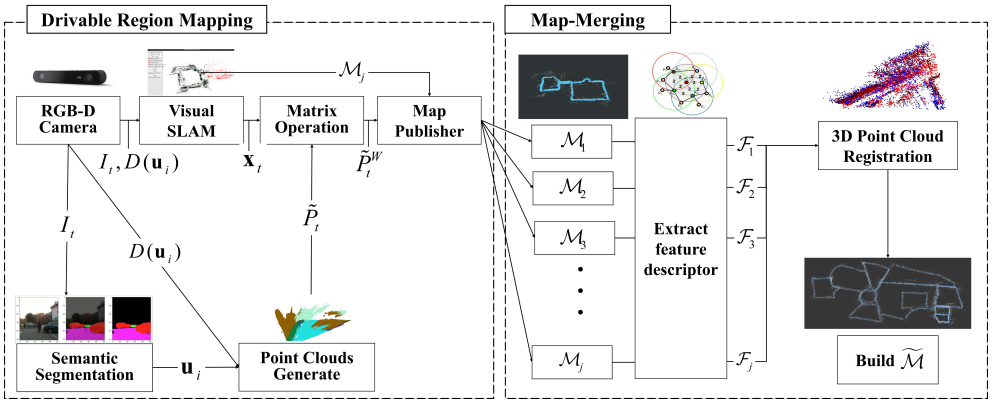


Figure 1. Schematic overview of the proposed method. It consists of precise drivable region mapping (**left**) and map-merging for large-scale mapping (**right**).

3.2. System Overview

Drivable Region Mapping: For drivable region mapping, both semantic information and the 3D map are required. We employed deep learning-based semantic segmentation method and visual SLAM with a single RGB-D Camera such as Asus Xtion and Intel Realsense. When the camera captures I_t, $\widetilde{P}_t$ is generated via deep-learning-based semantic segmentation results $\mathbf{u}_i$ and $D(\mathbf{u}_i)$. PIDNet [39] is used for semantic segmentation, and $\widetilde{P}_t$ represents a collection of points in the camera coordinates. The results of visual SLAM include $\mathcal{X} = \{\mathbf{x}_t\}_{t=1:t}$ and $\mathcal{M}_j$. ORB-SLAM3 is used for visual SLAM. Subsequently, $\widetilde{P}_t$ is transformed to $\widetilde{P}_t^W$ according to $\mathbf{x}_t$ and combined with $\mathcal{M}_j$. This sequence is repeated to create $\mathcal{M}_j$. When this transformation occurs, our method effectively addresses the mismatch between the 3D map and semantic information.

Map-merging: In this step, two $\mathcal{M}_j$ are merged. First, the overlapping region is specified, and the corresponding point clouds are extracted. Then, $\mathcal{F}_j$ is extracted from the extracted point clouds using FPFH [45]. The KNN and KDTree [52] algorithms are subsequently used to find correspondences and perform 3D point cloud registration via TEASER++ [51]. By applying the transformation matrix from TEASER++, the two maps are merged to generate a global map. This process is repeated for all $\mathcal{M}_j$, creating final global map $\widetilde{\mathcal{M}}$.

3.3. Drivable Region Mapping

The results of semantic segmentation $\mathbf{u}_i$ with depth value $D(\mathbf{u}_i)$ enable the transformation of pixels to $\mathbf{p}_i$. For every $\mathbf{u}_i$ in I_t, $\widetilde{P}_t$ is obtained, and Figure 2 shows the result of Equation (2). Additionally, $\widetilde{P}_t$ is in the camera coordinates. The generation of excessive point clouds from I_t leads to computational memory constraints, which prevent the direct utilization of $\widetilde{P}_t$. Therefore, the number of point clouds should be reduced through a voxel grid filter. For the simplicity, we denote $\widetilde{P}_t$ as filtered point clouds.

Integrated Semantic Segmentation and 3D Mapping: As we mentioned in system overview section, the semantic segmentation result $\widetilde{P}_t$ and the visual SLAM results ($\mathcal{M}_j$, $\mathcal{X}$) are obtained. To consolidate $\widetilde{P}_t$ into a single $\mathcal{M}_j$, it is necessary to transform camera coordinates $\widetilde{P}_t$ into world coordinates, following Equation (3). The update is subsequently performed according to Equation (4). However, during this update, many duplicated $\widetilde{P}_t^W$ can be generated at the same location, which degrades the computing performance. To prevent this, KDTree [52] is utilized to remove duplicate $\widetilde{P}_t^W$ within a specified threshold.

Additionally, loop closure is necessary. When a loop is detected in ORB-SLAM3, the system enters the loop closure phase. In the loop closure phase, $\mathcal{X}$ is modified through pose graph optimization and global bundle adjustment (BA). Consequently, $\mathcal{M}_j$ is also changed; thus, $\widetilde{P}_t^W$ must be adjusted accordingly.

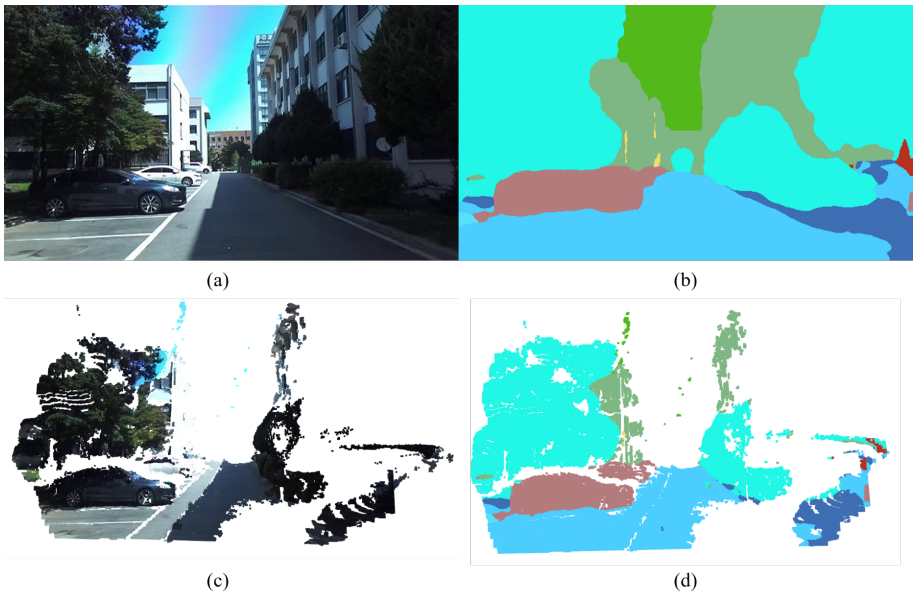


Figure 2. (**a**) RGB image. (**b**) The resulting image of semantic segmentation. (**c**) 3D point clouds created from RGB image and depth image. (**d**) 3D point clouds generated from the result of semantic segmentation and depth image.

According to the full BA of ORB-SLAM3, the optimization problem is as follows:

$$\{\mathbf{x}_t, \mathcal{M}_j\} = \arg\min_{\mathbf{x}_t, \mathcal{M}_j} E_{point} \tag{5}$$

$$E_{point} = \sum_{k \in \mathcal{K}} \sum_{j \in \chi_k} \rho(E(k,j)) \tag{6}$$

where $E(k,j) = \left\| \mathbf{u}^j - \pi(\mathbf{R}_k P^j + \mathbf{t}_k) \right\|_{\Sigma}^2$ is the re-projection error between matched 3D points $P^j \in \mathbb{R}^3$ in keyframes $\mathcal{K}$ and key points $\mathbf{u}^j \in \mathbb{R}^2$ with $j \in \chi_k$, the set of all matches. Additionally, $\pi(\cdot)$ is the projection function mentioned in Equation (1), and $\rho(\cdot)$ is the robust Huber cost function [53]. Further, the label information of $\tilde{P}_t^W$ is considered. The label-encoded error function E_c, which represents how correctly the current class label aligns with existing point cloud labels, is defined as

$$E_c = \sum_k \sum_j \left[\mathcal{L}(\mathcal{M}_j(P_t^*)) = \mathcal{L}\left(\mathbf{R}_k P^j + \mathbf{t}_k\right) \right] \tag{7}$$

where $[\cdot]$ indicates the Iverson bracket. Point $\mathcal{M}_j(P_t^*)$ is the nearest point from $\mathbf{R}_k P^j + \mathbf{t}_k$ within the existing 3D point map $\mathcal{M}_j$. Additionally, $\mathcal{L}(\cdot)$ indicates the label value of the input points. Therefore, the final BA is formulated as a nonlinear least-squares optimization problem as follows:

$$\left\{\mathbf{x}_t, \widetilde{\mathcal{M}}\right\} = \arg\min_{\mathbf{x}_t, \widetilde{\mathcal{M}}} E_{point} + E_c \tag{8}$$

Finally, the output of BA $\{\mathbf{R}, \mathbf{t}\}$ is used to adjust $\tilde{P}_t^W$.

$$\mathcal{M}_j \leftarrow \left(\mathbf{R}\tilde{P}_t^W + \mathbf{t}\right) \cup \mathcal{M}_j \tag{9}$$

As shown in Figure 3, the keyframe with the corresponding $\widetilde{P}_t^W$ is matched in real time. If I_t is not a keyframe, $\widetilde{P}_t^W$ will match with the closest keyframe before $t = T$. When the loop closure phase begins, changes to the keyframe pose are applied to $\widetilde{P}_t^W$. This ensures that

the drivable region created through semantic segmentation helps create a more accurate map during the loop closure phase.

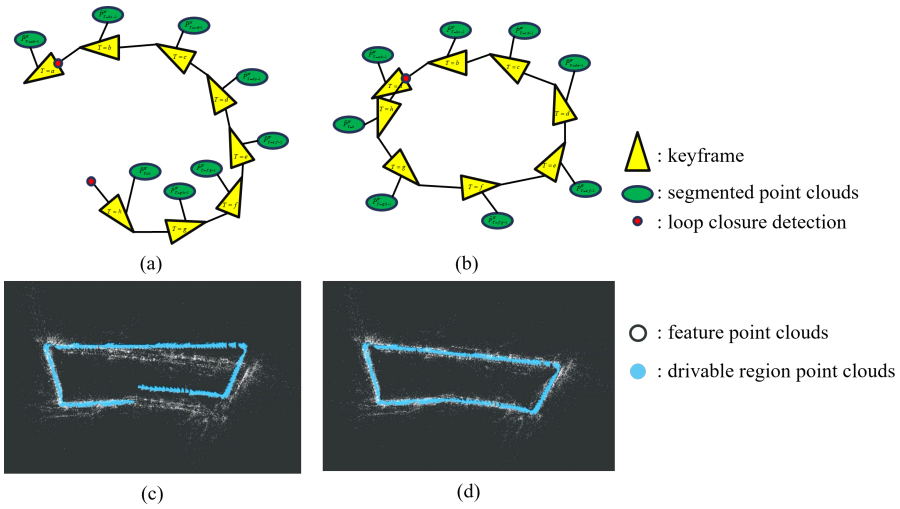


Figure 3. A schematic example of loop closure refinement and comparison before and after applying refinement to the drivable region map. (**a**) Schematic diagram before loop closure. Despite being at the same location, misalignment occurs owing to drift. (**b**) Schematic diagram after loop closure. Loop closure resolves issues caused by drift error and realigns the frame pose and point clouds. (**c**) Drivable region mapping result before loop closure. White point clouds represent feature points extracted from images, whereas sky blue point clouds denote the drivable region. (**d**) Drivable region mapping result after loop closure. White point clouds represent feature points extracted from images, whereas sky blue point clouds denote the drivable region.

3.4. Map-Merging

Algorithm 1 is the pseudo-code of the proposed map-merging algorithm. The map-merging process starts with preparing the data. Figure 4a,b shows example maps $\mathcal{M}_1$ and $\mathcal{M}_2$. Before map-merging , information about overlapping buildings or roads between multiple maps is needed. Overlapping regions are selected before data collection and considered in the 3D mapping process. When selecting overlapping regions, the following priorities are established: (1) a building-centric region, (2) a region where both roads and buildings coexist, and (3) a road-centric region. Additionally, when selecting an overlapping area, it is better to consider the point cloud of buildings rather than focusing on roads, especially in the case of straight segments. The overlapping region $P_{OR,j} = \left\{\mathbf{p}_{l,j}\right\}_{l=1:L}$ is extracted using Algorithm 2.

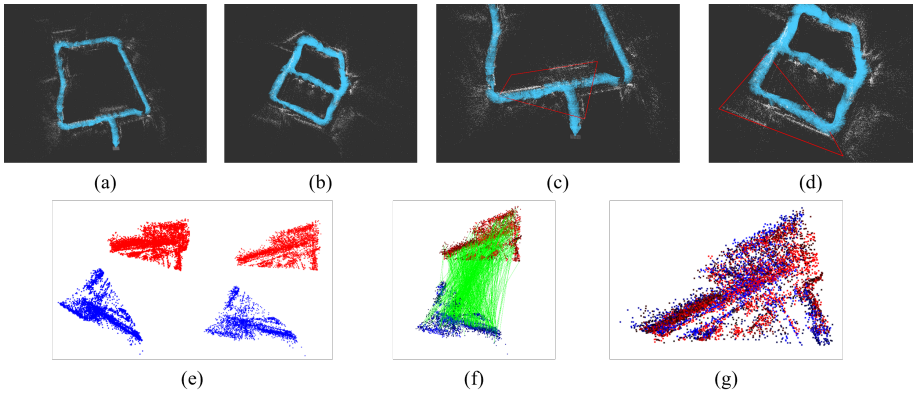


Figure 4. Map merging: (**a**) Model 1; (**b**) Model 2; (**c**) A red square denotes the overlapping region of Model 1; (**d**) A red square denotes the overlapping region of Model 2; (**e**) Voxel downsampling; (**f**) FPFH and find correspondence; (**g**) Point cloud registration via TEASER++.

Algorithm 1 Map-merging

Require: Input point clouds $P_{OR,a}$, $P_{OR,b}$, map $\mathcal{M}_a$, $\mathcal{M}_b$
Ensure: Output global map $\widetilde{\mathcal{M}}$
1 $\hat{P}_{OR,a}$, $\hat{P}_{OR,b} \leftarrow$ Downsampled $P_{OR,a}$, $P_{OR,b}$ by a voxel grid filter
2 $\mathcal{F}_a$, $\mathcal{F}_b \leftarrow$ FPFH($\hat{P}_{OR,a}$, $\hat{P}_{OR,b}$)
3 $\{C_i\}_{i=1:N_m} \leftarrow$ Estimate correspondences
4 **for** $i = 1 : N_m$ **do**
5 $\quad \mathbf{R}, \mathbf{t} \leftarrow \arg\min_{\mathbf{R},\mathbf{t}} \left\{ \sum_i \left\| n_i \cdot (\mathbf{T} \cdot \hat{P}_{OR,a} - \hat{P}_{OR,b}) \right\|^2 \right\}$
6 **end for**
7 $\hat{\mathcal{M}}_a = \mathbf{R}\mathcal{M}_a + \mathbf{t}$
8 $\widetilde{\mathcal{M}} \leftarrow \hat{\mathcal{M}}_a \cup \mathcal{M}_b$
return $\widetilde{\mathcal{M}}$

Algorithm 2 Extracting point clouds in the overlapping region

Require: input map $\mathcal{M}_j$, overlapping region's edge points $P_{Edge} = \{\mathbf{p}_i\}_{i=1:N}$
Ensure: overlapping region point clouds $P_{OR,j} = \left\{\mathbf{p}_{l,j}\right\}_{l=1:L}$
1 **for** $\mathcal{M}_j$ **do**
2 $\quad$ Set $counter = 0$, $\mathbf{p_{N+1}} = \mathbf{p_1}$
3 $\quad$ **for** $i = 1 : N$ **do**
4 $\quad\quad$ **if** point of $\mathcal{M}_j$'s x-axis between $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ and y-axis lower than $\max(\mathbf{p}_i, \mathbf{p}_{i+1})$ and y-axis above line by $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ **then**
5 $\quad\quad\quad$ $counter = counter + 1$
6 $\quad\quad$ **end if**
7 $\quad$ **end for**
8 $\quad$ **if** $counter$ = odd number **then**
9 $\quad\quad$ point of $\mathcal{M}_j \in P_{OR,j}$
10 $\quad$ **end if**
11 **end for**
return $P_{OR,j}$

In Algorithm 2, lines 1 to 7, each point in $\mathcal{M}_j$ compares the x-axis and y-axis with $P_{Edge} = \{\mathbf{p}_i\}_{i=1:N}$. P_{Edge} is a set of edge points in the overlapping region. It is specified and represented in the form of an N-polygon. A polygonal shape is selected to best preserve the information of the overlapping region, and z-axis information is not considered in this process. For $i = 1 : N$, if the point's x-axis is between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and the y-axis is lower than $\max(\mathbf{p}_i, \mathbf{p}_{i+1})$ but above the line created by $\mathbf{p}_i$, $\mathbf{p}_{i+1}$, then the counter is incremented by 1. In lines 8 to 11, if the counter is an odd number, then the point is added to $P_{OR,j}$.

The red box in Figure 4c,d indicates an overlapping region $P_{OR,1}$, $P_{OR,2}$. In Algorithm 1, line 1 downsamples $P_{OR,1}$, $P_{OR,2}$ via a voxel grid filter. Figure 4e is an example of this. This process helps to prevent point cloud registration from being dominated by road points and provides computational advantages. We found that a voxel size of 0.1 was the best parameter empirically, as it preserves the features of the point clouds without significant degradation. In Algorithm 1 line 4, $\mathcal{F}_a$, $\mathcal{F}_b$ are extracted from downsampled point clouds $\hat{P}_{OR,a}$, $\hat{P}_{OR,b}$ by FPFH and, in line 3, correspondences $\{C_i\}_{i=1:N_m}$ are estimated via the KNN algorithm. Figure 4f shows the estimated correspondences and their connections from $\mathcal{F}_6$ and $\mathcal{F}_8$. In Algorithm 1 lines 4 to 6, the registration step estimates the relative transformation between two point clouds: $\mathbf{R}, \mathbf{t} \leftarrow \arg\min_{\mathbf{R},\mathbf{t}} \left\{ \sum_i \left\| n_i \cdot (\mathbf{T} \cdot P_{OR,a} - P_{OR,b}) \right\|^2 \right\}$, which aligns $P_{OR,a} = \mathbf{T} \cdot P_{OR,b}$. The 3D point cloud registration tool TEASER++ was used in this process. In line 7, the result of the estimation is aligned with $\hat{\mathcal{M}}_a$. This process can be observed in Figure 4g. If registration does not proceed correctly, $P_{OR,1}$, $P_{OR,2}$ are redefined. Finally, in line 8, $\hat{\mathcal{M}}_a$ is merged with $\mathcal{M}_b$ to produce $\widetilde{\mathcal{M}}$.

4. Experiments and Results

4.1. Datasets

For semantic segmentation, 4093 labeled images—3475 from the Cityscapes [54] dataset and 618 from a custom dataset—were used for training. The Cityscapes dataset provided sufficient labels for vehicles and roads. The custom dataset focused on capturing roads near Jeonbuk National University, especially pathways with trees and stairs. For annotation, the Computer Vision Annotation Tool was utilized.

For the drivable region mapping, a total of eight paths were captured, as shown in Figure 5. Data collection was performed using a Zed2i RGB-D camera installed on a vehicle. Further details about the dataset are provided in Table 1. Additionally, the Complex Urban dataset [55], a benchmark dataset taken in an urban environment, was also used to evaluate the performance of the proposed method. Among the datasets, the Urban 28 sequence was used, as shown in Figure 6. Table 2 provides further details about the dataset. Unfortunately, the Complex Urban dataset does not contain depth images. Therefore, SLAM was performed in stereo mode, and the drivable region was confirmed through depth estimation. In the depth-estimation process, the Depth Anything [56] model was utilized. Through the evaluation of these two datasets, we demonstrated that the proposed method is effective in large-scale environments.

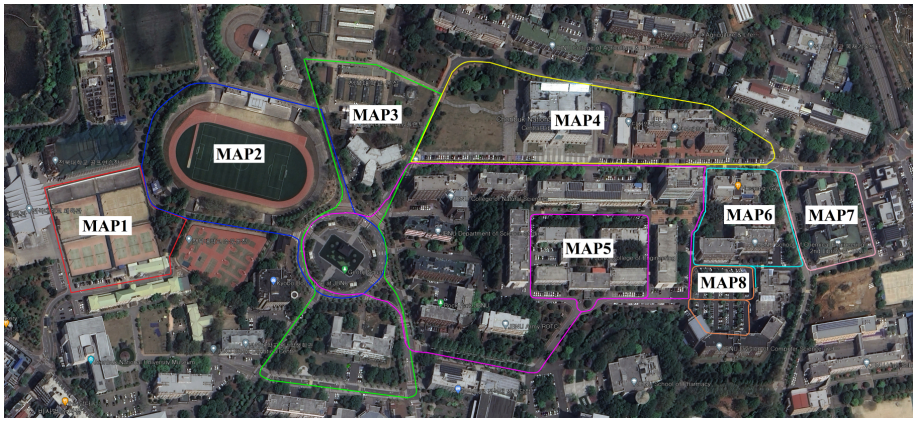


Figure 5. Satellite image of Jeonbuk National University. Red, blue, green, yellow, pink, sky blue, light pink, and brown represent the paths of map1 to map8, respectively.

Table 1. The 3D map information.

	Map1	Map2	Map3	Map4	Map5	Map6	Map7	Map8
Length (m)	491	754	1062	911	1482	531	374	510
Area (m^2)	15,947	37,046	38,689	33,237	76,572	11,966	9215	5030
Images	2472	3119	4755	3447	6423	3361	1495	3987
Time (s)	164	211	326	231	434	168	106	156

Table 2. Complex Urban dataset results.

	Map1	Map2	Map3	Merged Map
Length (m)	3179	2219	5861	11,285
Images	5673	3563	11,671	19,745
RMSE	5.674	11.20	14.75	18.79

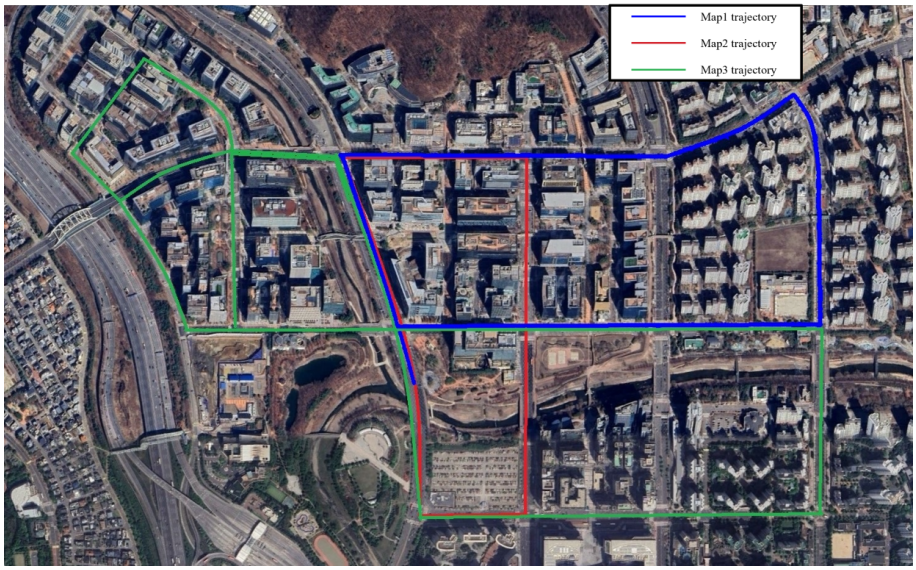


Figure 6. Urban 28 dataset [55] satellite image and divided course. The course was manually divided into three sections and displayed on a satellite image. The satellite images were captured via Google Maps. The lines on the satellite images were drawn manually. The trajectories for maps 1, 2, and 3 are shown in blue, red, and green.

4.2. Drivable Region Mapping Results

PIDNet [39] has three variants based on model size: PIDNet_S, PIDNet_M, and PIDNet_L. Among these, PIDNet_L and PIDNet_M were considered unsuitable for real-time usage due to their higher number of parameters, so PIDNet_S was selected. Compared to other models, the PIDNet_S model achieves a final accuracy of 92.3% and a mean intersection over union (mIoU) of 80.2%, as shown in Table 3. According to Equation (10), accuracy is calculated by dividing the number of correctly classified pixels by the total number of pixels in the image, meaning 92.3% of the pixels were correctly classified. With a frame rate of 45.2 FPS, PIDNet_S demonstrated optimal performance for real-time applications. Although DDRNet_23 offers slightly higher accuracy and supports real-time execution, we prioritized mIoU as the key evaluation metric. For this reason, PIDNet_S was selected. In the future, the semantic segmentation model may be upgraded to a higher-performing alternative.

Table 3. Training results of the semantic segmentation model using the Cityscapes dataset and our custom dataset.

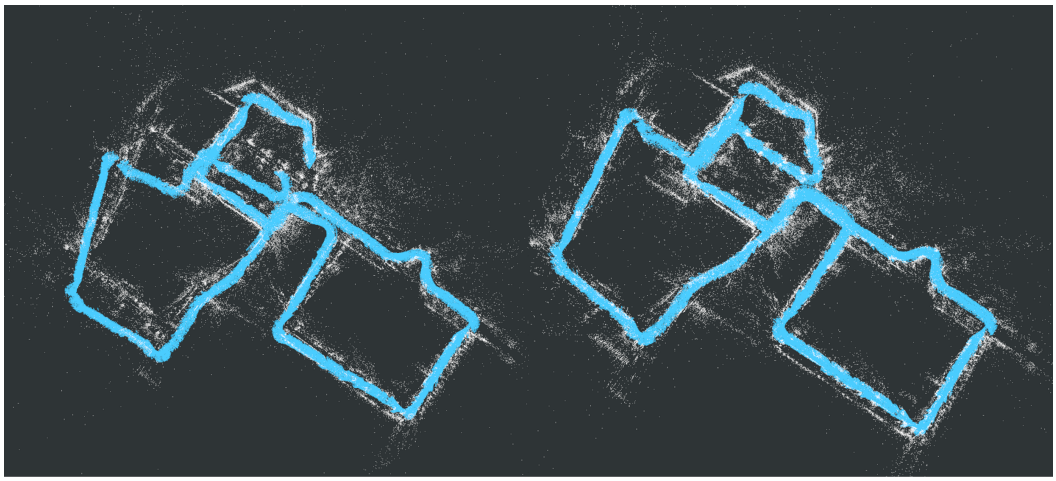
Segmentation Model		Accuracy (%)	Loss	mIoU (%)	FPS	Params
PIDNet_S	[39]	92.3	**0.111**	**80.2**	**45.2**	7.6 M
DeeplabV3	[57]	92.4	0.769	65.6	5.2	58.3 M
PsPUnet	[58]	90.1	0.312	75.2	24.9	46.7 M
BiSeNetV2	[59]	90.7	0.766	74.8	33.7	12.8 M
DDRNet_23	[60]	**96.1**	0.212	73.4	35.7	20.1 M

$$Accuracy = \frac{Number\ of\ Correctly\ Classified\ Pixels}{Total\ Number\ of\ Pixels\ in\ the\ Image} \quad (10)$$

The implementation results of loop closure for the drivable region mapping are shown in Figure 7. In Figure 7a, the point clouds of buildings overlapped with the drivable region, and the trajectory was not updated because of the unsuccessful loop closure of the drivable region. By contrast, Figure 7b exhibits successful loop closure of the drivable region.

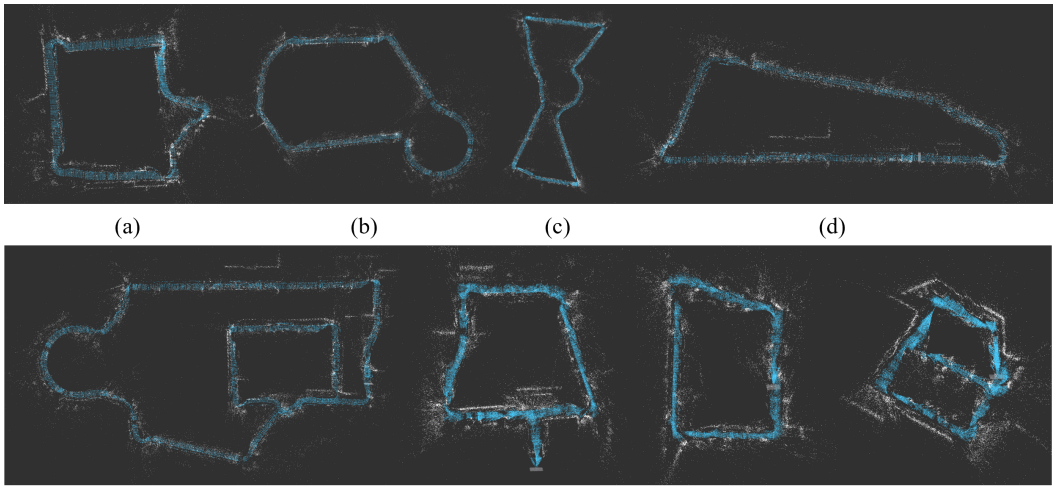
In Figure 8a–h, the results of the drivable region mapping using data captured along the paths presented in Figure 5 are shown. Sequentially, they correspond to Maps 1 to 8. The cyan point clouds represent the drivable region, and the white point clouds indicate features from ORB-SLAM3.

Figure 9 shows the result of the drivable region mapping using the Complex Urban dataset. Owing to the use of stereo images rather than RGB-D images, it was necessary to align the scale difference between the map results. Thus, alignment using evo [61] was performed, followed by performance evaluation, i.e., Root Mean Square Error (RMSE) and qualitative result evaluation. The results are shown in Table 2.



(a) (b)

Figure 7. (**a**) Drivable region mapping result without loop closure (**b**) Drivable region mapping result with loop closure.



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

(e) (f) (g) (h)

Figure 8. Drivable region mapping results at Jeonbuk National University: (**a**) map1, (**b**) map2, (**c**) map3, (**d**) map4, (**e**) map5, (**f**) map6, (**g**) map7, (**h**) map8.

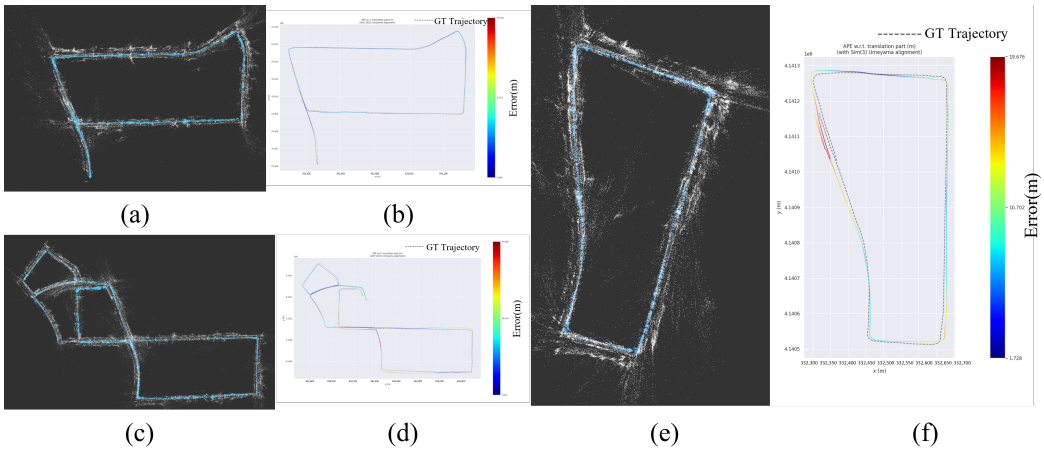


Figure 9. Comparison results of the Urban 28 dataset [55] using the proposed method with the provided ground truth trajectory. The trajectory of each map is manually divided, as shown in Figure 6. The comparison was conducted using the EVO [61]. The dashed line represents the ground-truth trajectory of Urban Region 28. (**a**) Map 1 drivable region mapping results. (**b**) Map 1 trajectory-comparison image. (**c**) Map 3 drivable region mapping results. (**d**) Map 3 trajectory-comparison image. (**e**) Map 2 drivable region mapping results. (**f**) Map 2 trajectory-comparison image.

4.3. Map-Merging Results

For map-merging , the process described in Figure 4 was sequentially applied to the data captured in Figure 8a–h, resulting in Figure 10c. For comparison, map-merging was also performed using the trajectory provided by ORB-SLAM3, yielding Figure 10b. The frame of the most visually similar image in each map was manually identified. The quaternion values specified in the trajectory were then used to generate the transformation matrix for each map, which was applied to the world coordinate system. Figure 10a represents the best result obtained by visually comparing and aligning the point clouds of each map for the optimal merging outcome.

A comparison of Figure 10a,b reveals that Figure 10b has significant errors. Discrepancies are particularly noticeable in the central circular section. By contrast, Figure 10c demonstrates precise alignment, indicating the successful execution of our proposed method.

In Figure 11a,b, the increase in data distorted the overall map, even in regions where the original loop closure performed well. Additionally, despite using the same dataset and settings, large-scale map generation consistently produces different results. By contrast, Figure 11c is the result of data captured in separate regions to create maps, avoiding issues during map creation with multiple loops. This approach produced more stable results during large-scale mapping.

In Figure 12a,b, the map and trajectory of the provided ground truth are shown. Figure 12b shows the trajectories of maps 1, 2, and 3, which are displayed in the same way as those in Figure 6. As shown in Figure 12c,d, ORB-SLAM3 failed to localize the camera pose. Therefore, the mapping result also failed even though we reran the SLAM after the initialization process. Figure 12e,f shows the results of the proposed method. Compared with the ORB-SLAM3 result, our proposed method produces better results. It is also evident that, in a large-scale area, dividing the map region and generating a map within those areas yields better results. These findings are summarized in Table 2.

As shown in Figures 11a,b and 12c,d, the results of existing methods that perform mapping without dividing the path do not consistently yield the same outcomes because of drift error, resulting in failures in achieving proper mapping. However, as shown in Figures 11c and 12e, the results obtained via the proposed method demonstrate significant improvement.

The proposed method successfully performs drivable region mapping, as demonstrated in Figures 11c and 12e. It accurately detects roads where vehicles can travel, enabling the creation of maps at a large scale.

Moreover, to the best of our knowledge, this study represents the first instance of large-scale drivable region mapping using only a camera.

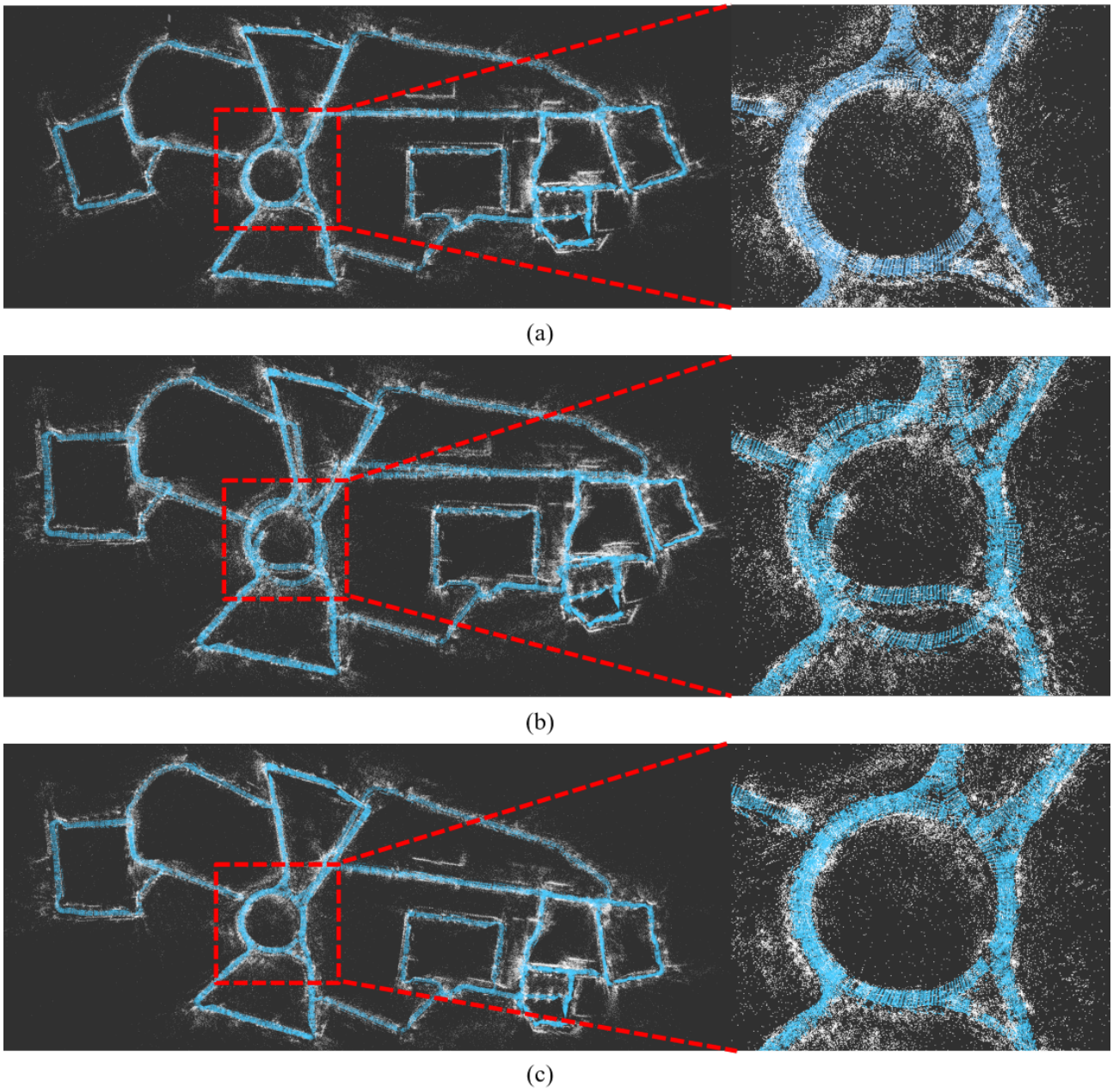
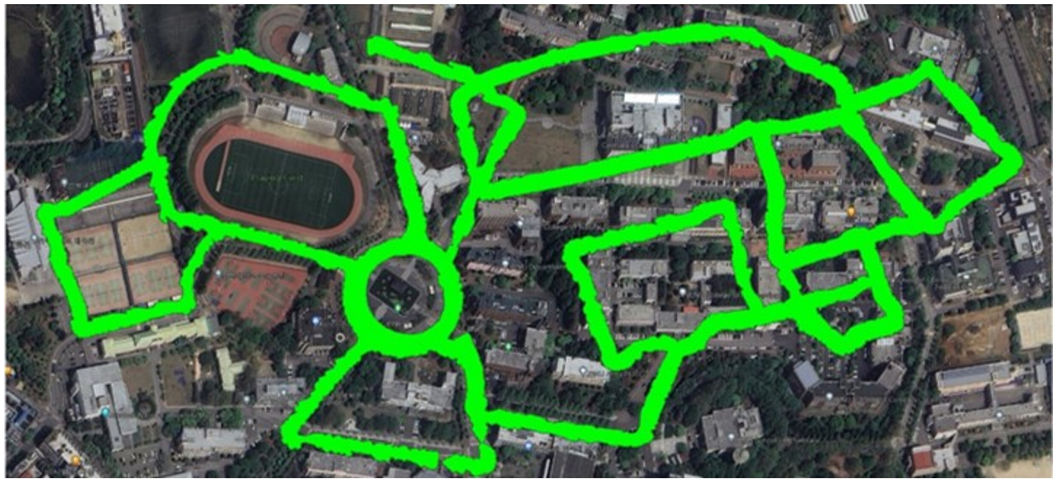


Figure 10. Map-merging results of Jeonbuk National University. A comparison of map-merging using three different methods is shown in the accompanying image. By zooming in on a portion of the map, we compare the quality on the quality on the degree of overlap in the drivable regions. The more accurately the central circles overlap, the better the result. (**a**) Map-merging by manually transforming the positions of the eight maps. (**b**) Map-merging using image similarity and the poses of the eight maps. (**c**) Map-merging via the proposed method.



(a)



(b)



(c)

Figure 11. A large-scale drivable region map created using two methods (a single run without splitting data; the proposed method) is manually overlaid on a satellite image. To ensure a fair comparison, only the drivable regions were retained. (**a**,**b**) Each image depicts a large-scale drivable map created from data captured in a single run. Despite using the same dataset in the same environment, the results showed significant inconsistencies. (**c**) Image of a large-scale drivable map created via the proposed method.

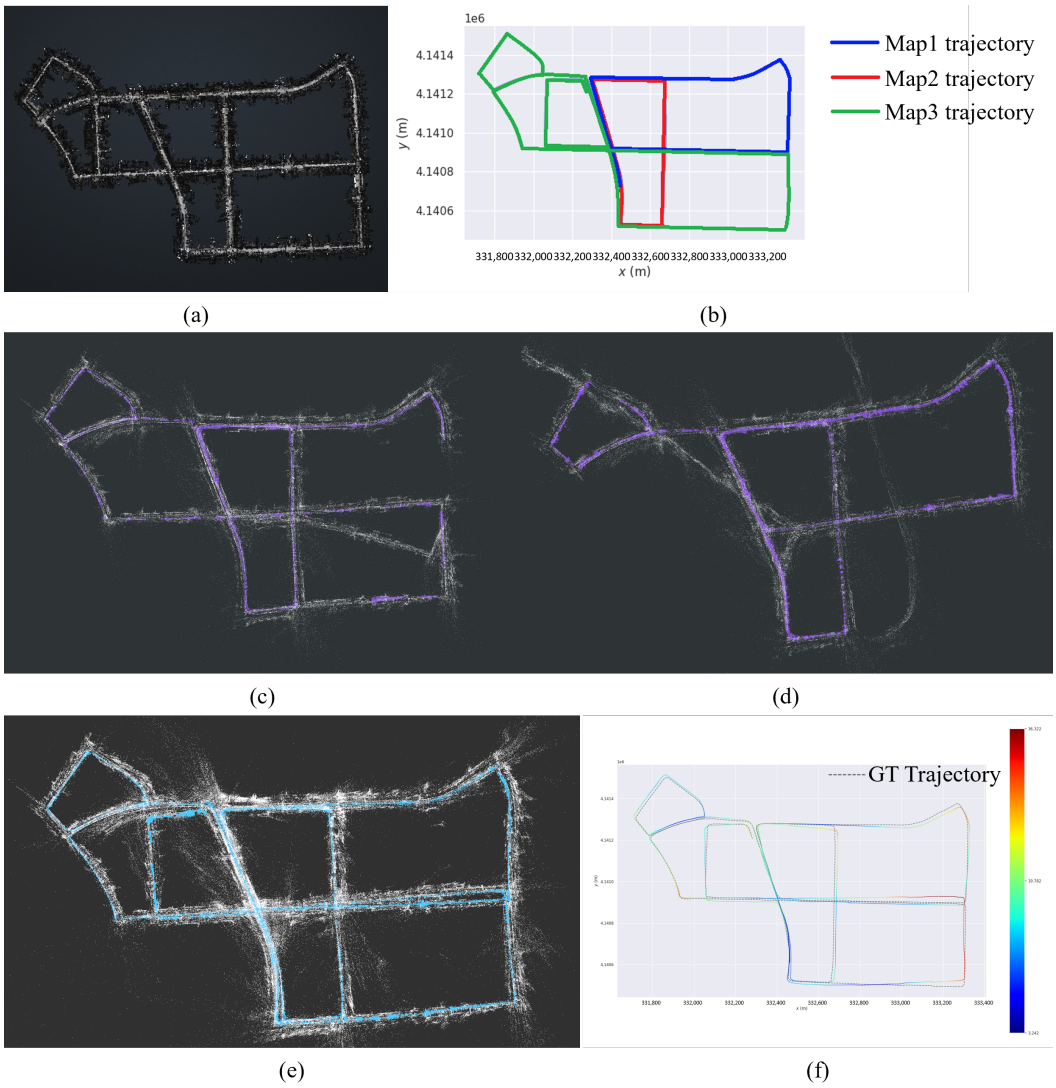


Figure 12. Urban 28 dataset [55] drivable region mapping by two methods (single run without splitting data; proposed method) results. (**a**) The provided ground-truth point cloud map of Urban 28. (**b**) The provided ground-truth trajectory of Urban 28. Blue, red, and green represent the trajectories of maps 1, 2, and 3, respectively. (**c**) Result of the first run of drivable region mapping without splitting the data. The resulting map shows misalignment. (**d**) Result of the second run of drivable region mapping without splitting the data. The resulting map shows misalignment. (**e**) A drivable region map produced by dividing the map and merging it with the proposed method. (**f**) Comparison between the trajectory generated via the proposed method and the ground-truth trajectory via EVO [61]. The dashed line represents the ground-truth trajectory of Urban 28.

5. Conclusions

This study proposed a method for mapping large-scale drivable regions that uses semantic segmentation, visual simultaneous localization and mapping, and map-merging. Semantic visual simultaneous localization and mapping identifies regions where robots can safely navigate, using semantic segmentation and visual localization to create maps that ensure safe movement. The created maps are merged into a single large-scale map using our proposed map-merging method, which uses 3D point cloud registration.

The semantic segmentation and visual simultaneous localization and mapping used for real-time drivable region mapping demonstrated adequate real-time performance, enabling the creation of drivable region maps in real-world scenarios. Semantic segmentation and visual simultaneous localization and mapping were integrated through pose transformation and loop closure.

In the process of map-merging to create large-scale maps, a method that uses 3D point cloud registration based on the feature descriptors of overlapping regions' 3D point clouds was proposed. This method yielded positive results, particularly with respect to the drift error of visual simultaneous localization and mapping, which tends to increase as the number of images increases. Moreover, our method demonstrated good results compared with real satellite maps.

The proposed method successfully generated a large-scale map of drivable regions. Our method was verified with both our large-scale custom dataset and publicly available benchmark dataset. Furthermore, the proposed map-merging method mitigated drift errors during simultaneous localization and mapping (SLAM) in large-scale environments. On average, only 19% of the data—the ratio of images that correspond to the overlapping area among the entire images—was used for map-merging. Despite utilizing a relatively small amount of data, the process was highly effective. These findings highlight the efficiency and accuracy of our approach to drivable region mapping, a critical component for path planning in autonomous driving systems.

Our future work will focus on visual localization and map updates on large-scale drivable maps composed of 3D point clouds. Additionally, we aim to explore visual-based large-scale drivable mapping in areas with limited features, such as rural environments, as opposed to university campus settings.

Author Contributions: Conceptualization, C.H. and D.Y.; methodology, C.H. and D.Y.; software, C.H. and D.Y.; validation, C.H.; formal analysis, C.H.; investigation, C.H. and D.Y.; resources, G.W. and S.C.K.; data curation, C.H.; writing—original draft preparation, C.H. and D.Y.; writing—review and editing, C.H. and H.J.; funding acquisition, H.J. All authors have read and agreed to the published version of the manuscript

Funding: This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)—Innovative Human Resource Development for Local Intellectualization program grant, funded by the Korean government (MSIT) (IITP-2024-RS-2024-00439292, 50%), and the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [23ZK1100, Honam Region Regional Industry-based ICT Convergence Technology Advancement Support Project]. If you intend to utilize the contents of this report, you must disclose that the research was funded by the Electronics and Telecommunications Research Institute (ETRI).

Data Availability Statement: The data presented in this study are available upon request from the corresponding author, and in some of the referenced papers.

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

References

1. Matsuki, H.; Murai, R.; Kelly, P.H.; Davison, A.J. Gaussian splatting slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle WA, USA, 17–21 June 2024; pp. 18039–18048.
2. Xu, H.; Liu, P.; Chen, X.; Shen, S. D^2SLAM: Decentralized and Distributed Collaborative Visual-Inertial SLAM System for Aerial Swarm. *IEEE Trans. Robot.* **2024**, *40*, 3445–3464. [CrossRef]
3. Lu, B.X.; Tsai, Y.C.; Tseng, K.S. GRVINS: Tightly Coupled GNSS-Range-Visual-Inertial System. *J. Intell. Robot. Syst.* **2024**, *110*, 36. [CrossRef]
4. Xu, M.; Zhang, Z.; Wei, F.; Hu, H.; Bai, X. Side adapter network for open-vocabulary semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 2945–2954.
5. Zhang, Z.; Yang, B.; Wang, B.; Li, B. Growsp: Unsupervised semantic segmentation of 3d point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 17619–17629.

6. Ding, J.; Xue, N.; Xia, G.S.; Schiele, B.; Dai, D. Hgformer: Hierarchical grouping transformer for domain generalized semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 15413–15423.
7. Wang, J.; Rünz, M.; Agapito, L. DSP-SLAM: Object oriented SLAM with deep shape priors. In Proceedings of the 2021 International Conference on 3D Vision (3DV), London, UK, 1–31 December 2021; IEEE: New York, NY, USA, 2021; pp. 1362–1371.
8. Wu, Y.; Zhang, Y.; Zhu, D.; Deng, Z.; Sun, W.; Chen, X.; Zhang, J. An object slam framework for association, mapping, and high-level tasks. *IEEE Trans. Robot.* **2023**, *39*, 2912–2932. [CrossRef]
9. Wang, Y.; Xu, B.; Fan, W.; Xiang, C. Qiso-slam: Object-oriented slam using dual quadrics as landmarks based on instance segmentation. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2253–2260. [CrossRef]
10. Lee, H.C.; Lee, S.H.; Choi, M.H.; Lee, B.H. Probabilistic map merging for multi-robot RBPF-SLAM with unknown initial poses. *Robotica* **2012**, *30*, 205–220. [CrossRef]
11. Birk, A.; Carpin, S. Merging Occupancy Grid Maps From Multiple Robots. *Proc. IEEE* **2006**, *94*, 1384–1397. [CrossRef]
12. Ma, L.; Zhu, J.; Zhu, L.; Du, S.; Cui, J. Merging grid maps of different resolutions by scaling registration. *Robotica* **2016**, *34*, 2516–2531. [CrossRef]
13. Cui, M.; Femiani, J.; Hu, J.; Wonka, P.; Razdan, A. Curve matching for open 2D curves. *Pattern Recognit. Lett.* **2009**, *30*, 1–10. [CrossRef]
14. Amigoni, F.; Gasparini, S.; Gini, M. Building Segment-Based Maps Without Pose Information. *Proc. IEEE* **2006**, *94*, 1340–1359. [CrossRef]
15. Huang, W.H.; Beevers, K.R. Topological map merging. *Int. J. Robot. Res.* **2005**, *24*, 601–613. [CrossRef]
16. Bernuy, F.; Ruiz-del Solar, J. Topological semantic mapping and localization in urban road scenarios. *J. Intell. Robot. Syst.* **2018**, *92*, 19–32. [CrossRef]
17. Groß, J.; Ošep, A.; Leibe, B. Alignnet-3d: Fast point cloud registration of partially observed objects. In Proceedings of the 2019 International Conference on 3D Vision (3DV), Québec City, QC, Canada, 16–19 September 2019; IEEE: New York, NY, USA, 2019; pp. 623–632.
18. Le, H.M.; Do, T.T.; Hoang, T.; Cheung, N.M. SDRSAC: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 124–133.
19. Pais, G.D.; Ramalingam, S.; Govindu, V.M.; Nascimento, J.C.; Chellappa, R.; Miraldo, P. 3dregnet: A deep neural network for 3d point registration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 7193–7203.
20. Chiciudean, V.; Florea, H.; Beche, R.; Oniga, F.; Nedevschi, S. Data augmentation for environment perception with unmanned aerial vehicles. *IEEE Trans. Intell. Veh.* **2024**. [CrossRef]
21. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625. [CrossRef] [PubMed]
22. Ma, Z.W.; Cheng, W.S. Visual-Inertial RGB-D SLAM with Encoder Integration of ORB Triangulation and Depth Measurement Uncertainties. *Sensors* **2024**, *24*, 5964. [CrossRef] [PubMed]
23. Jo, H.; Lee, W.; Kim, E. Mixture density-PoseNet and its application to monocular camera-based global localization. *IEEE Trans. Ind. Inform.* **2020**, *17*, 388–397. [CrossRef]
24. 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]
25. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
26. Lu, F.; Milios, E. Globally consistent range scan alignment for environment mapping. *Auton. Robot.* **1997**, *4*, 333–349. [CrossRef]
27. Elhashash, M.; Qin, R. Cross-view SLAM solver: Global pose estimation of monocular ground-level video frames for 3D reconstruction using a reference 3D model from satellite images. *ISPRS J. Photogramm. Remote Sens.* **2022**, *188*, 62–74. [CrossRef]
28. Arun, A.; Ayyalasomayajula, R.; Hunter, W.; Bharadia, D. P2SLAM: Bearing based WiFi SLAM for indoor robots. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3326–3333. [CrossRef]
29. Duan, X.; Hu, Q.; Zhao, P.; Wang, S. A low-cost and portable indoor 3D mapping approach using biaxial line laser scanners and a one-dimension laser range finder integrated with microelectromechanical systems. *Photogramm. Eng. Remote Sens.* **2022**, *88*, 311–321. [CrossRef]
30. Zhang, D.; Xu, B.; Hu, H.; Zhu, Q.; Wang, Q.; Ge, X.; Chen, M.; Zhou, Y. Spherical Hough Transform for Robust Line Detection Toward a 2D–3D Integrated Mobile Mapping System. *Photogramm. Eng. Remote Sens.* **2023**, *89*, 50–59. [CrossRef]
31. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
32. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, Part III*; Springer: Cham, Switzerland, 2015; pp. 234–241.
33. Ning, S.; Ding, F.; Chen, B. Research on the method of foreign object detection for railway tracks based on deep learning. *Sensors* **2024**, *24*, 4483. [CrossRef]

34. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
35. Zhang, Z.; Zhang, X.; Peng, C.; Xue, X.; Sun, J. Exfuse: Enhancing feature fusion for semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 269–284.
36. Fang, Y.; Wang, W.; Xie, B.; Sun, Q.; Wu, L.; Wang, X.; Huang, T.; Wang, X.; Cao, Y. Eva: Exploring the limits of masked visual representation learning at scale. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 19358–19369.
37. Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; Li, L. Solo: Segmenting objects by locations. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XVIII*; Springer: Cham, Switzerland 2020; pp. 649–665.
38. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 9157–9166.
39. Xu, J.; Xiong, Z.; Bhattacharyya, S.P. PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 19529–19539.
40. Gosala, N.; Petek, K.; Drews, P.L.J., Jr.; Burgard, W.; Valada, A. SkyEye: Self-Supervised Bird's-Eye-View Semantic Mapping Using Monocular Frontal View Images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 14901–14910.
41. Lin, S.; Wang, J.; Xu, M.; Zhao, H.; Chen, Z. Topology aware object-level semantic mapping towards more robust loop closure. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7041–7048. [CrossRef]
42. Chen, H.; Zhang, G.; Ye, Y. Semantic loop closure detection with instance-level inconsistency removal in dynamic industrial scenes. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2030–2040. [CrossRef]
43. Qin, C.; Zhang, Y.; Liu, Y.; Lv, G. Semantic loop closure detection based on graph matching in multi-objects scenes. *J. Vis. Commun. Image Represent.* **2021**, *76*, 103072. [CrossRef]
44. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Beetz, M. Learning informative point classes for the acquisition of object model maps. In Proceedings of the 2008 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 17–20 December 2008; pp. 643–650. [CrossRef]
45. Rusu, R.B.; Blodow, N.; Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3212–3217. [CrossRef]
46. Arbeiter, G.; Fuchs, S.; Bormann, R.; Fischer, J.; Verl, A. Evaluation of 3D feature descriptors for classification of surface geometries in point clouds. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, 7–12 October 2012; pp. 1644–1650. [CrossRef]
47. Salti, S.; Tombari, F.; Di Stefano, L. SHOT: Unique signatures of histograms for surface and texture description. *Comput. Vis. Image Underst.* **2014**, *125*, 251–264. [CrossRef]
48. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*; Schenker, P.S., Ed.; International Society for Optics and Photonics, SPIE: Cergy-Pontoise, France, 1992; Volume 1611, pp. 586–606. [CrossRef]
49. Segal, A.; Haehnel, D.; Thrun, S. Generalized-icp. In Proceedings of the Robotics: Science and Systems, Seattle, WA, USA, 28 June–1 July 2009; Volume 2, p. 435.
50. Cho, H.M.; Jo, H.; Kim, E. SP-SLAM: Surfel-Point Simultaneous Localization and Mapping. *IEEE/ASME Trans. Mechatronics* **2022**, *27*, 2568–2579. [CrossRef]
51. Yang, H.; Shi, J.; Carlone, L. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Trans. Robot.* **2021**, *37*, 314–333. [CrossRef]
52. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
53. 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]
54. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
55. Jeong, J.; Cho, Y.; Shin, Y.S.; Roh, H.; Kim, A. Complex urban dataset with multi-level sensors from highly diverse urban environments. *Int. J. Robot. Res.* **2019**, *38*, 642–657. [CrossRef]
56. Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; Zhao, H. Depth anything: Unleashing the power of large-scale unlabeled data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 10371–10381.
57. Chen, L.C.; Papandreou, G.; Schroff, F.; Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv* **2017**, arXiv:1706.05587.
58. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
59. Yu, C.; Gao, C.; Wang, J.; Yu, G.; Shen, C.; Sang, N. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *Int. J. Comput. Vis.* **2021**, *129*, 3051–3068. [CrossRef]

60. Hong, Y.; Pan, H.; Sun, W.; Jia, Y. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv* **2021**, arXiv:2101.06085.
61. Grupp, M. Evo: Python Package for the Evaluation of Odometry and SLAM. Available online: https://github.com/MichaelGrupp/evo (accessed on 1 December 2017).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.