electronics

MDPI

Article

Lane-Level Map Generation and Management Framework Using Connected Car Data

Jungseok Kim, Jeongmin Moon and Changjoo Moon *

Intelligent Data Processing Laboratory, Department of Smart Vehicle Engineering, Konkuk University, Seoul 05029, Republic of Korea; dyori04@konkuk.ac.kr (J.K.); mjm9804@konkuk.ac.kr (J.M.)
* Correspondence: cjmoon@konkuk.ac.kr

Abstract: This study proposes a lane-level map generation and management framework using connected sensor data to reduce the manpower and time required for producing and updating high-definition (HD) maps. Unlike previous studies that relied on the onboard processing capabilities of vehicles to collect map-constructing elements, this study offloads computing for map generation to the cloud, assigning vehicles solely the role of transmitting sensor data. For efficient data collection, we divide the space into a grid format to define it as a partial map and establish the state of each map and its transition conditions. Lastly, tailored to the characteristics of the road elements composing the map, we propose an automated map generation technique and method for selectively collecting data. The map generation method was tested using data collected from actual vehicles. By transmitting images with an average size of 350 KB, implementation was feasible even with the current 5G upload bandwidth. By utilizing 12,545 elements, we were able to achieve a position accuracy and regression RMSE of less than 0.25 m, obtaining 651 map elements to construct the map. We anticipate that this study will help reduce the manpower and time needed for deploying and updating HD maps.

Keywords: connected car; vehicle sensor data; lane-level map; big data platform; cloud computing

Citation: Kim, J.; Moon, J.; Moon, C. Lane-Level Map Generation and Management Framework Using Connected Car Data. *Electronics* **2023**, *12*, 3738. https://doi.org/10.3390/electronics12183738

Academic Editors: Calin Iclodean, Bogdan Ovidiu Varga and Felix Pfister

Received: 14 August 2023
Revised: 31 August 2023
Accepted: 2 September 2023
Published: 5 September 2023

1. Introduction

Overcoming the limits of sensor-based standalone autonomous driving to achieve higher-level autonomous driving is inevitable, and the usage of a high-definition (HD) map is a representative method [1,2] proposed for this purpose. HD maps can be useful in difficult sensor-based sensing tasks, such as lane recognition and GPS-based positioning because they provide precise road information. However, these maps are only provided in limited areas due to limitations in the construction method. In the existing construction method, a mobile mapping system (MMS)-equipped vehicle is used to survey roads. Roads and road elements (e.g., road lanes, traffic signs, etc.) [3] are then manually drawn based on the acquired data. The mapping range obtained through MMS driving is limited, and obtaining survey data for a wide range of roads takes a long time. Consequently, updating the data becomes difficult when the road environment changes. The quality of the collected data is determined by the collection environment; hence, it may decrease in the event of bad weather or when the sensor recognition range is blocked by other vehicles. The collected data requires labeling through human drawing; therefore, the construction and updating of an HD map requires a huge amount of human resources and time, making its utilization difficult to spread widely.

To address this concern, various studies have been conducted to construct HD maps using the data obtained from regular vehicles. A lane-level map has been constructed by obtaining elements that compose the road environment from the raw data of sensors [4,5] installed in vehicles or by utilizing the output of driving assistant systems, such as the lane-keeping system [6,7]. Furthermore, research is actively being conducted to detect changes in roads through crowdsourcing [4,8] and reflecting them in HD maps. With

the development of network technology and the emergence of "connected cars" (i.e., cars connected to the internet through cellular communication), an environment where sensor data from vehicles can be collected in real time is being established. The 5G Automotive Association (5GAA) has developed a use case for sharing the raw sensor output of driving vehicles through networks [9], enabling a real-time collection of vehicle sensor data in the cloud.

This study proposes a framework for collecting sensor data from vehicles during driving to create and manage HD maps on a cloud server, paying attention to the connected car trend. This framework does not require additional computing on the vehicle's onboard processor to update the HD map or rely on high-performance sensors, such as light detection and ranging (LiDAR). The server collects a connected car's global position and images from the camera of vehicles driving on the road and extracts the map elements composing the road environment (e.g., lanes and traffic signs). The extracted map elements are used to construct or update the HD maps. We divide the area into small grids and define these grids as a partial map. The extracted road elements are then placed on the partial map. Each map element placed on the partial map is automatically managed based on the update cycle of LDM layer 1 or 2 [10,11], which represents static information, thereby enabling the determination of its reliability.

The proposed method increases feasibility by using stereo camera data, one of the sensors commonly equipped in SAE autonomous level 1 or 2 vehicles. All data processing procedures, except for raw data compression and transmission, are performed on the server to minimize the computing load of the connected car used to manage the map. Criteria are established based on the state of the connected car and the completeness of the map in its location to request image data, reducing unnecessary data collection and minimizing network and computing resource usage. Finally, the sensor data collection, map generation, and management of the generated map are automated, supporting fast HD map generation and updates over a wide area.

The primary contributions of our study can be summarized as follows:

- We generated maps by simulating data that can be obtained from actual vehicles in operation, using only consumer-grade sensors;
- All processing required for map generation, excluding data compression and transmission, was performed on the server. This reduces the processing burden on vehicles, eliminating the need for high-performance processors. We verified that the data size can be transmitted through commercial 5G networks;
- We divided the areas targeted for map generation into grid forms called "PartialMap" and defined their states and transitions. This allows us to selectively identify areas needing data collection and updates;
- We leveraged a deep neural network inference (DNN) to extract road-composing elements in the on-premises server and utilized their morphological features to generate maps, thereby reducing processing resources and storage requirements.

The remainder of this paper is structured as follows: Section 2 introduces previous research on HD map automatic construction and management; Section 3 presents the overall architecture and activity diagram of the HD map creation and management; Section 4 describes the process of selecting connected cars for data collection, extracting map elements, assigning three-dimensional (3D) coordinates, and storing data in the database; Section 5 provides an overview of the entire management process presented in this work, whereas Section 6 suggests algorithms to construct lane-level map. Section 7 outlines the implementation results of the proposed technique and the testing process using a sample dataset, and Section 8 provides the conclusions and discussions.

2. Related Works

2.1. Network Solutions for Connected Car

The advancement in network solution technologies connecting vehicles to the internet has made it possible to collect data in real time from vehicles, thereby enabling the monitor-

ing of roads and traffic conditions to implement an intelligent transport system (ITS). The following are representative network solutions.

1. Long Range Wide Area Network (LoRaWAN)

A low-power area network (LPWAN) is a low-power network solution for long-distance communication with IoT devices such as smart farms and energy monitoring. LoRaWAN is a specialized protocol of LPWAN for long-distance communication. Research is being extensively conducted on applying this technology to vehicles for real-time communication. Mohammad et al. [12] investigated the performance of LoRaWAN when used on mobile devices with mobility. They demonstrated that while LoRaWAN has drawbacks such as high latency and low data transmission time, it is a network capable of operating with devices moving at high speeds and under various traffic loads. Gabriele et al. [13] studied the applicability of LoRaWAN in vehicles moving at high speeds. Using transmitters mounted on vehicles moving at various speeds from 20 km/h to 90 km/h, they showed that LoRaWAN maintains robust communication even in vehicles traveling at high speeds.

2. IEE 802.11p

IEEE 802.11p is a standard [14] for vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication to implement ITS. It uses the wireless access in vehicular environments (WAVE) protocol suite to immediately establish connections for interactions with rapidly moving vehicles and can provide stable communication in various road and weather conditions, unlike conventional Wi-Fi networks. On the other hand, due to its limited communication range (approx. 1 km) and low scalability, cellular vehicle to everything (C-V2X) is being proposed as an alternative.

3. NR C-V2X

C-V2X is a vehicle standard based on a cellular network, initially developed using long-term evolution [15]. With the advancement of 5G technology, the 5G-based communication standard New Radio (NR) C-V2X [16] has been developed, which supports improved bandwidth and lower latency compared to LTE. This comprises V2V, V2I communication directly between devices using Sidelink technology [17], and vehicle to network (V2N) using cellular network infrastructure. With its high bandwidth and low latency, NR C-V2X can be used for remote control, autonomous driving, information and data sharing, and platooning [18].

This study, assuming the commercialization of C-V2X capable of satisfying the characteristics of systems that require the upload of large volumes of data, implemented communication with vehicles using mobile devices operating on a 5G network.

2.2. Collecting, Storing, and Processing Vehicle Sensor Data

With the recent advancement of vehicle sensor systems, the amount of sensor data generated from vehicles has also dramatically increased [19]. This has resulted in a growing interest in studying vehicle sensor data from a big data perspective. Attila et al. [20] proposed an Internet-of-Things platform for connected cars that aims to process sensor data by introducing cloud computing and big data processing technologies. They reliably collected and processed data from multiple sensors. Aelee et al. [21] defined a data schema for collecting and processing sensor data from moving vehicles. They used this schema to build a distributed data processing platform that allows the detection of abnormal data from vehicles. Theodoros et al. [22] proposed a system that introduces a distributed architecture platform for the continuous collection, storage, and analysis of vehicle state data. Meanwhile, David et al. [23] collected real-time road environment data through a vehicle's onboard diagnostics II port and smartphone devices. These data were transmitted to the monitoring system via short-range communication technologies, such as Bluetooth Low Energy, Wi-Fi, and cellular networks. Kieun et al. [24] proposed a system that utilizes the black boxes installed in vehicles for real-time road monitoring. Their system successfully

disseminated relevant information to the affected vehicles within a minute, achieving near real-time communication. Research focusing on the collection and processing of vehicle sensor data to monitor road and driving conditions in real time is currently underway. The present study also follows this trend and extracts valuable information from the sensor data of the driving vehicle.

2.3. Automated HD Map Generation

Many studies on map creation and update automation have been conducted to overcome the limitations of traditional HD map construction methods using the MMS. Onkar et al. [5] proposed an end-to-end system for generating a 3D map via crowdsourcing, utilizing only a consumer-grade single camera and GPS. This system detects traffic signs and lanes and reconstructs them in 3D coordinates using triangulation and road surface estimation techniques between image frames of each drive. Additionally, by clustering the 3D landmarks of traffic signs and lanes obtained from multiple driving data and using bundle adjustment, they precisely represented them in 3D pose. Massow et al. [6] developed a fundamental approach for road environment inferring and map updating using hundreds of millions of terabytes of vehicular probe data, considering the increase in the adoption of connected cars. They also designed a framework that enhanced scalability and used it to perform lane inferring by utilizing the GPS data and the semiautonomous driving technology of the vehicle, ultimately providing a proof of concept. Kitae et al. [8] proposed an algorithm that processes the sensor data collected from commonly used low-cost cameras and a consumer-grade global navigation satellite system (GNSS) in vehicles to acquire map landmarks. They created a map and updated it through a comparison with the existing HD map. Instead of using vehicles dedicated to data collection, such as the MMS, statistical processing was conducted using the data collected from commercial vehicles like buses. The existing HD map data were integrated to compensate for the low accuracy, a drawback of crowdsourced data.

Research is actively being conducted on real-time HD maps, updating using data from high-performance sensors such as LiDAR, mounted on autonomous vehicles. Chansoo et al. [25] proposed a crowdsourcing mapping process to create a new feature layer for the HD map that is transmitted to the map cloud via a cellular network and integrated after alignment. Qi et al. [26] proposed an artificial intelligence neural network called "HDMapNet", which uses images acquired from multiple cameras for end-to-end HD map generation. Their neural network predicted the map elements observed in a bird's-eye view into a vectorized map. Meanwhile, Junwon et al. [27] introduced the Edge–Fog–Cloud architecture to generate HD maps from vehicle sensor data. The Fog server performed the normal distribution transform–simultaneous localization and mapping (NDT-SLAM) to generate an HD map from the LiDAR point cloud, while the cloud server converted the generated HD map to a global coordinate frame for compilation.

This study followed an approach similar to that in the previous research [5–8] to generate a map from the consumer-grade sensor data. Also, DNN inference was adopted for road element detection similar to [26]. However, unlike previous studies [5–8,25,26], we minimized the load on the onboard processors of vehicles by transmitting sensor data to an on-premises cloud server. This allowed for more high-performance computations than that which the processors in the vehicles could achieve. The data used for map generation demonstrated a size that can be sufficiently uploaded even with the current commercial 5G upload bandwidth. Moreover, through methods for selective data collection (collection canager; Section 4) and managing elements within the map (partial map, map manager; Section 5), we efficiently utilized network resources and reduced unnecessary processing loads. Lastly, building upon prior research that either simply determined the element's positions through clustering [6,7] or adjusted the position of elements from a computer vision perspective, which was sensitive to external factors such as light and shadows [5], our approach incorporated the morphological characteristics of road elements (linear, planar). This allowed us to refine the vehicle's pose, filter out outliers, and subsequently enhance

accuracy. In this field, one commonly used approach is the division of space into grids to represent the location information of map elements. Sooyeon et al. [28] utilized grid-based mapping to depict dynamic objects on the road. Unlike their study that mapped dynamic objects 1:1 to a cell of a grid, we introduced grid mapping to index the elements composing the map.

3. Overall System Architecture

This section describes the overall system architecture used to conduct data collection for connected cars and HD map generation and management. The architecture components and their interactions are also discussed herein.

The proposed system architecture is divided into three main parts: data collection, processing, and storage (Figure 1).

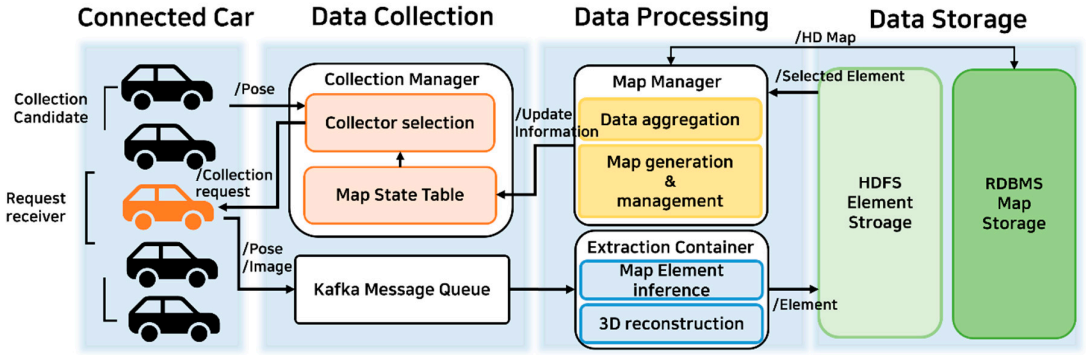


Figure 1. Overall system architecture.

The data collection is divided into the collection manager part, which selects connected cars from which to collect sensor data, and the Kafka message queue part, which caches and delivers the collected sensor data to the data processing part. The collection manager classifies the cars connected to the system into two groups: collection candidates, which have not yet sent data collection requests, and request receivers, which have already sent data collection requests. All cars connected to the system can transmit their 6DOF pose and pose accuracy. The collection manager then selects the vehicles to collect data from through the map state table, which stores the map state at a specific location. The map state table is a key-value database that divides the entire map into grids, with the key being the reference point of each grid and the value being the map information. The selected vehicles classified as request receivers send their location and images to the server, while the unselected ones classified as collection candidates only transmit pose information. The data collected from the selected vehicles are managed in the Kafka message queue and sent to the data processing part.

The data processing part consists of the extraction container and the map manager. The former uses a DNN to detect map elements composing the HD map. The detected elements are then given 3D coordinates through stereo image matching and forwarded to the data storage part. The map manager generates an HD map using the map elements extracted by the extraction container and manages the existing map. The data storage part uses the Hadoop distribution file system (HDFS) [29] Cluster to store the element information extracted by the extraction container and deliver it to the map manager. The RDBMS database is utilized to store the HD map generated by the map manager.

Figure 2 presents the UML activity diagram [30] for the entire system. The orange boxes represent the interaction between the connected car and the data collection part. The blue boxes depict the map element extraction from the sensor data and map management. Lastly, the green boxes present the map storage process in the RDBMS.

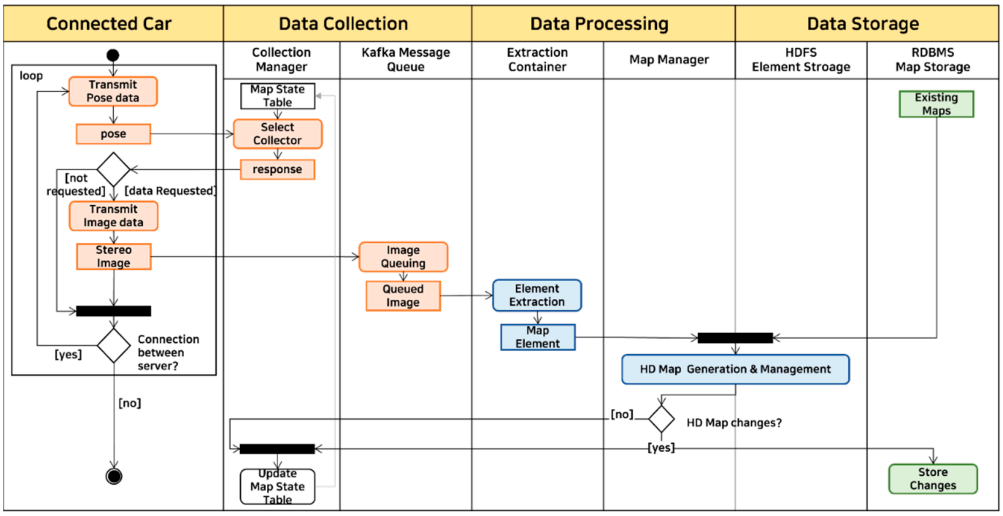


Figure 2. UML activity diagram for the entire system.

The connected car sends the pose data and uncertainty matrix for pose accuracy to the collection manager as long as the connection with the server is maintained. The collection manager requests for the image data if a map is being generated or outdated in the area where the connected car that sent the data is located or if the vehicle's pose accuracy is below a certain threshold. The collection manager notifies whether or not a data request is required in its response. Upon receiving a data request, the connected car transfers the stereo image data acquired from the vehicle sensor to the Kafka message queue [31]. Section 4.1 describes the detailed collection process.

Subsequently, the image data are transmitted to the extraction container via the Kafka message queue. The extraction container infers the map elements present in the image through object detection and assigns 3D coordinates. The map manager then uses the newly created element information to generate a map or update an existing one. The elements generated in the extraction container and the maps created in the map manager are saved as files in the HDFS cluster. Section 5 discusses the generation and management methods in detail.

In case of any change in the map, the updated map elements are stored in the RDBMS map storage. The new map state, including the last update time, is reflected in the collection manager's map state Table for use when collecting data from the connected car later.

4. Data Collection and Road Element Extraction

This section describes the role of a vehicle in collecting data and the cloud components involved in the data collection, including the collection manager and the message queue. The vehicle uses a commercial 5G mobile cellular network to communicate with the collection manager via HTTP and upload image data. The images are compressed enough to be uploaded within the 5G link's upload bandwidth. The images are sufficiently compressed to be transformed into a format uploadable within the 5G link's bandwidth. The extraction manager's function in generating the road elements from the collected data and the representation of these elements are also presented here.

4.1. Data Collection and Assessment

Figure 3 provides a detailed representation of the interaction between the vehicle and the data collection part, as described in Figure 2. The orange boxes represent the activities of the connected car, while the blue ones depict those of the data collection part.

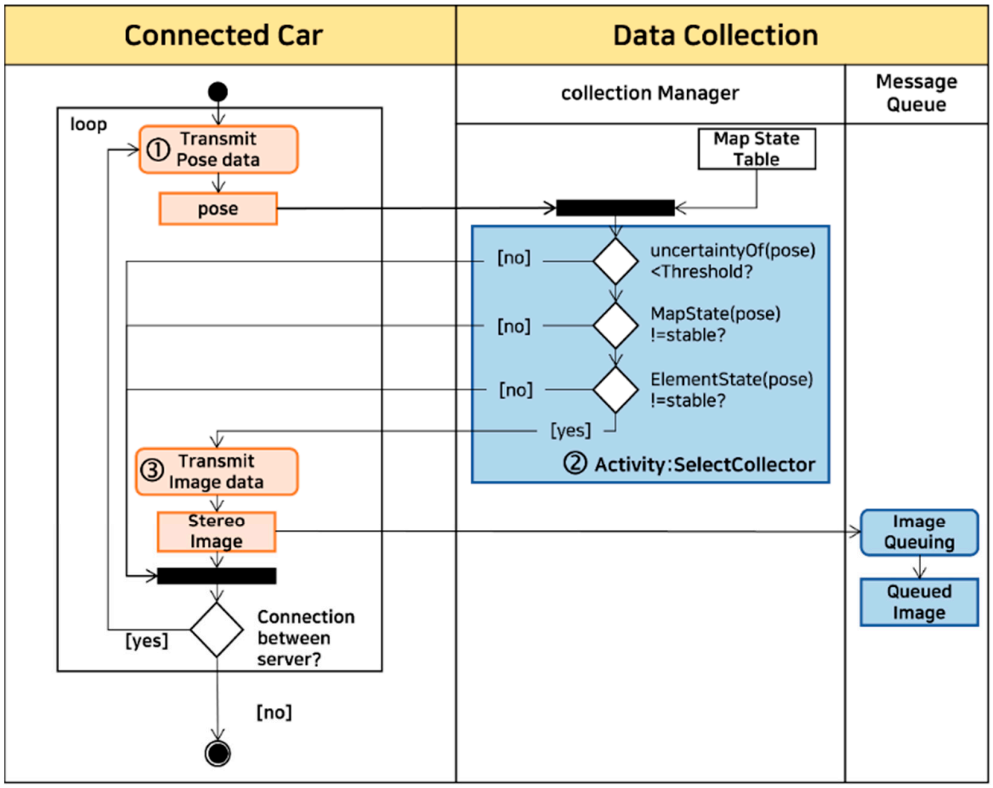


Figure 3. Interaction between the vehicle and the data collection part.

The connected car, hereinafter referred to as the "collector", collects two data types, that is, pose and stereo images. The collection manager evaluates the accuracy of the collected data from the collector and requests it to send the collected image data to the message queue.

1. The collector sends the 6DOF pose and pose uncertainty information to the collection manager using the HTTP Post Protocol. The two data types are fused using an Unscented Kalman filter (UKF) [32,33] to enable the acquisition of an uncertainty matrix for each term in the pose. UKF is an extension of the Kalman filter for nonlinear systems using a deterministic sampling technique. Kalman filter is a recursive algorithm used for estimating the evolving state of a linear system from noisy measurements.
2. The collection manager bases its data collection requests on the three following factors: pose uncertainty of the collector; the state of the map, where the collector is located; and the state of the elements within the collector's observation range. The uncertainty threshold is heuristically determined by considering the lane width in urban areas, with the position uncertainty covariance determinant set to 1.5 m and the orientation uncertainty covariance determinant set to 5.7°. We assumed that the distribution of the pose follows a normal distribution characterized by a covariance matrix as its uncertainty matrix. In urban areas, the width of a lane is an average of 3 m by Korean standards. When the position uncertainty exceeds 1.5 m, it becomes indeterminate which lane is occupied. If the orientation exceeds 5.7°, the position uncertainty of an element away from the vehicle, the maximum distance suitable for stereo matching, is approximately 1.5 m. This makes it similarly challenging to determine which lane it belongs to, as in the case of position uncertainty. To prevent an inefficient duplication of data collection, the system will not send a collection request if the map of the area where the vehicle is located has been updated with the recently collected data and is in a stable state, or if the elements within the vehicle's observable range have already been updated and are in a stable state. Section 5.2 presents the detailed conditions for the data collection.

3. If the collector receives an image collection request from the collection manager, it transmits the image message to its assigned "Topic" of the Kafka message queue using a Kafka producer. A "topic" in Kafka is a category or feed name where records are published. When the collector receives an image data collection request from the collection manager, it immediately performs Base64 encoding and lz4 compression [34] on stereo images, such that they can be received by the Kafka broker. This process compresses the data by approximately 80%, enabling an effective transmission over 5G mobile networks [35]. The Kafka producer in the collector sends image data to the broker, acting as the manager of the topic via the TCP/IP network protocol. The broker then dispatches the image data to 'extraction containers' in the form of a Docker container [36] assigned to each collector (vehicle). Using the Kafka consumer, the extraction container monitors its designated topic and processes the latest data from the topic to generate an extracted element, as detailed in Section 4.2.

4.2. Road Element Extraction

Figure 4 is a flowchart representing the entire process of extracting elements from the image data collected by the extraction container. Each container is assigned to one topic on the Kafka message queue.

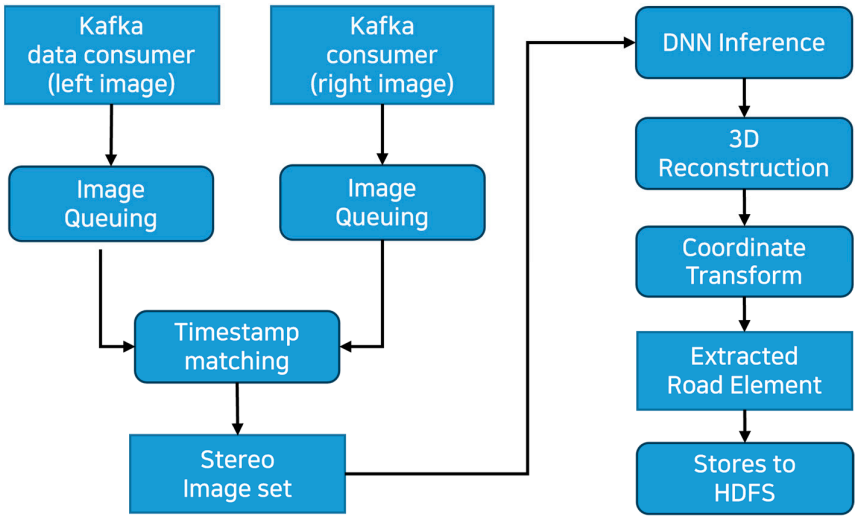


Figure 4. Process flowchart of element extraction from the image data.

The arrival speed of images from the left and right cameras can differ; therefore, the extraction container queues images acquired from both sides of the stereo camera. The road element detection for images collected at the same moment is performed using a DNN. Reconstruction and coordinate transform are conducted to obtain the 3D coordinates of the detected elements.

This study used object detection to acquire the road map elements. Another option for road environment perception using the DNN inference is semantic segmentation, which identifies the object class of each pixel in an image. However, handling pole-like objects through semantic segmentation is challenging due to the difficult separation of individual instances. Considering this, we chose an object detection model for use in this work. The 3D Reconstruction process is implemented by generating a sparse point cloud through the camera intrinsic and disparity map. The image is preprocessed based on the pixel coordinates of the elements acquired through a DNN. The road markings or pole-like objects are then separated from the image. The disparity, defined as the difference between the left and right images, is generated using the stereo semi-global block matching algorithm [37,38]. By applying the camera intrinsic, the 3D coordinates are gained for each pixel with an assigned depth.

The reconstructed point cloud may contain noise in addition to the actual elements. In relation to this, we performed two processes to remove some of this noise. First, we performed voxel down-sampling [39,40] to reduce the number of points and storage capacity and then filtered out the point cloud erroneously restored in duplicate positions. We adopted statistical outlier removal [41] on the down-sampled point cloud. The algorithm analyzes the distance from each point to its neighboring points. Points that exceed a threshold calculated based on the mean and the standard deviation of the distances are considered as outliers to be removed. The resulting elements were saved in file format (Section 4.3) to enable their transmission to the map manager.

4.3. Element Representation

This section describes the data structure designed to store the elements acquired in Section 4.2. We refer to these elements as "extracted elements", which serve as the reference targets for the subsequent map generation or updates. We divided the map elements into two groups: continuous road markings (e.g., lane markings) and discrete or pole-like objects (e.g., traffic signs) [42]. The elements were stored in the HDFS in the format described in Table 1.

Table 1. Format of the road elements.

Continuous road marking: E_{con}
Vehicle_pose: [p_x, p_y, p_z, Roll, Pitch, Yaw]
Uncertainty matrix: U (3 × 3 matrix)
Raw point cloud: [ply file]
Discrete pole-like object: E_{dis}
Vehicle_pose: [p_x, p_y, p_z, Roll, Pitch, Yaw]
Uncertainty matrix: U (3 × 3 matrix)
Element_center: [x,y,z]
Raw_cropped_image: (int) Image[width][height]
Object_label: [(string) label]

For continuous road marking, the entire point cloud after noise reduction is saved. For discrete objects, the estimated center point of the object and 2D image of the object are stored together through the point cloud, allowing each object to be distinguished. Additionally, the vehicle pose at the time of data collection is saved along with the element to enable map manager to know the location where the element was collected. This enables the selection of the optimal vehicle for data collection during future updates.

5. Map Management

This section describes the overall techniques used for management. To divide the entire region, the map was generated into manageable units, and the region was divided into square grids similar to the form of a grid map. Each square grid is defined herein as a partial map. The grid's reference point is defined using the East–North–Up coordinate system [43] that employs the central origin of UTM-K [44] as (0,0,0), which is the coordinate standard of Korea's precise map. ENU coordinate system is a local tangent plane coordinate used to represent the positions of surrounding objects relative to a specific point. Each partial map measured 10 m in both the east and north directions. This size was chosen to allow all elements within the partial map's range to be identified from each grid vertex considering the maximum distance (i.e., 15 m) at which the disparity of the consumer-grade stereo camera is maintained.

The Map State Table used by the collection manager to select vehicles for the data collection was defined. The states that the partial map can possess were also delineated to guide the behaviors of both the collection manager and the map manager. Additionally, the state transition of the partial map was composed in the form of a finite-state machine to define the map's life cycle.

The elements extracted by the extraction container were assigned to each partial map based on the location of the vehicle that collected them. This allows for a convenient data selection when a specific partial map expires, and an update is required, because the vehicles for the data collection can be selected based on the range of the relevant partial map. The map manager's activity was designed based on the partial map's lifecycle.

5.1. Map State Table

The collection manager utilizes the map state table to manage the partial maps states. The map state table is a key-value database that uses the partial map origin coordinates as the key and the map information as the value. Table 2 is the form of the Map State Table.

Table 2. Key and variables of the map state table.

Key: [(int) PartialMapCoordinates_X, (int) PartialMapCoordinates_Y]
Value: (enum) State Not_Active = 0 Construction = 1 Accessible = 2 Stable = 3 Expired = 4 (timestamp) LastUpdateTime (list) Elementlist [(int) element_ID, (enum) element_State, (list) ElementFOV] (enum) element_State Construction = 0 Stable = 1 Expired = 2 (float32) dataRate

The State represents the status of the corresponding partial map. The ElementList indicates the element information and status within the partial map, allowing the collection manager to determine whether or not to request data collection for the vehicle. The map expiration is determined based on the LastUpdateTime.

The ElementFOV variable contains information on the observed position range and statistics of an element, minimizing the duplicate data collection for areas that had already been explored and increasing the data collection efficiency. The rate of the pose data collected from the connected car is stored to determine whether to request data collection on vehicles located in the "not active" state partial map.

5.2. Life Cycle of a Partial Map

The finite-state machine depicted in Figure 5 presents the life cycle of a partial map.

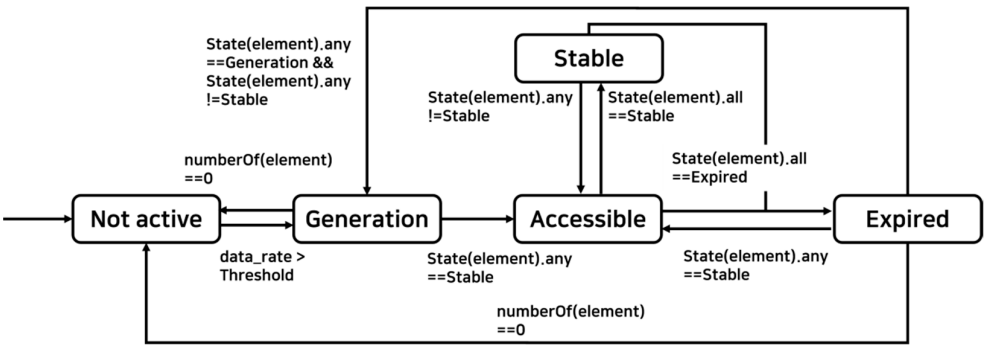


Figure 5. Finite-state machine of a partial map.

The initial state of each partial map is called the "not active" state. This state indicates that no elements belong to it yet due to the lack of data collected through the vehicle. The collection manager monitors the vehicles within the map's coverage area. When the number of vehicles with valid data within the map range exceeds the threshold per unit time, the state is changed to "map construction", and the vehicle is requested to transmit data. If there is even one element in the partial map whose position distribution remains below the threshold, that element is deemed stable, and its status is changed to "accessible." If all the partial map elements have a position distribution below the threshold, the map status is changed to "stable", and it no longer requests data from the collection manager. If the status is changed to "expired" after a certain period [10] has passed since the last update of the existing elements and the partial map. This change in status prompts a request for data collection to update them. The collection manager checks the update status for each element in the partial map and requests data collection from vehicles that can observe the expired elements. When an update is completed, and the elements with a position distribution below the threshold are identified, the partial map transitions to the "accessible" state. These elements are then registered in the partial map database. The "expired" state reverts to the "construction" state if new elements, which do not align with the existing elements, are observed from the collected data. Data collection is then requested from vehicles capable of observing these elements until they obtain a position accuracy below the threshold, or until no further observational data are available, leading to the expiration of new elements.

5.3. Map Management

This section explains the method of determining the "ElementFOV" parameter that represents the range within which a vehicle can acquire an element among the variables stored in the map state table. This parameter helps collect only the necessary data for updates by determining whether or not the data collected from the vehicles for map updates are observing an existing element. ElementFOV is assigned to each map element and comprises the convex hull (series of points) formed by the location, where the extracted element constituting the map element was collected, the maximum angle formed by the extracted element and the vehicle heading, and the range of the element's and coordinates based on the vehicle's coordinate system. The convex hull [45] represents the location range from which the element can be collected based on the statistics of the positions where the extracted element was gathered. The maximum angle between the extracted element and the vehicle's heading at the time of collection allows the determination of whether the element exists within the range perceivable by the sensor and the inference network. Lastly, the range of the elements and coordinates in the vehicle-based coordinate system allows the determination of the element's position relative to the vehicle when it was detected. If no vehicle is available to collect data within the convex hull, this variable is used to expand the range of vehicles that can collect the element. When a specific element expires, the map manager determines which vehicles can observe that element through ElementFOV. Figure 6 visualizes the variables constituting ElementFOV.

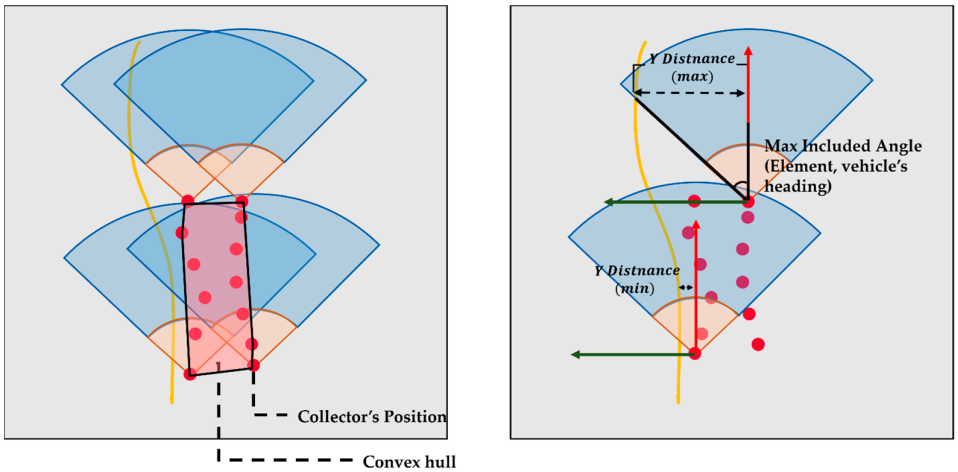


Figure 6. ElementFOV variables.

6. Map Generation

This section describes the overall map generation algorithms. The map manager processes the "extracted elements" collected from the vehicles in each partial map with the "Map Construction" or "Accessible" state under three stages. In the first stage, an analysis utilizing the morphological characteristics of the extracted elements is conducted to filter out the elements produced by the incorrect inference or stereo matching. In the second stage, the elements consecutively collected from the same vehicle are used to correct the posture at the time of collection. In the third and final stage, "map elements" are generated for registration in the partial map by performing matching and regression with the elements collected from different vehicles.

6.1. Outlier Filtering

The first step involves an analysis to filter out the false-positive or incorrectly restored elements. This is applied differently to discrete objects and continuous road markings. Plane segmentation [46] for discrete objects is performed by taking advantage of their surface-like form. For continuous road markings, elements that are not line-like features are filtered out to remove the incorrectly acquired elements. The decision-making process for the continuous road marking removal is as follows: (1) the element points are sorted in order of vehicle proximity, and (2) an oriented bounding box [47] is created encompassing the first point and the subsequent n points. We used the length of the short edge of this box to determine the point dispersion and compare the long edge direction with the longest edge direction of the oriented bounding box that encompasses the element. Table 3 is the pseudo code of this filtering process. We applied a filtering process to the elements based on the following criteria:

1. If the average length of the short edge of the oriented bounding box is more than twice the lane width.
2. If the variance of the short edge length of the oriented bounding box for an element is greater than twice the variance observed across all elements.
3. If the elements for which the direction of their encompassing oriented bounding box deviates by more than 45° from the vehicle's heading.

Table 3. Pseudo code of the filtering process.

```
//Definitions
e_obb: oriented_bounding_box(element)
w_obb[i]: oriented_bounding box(elemenI.pIints(I: I + window_size)
sort_by_axis(): ascending order sorting function along the given axis in the vehicle-centric
coordinate
t_angle_diff: threshold_angle_difference

/ / Check angle difference with vehicle heading
if abs(vehicle_heading–atan(e_obb.long_edge/e_obb.short_edge)) > t_angle_diff:
discard()
else:
sort_by_axis(element.points, x)
  length_list = []
for I in range(num(element.points)-window_size):
length_list.append(w_obb[i].short_edge)

If mean(length_list) > lane_width * threshold_widness OR
mean(length_list) + std_dev(length_list) > lane_width *
threshold_max_widness:
discard()
else:
  keep()
```

6.2. Yaw Correction

The second step involves aligning the vehicle's heading (yaw) with the extracted elements obtained from a single vehicle, which is done using the continuity of the road markings (lanes) to correct the yaw errors. This process aims to compensate for the measurement errors caused by the reduced heading accuracy of the GPS or the sensor drift in the IMU. The error is adjusted using the difference obtained when two consecutive observations made in a short time interval yield different results for the same continuous element. This process is performed under the assumption that the yaw error exhibits similar values within the partial map and under the condition that elements E_n and E_{n-1}, which are observed at consecutive moments, are the same element. Thus, after performing the yaw error correction, the vector starting from point P_n in E_n and ending at point P_{n-1} in E_{n-1} should be parallel to $\overrightarrow{l}_t$ when linearly approximating E_t with $\overrightarrow{l}_t + P_t$ (Equation (1)).

$$R(\theta_b) = \text{rotation matrix of the yaw error value}$$
$$E_t = \text{Element observed at time t; linearly approximated to } \overrightarrow{l}_t + P_t$$
$$R(\theta_t) = \text{ rotation matrix of the vehicle}'\text{s yaw angle at time "t"}$$
$$O_t = \text{vehicle position at time "t"}$$

$$P_t = \text{arbitrary point belonging to element } E_t \text{ in the global coordinate system at time t}$$

$$P_{t,vehicle} = \text{'}P'_t \text{ represented in the vehicle}'\text{s coordinate system}$$
$$k = \text{scale variable}$$

$$kR^T(\theta_b)R(\theta_n)\overrightarrow{l}_n = R^T(\theta_b)R(\theta_n)P_{n,vehicle} + O_n - R^T(\theta_b)R(\theta_{n-1})P_{n-1,vehicle} - O_{n-1} \quad (1)$$

Figure 7 illustrates the yaw correction process.

The abovementioned process can show significant variations due to external factors (e.g., lane inference results or restoration quality). Moreover, it cannot be applied in intervals where the pose is updated using a Kalman filter. Considering this, the algorithm is applied to elements for which the short edge of the oriented bounding box encompassing the element is 0.25 m or less, which is twice the lane width. The average value of the drift errors that fall within one standard deviation is chosen as the yaw drift error.

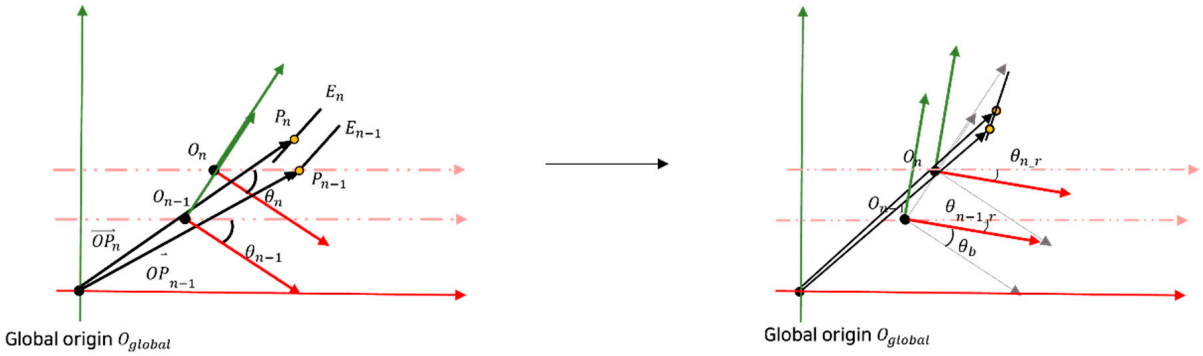


Figure 7. Yaw error correction based on a continuous element.

6.3. Element Matching

After the filtering process and the pose correction, the elements are matched and ultimately registered in the partial map. This process involves matching the elements collected from the same vehicle and matching those collected from different vehicles. For the latter, matching is performed after matching the elements from the same vehicle. The vehicle's position uncertainty matrix at the time of collection and the model derived from regressing each element are used. The uncertainty matrix represents the spatial distribution of the collected element, while the regression model calculates the translational relationship between the two elements being matched. In a single partial map, the matching process is conducted by starting with the elements having the lowest uncertainty matrix. During this process, the elements with overlapping spatial distributions and similar regression models are identified and selected as the matching candidates.

1. Elements with overlapping spatial distributions are found through the following process: the relationship between the elements is easily determined by approximating the spatial distribution based on the uncertainty matrix value at the time of collection using a pixel grid with 0.05 m resolution and storing it in a table referred to as the "candidate table". The existence of overlapping elements for all extracted elements from the partial map is checked against their approximated spatial distribution in the "candidate table" using the pixel grid. Nonexistence will lead to the element's registration of its spatial distribution in the candidate grid, while existence leads to the performance of matching with that element. If the spatial distribution overlaps with multiple elements, matching is performed with the element having a wider overlapping spatial distribution. This is determined by introducing the score based on distance from each element, as in Equation (2).

$$\begin{gathered} \text{Score} = \sum_0^n \frac{1}{distanceFromE_1 * distanceFromE_2} \\ \text{n} = \text{number of pixels overlapped} \end{gathered} \tag{2}$$

The elements with a distribution below the threshold are fixed in the candidate grid and stored in the partial map, no longer undergoing any matching process. After executing the abovementioned process for all the extracted elements in the partial map, unstable elements with a regression score lower than the mean score of all the unstable elements are removed from the element list. We adopted the R-squared score [48] to evaluate the elements' regression model. Figure 8 presents the flowchart of finding elements with position distributions that overlap with each other.

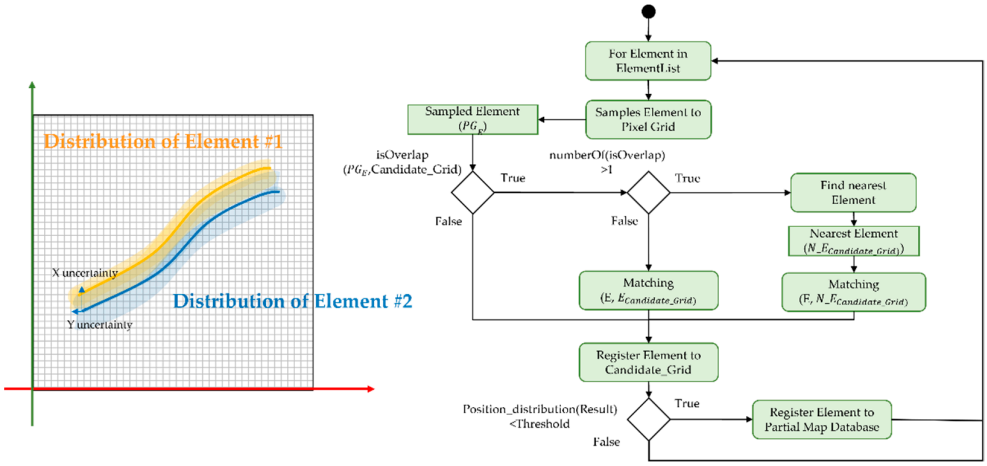


Figure 8. Flowchart for finding elements with overlapping position distributions.

2. We match two target elements by first estimating their relative positioning by calculating the distance between them. First, the distance between the two elements is calculated to estimate their positional relationship. In this step, a quadratic regression model of the element, which is derived from its constituent points, is used to approximate the lane shape and represent its curvature [49]. Vector $\overrightarrow{d(P_{E_1,t}, E_2)}$ is defined to determine the displacement between the two elements, E_1 and E_2. This vector originates from point $P_{E_1,t}$ in E_1 and terminates at a point in the regression model R_2 of E_2 orthogonal to R_2 at that point. Vector d is calculated at regular intervals for each element. Considering the potential errors stemming from the reconstruction quality of the element, $\overrightarrow{displacement(E_1, E_2)}$, the displacement between the two elements is determined through Equation (7) using a weighted average based on the standard deviation and considering both the mean and the standard deviation of the displacement.

$$\mathrm{E}\left(\overrightarrow{d(P_{E_1,t}, E_2)}\right) = \frac{\sum_{t=0}^{n} \overrightarrow{d(P_{E_1,t}, E_2)}}{n} \tag{3}$$

$$\sigma\left(\overrightarrow{d(P_{E_1,t}, E_2)}\right) = \sqrt{\frac{\sum_{t=0}^{n} \left\{\overrightarrow{d(P_{E_1,t}, E_2)} - \mathrm{E}\left(\overrightarrow{d(P_{E_1,t}, E_2)}\right)\right\}^2}{n}} \tag{4}$$

$$\mathrm{E}\left(\overrightarrow{d(E_1, P_{E_2,t})}\right) = \frac{\sum_{t=0}^{m} \overrightarrow{d(E_1, P_{E_2,t})}}{m} \tag{5}$$

$$\sigma\left(\overrightarrow{d(E_1, P_{E_2,t})}\right) = \sqrt{\frac{\sum_{t=0}^{m} \left\{\overrightarrow{d(E_1, P_{E_2,t})} - \mathrm{E}\left(\overrightarrow{d(E_1, P_{E_2,t})}\right)\right\}^2}{m}} \tag{6}$$

$$\overrightarrow{displacement(E_1, E_2)} = \frac{\mathrm{E}\left(\overrightarrow{d(P_{E_1,t}, E_2)}\right) \times \sigma\left(\overrightarrow{d(E_1, P_{E_2,t})}\right)^2 - \mathrm{E}\left(\overrightarrow{d(E_1, P_{E_2,t})}\right) \times \sigma\left(\overrightarrow{d(P_{E_1,t}, E_2)}\right)^2}{\sigma\left(\overrightarrow{d(E_1, P_{E_2,t})}\right)^2 + \sigma\left(\overrightarrow{d(E_1, P_{E_2,t})}\right)^2} \tag{7}$$

Lastly, the parallel translation value for matching each element is determined using the element's uncertainty matrix "U" as a weight according to Equation (8).

$$\mathrm{Translation}(E_1) = \left(\frac{\overrightarrow{displacement(E_1, E_2)}_1 \times U_{11,E_2,}}{U_{11,E_1,} + U_{11,E_2,}}, \frac{\overrightarrow{displacement(E_1, E_2)}_2 \times U_{22,E_2,}}{U_{22,E_1,} + U_{22,E_2,}}\right) \tag{8}$$

After translating each element, quadratic regression is re-executed on the points constituting the two elements to recalculate the model and the root mean square error (RMSE) score, finalizing the matching. The position distribution is then updated to the weighted average variance using the uncertainty of the two elements as weights. If the updated position distribution and the RMSE of the regressed element are within a threshold, the regressed element, which is the matching output, is registered to the partial map as a "map element". If this condition is not met, the process is repeated with the other elements in the vicinity with overlapping position distributions.

3. When matching elements collected from the same vehicle, the elements that overlap at consecutive moments are considered as the same elements for matching. Accordingly, the process of finding overlapping elements is performed. However, instead of considering the uncertainty, only the points constituting the element are approximated to the pixel grid to construct the candidate table. The process of finding the positional relationship between two elements observed in the same lane is omitted, directly proceeding with quadratic regression.

7. Implementation and Evaluation

7.1. Experiment Environment

Tables 4 and 5 present the environment of this study.

Table 4. Hardware environment.

GNSS	Stereo Camera and IMU	Mobile Communication	Collection LAPTOP	On-Premises Cloud
SPARKFUN ZED-F9P RTK	ZED2 Stereo System	Snapdragon 8 Gen 1 SM8450	i5-8400 GTX1050	i9-12900 RTX3090

Table 5. Software environment.

Messaging Platform	Traffic Sign Detection	Lane Detection	Point Cloud Library
Apache Kafka	Traffic-Sign Detection [50]	Global Association Network [51]	Open3d [52]

Data collection for map generation was carried out along the trajectory shown in Figure 9. It was conducted six times clockwise and three times counterclockwise, taking a total of 31 min from 3:52 PM to 4:32 PM. Data were gathered at rates of 200 Hz for IMU, 1 Hz for the GNSS, and 10 Hz for images. The following is a brief description of the neural network used in the implementation.

- Traffic-Sign Detection

Traffic-Sign Detection [50] is an object detection system that utilizes the Region - based Fully Convolutional Network (R-FCN) detection framework. The model utilizes weight parameters obtained from transfer learning on the German Traffic Sign Benchmark dataset [53], building upon a base model that was pre-trained using the Microsoft COCO dataset [54]. When using Resnet101 [55] as the backbone network, the model exhibited a mean Average Precision (mAP) of 95.15 and a total execution of 85.45 ms on an NVIDIA Titan Xp.

- Global Association Network

The Global Association Network (GANet) [51] is a network that performs lane detection from forward-facing images. The model consists of a feature learning part, a lane-aware feature aggregator (LFA), and two decoder heads formed by FCNs. The results from both heads are combined to accurately reconstruct the lanes. The feature learning part consists of a Convolutional Neural Network (CNN) backbone, a self-attention layer, and a Feature

Pyramid Network (FPN) neck to extract multi-level representations from the input image. The decoder heads, composed of the keypoint head and offset head, utilize output of the feature learning part to generate confidence maps and offset maps respectively. LFA positioned before the keypoint head enhances local correlations between adjacent keypoints to aid in the reconstruction of continuous lane lines. In the lane reconstruction process, the starting point of the lane is acquired from the offset map and set as the center of the lane cluster. Using a combination of the confidence map and the offset map, keypoints belonging to the same lane are obtained. The lane is then constructed by clustering keypoints around the starting point. Using ResNet-34 [55] as the Backbone Network, the model trained using Tusimple Dataset [56] showed a processing time of 7.8 ms and F1 score 97.68 when tested on an NVIDIA Tesla V-100.

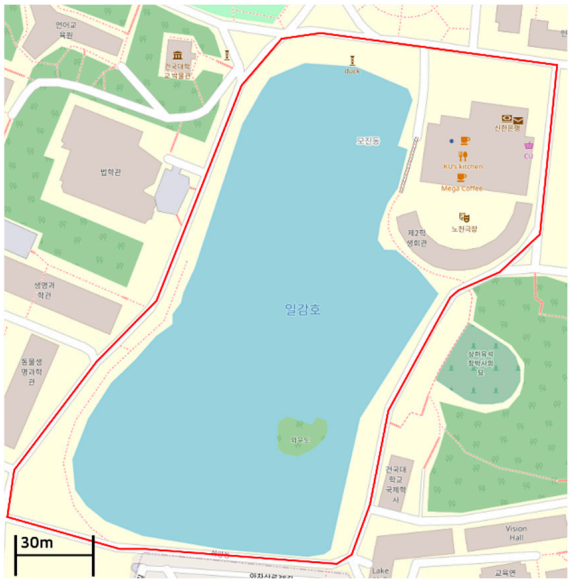


Figure 9. The trajectory driven for data collection.

7.2. Implementation Result

Figure 9 visualizes the raw extracted elements are plotted in the ENU coordinate system. Filtering, yaw correction, and matching are performed on these elements to process them into a lane-level map. In total, 192 partial maps were generated, consisting of 12,545 extraction elements.

- Filtering Extracted Element

Figure 10 depicts the filtering process of removing incorrectly detected elements. The incorrectly detected elements were colored in red.

- Yaw Correction

Figure 11 shows elements after yaw correction.

- Element Matching

The final map is constructed by aligning each element of the map. The point cloud of the matched element is displayed in Figure 12.

Figure 13 displays elements from the matching results with a position accuracy of less than 0.25 m, and when the point cloud forming the element is regressed, it has an RMSE of less than 0.25 m. The accuracy criterion used is based on the quality standard of 0.25 m for the South Korean HD map [42].

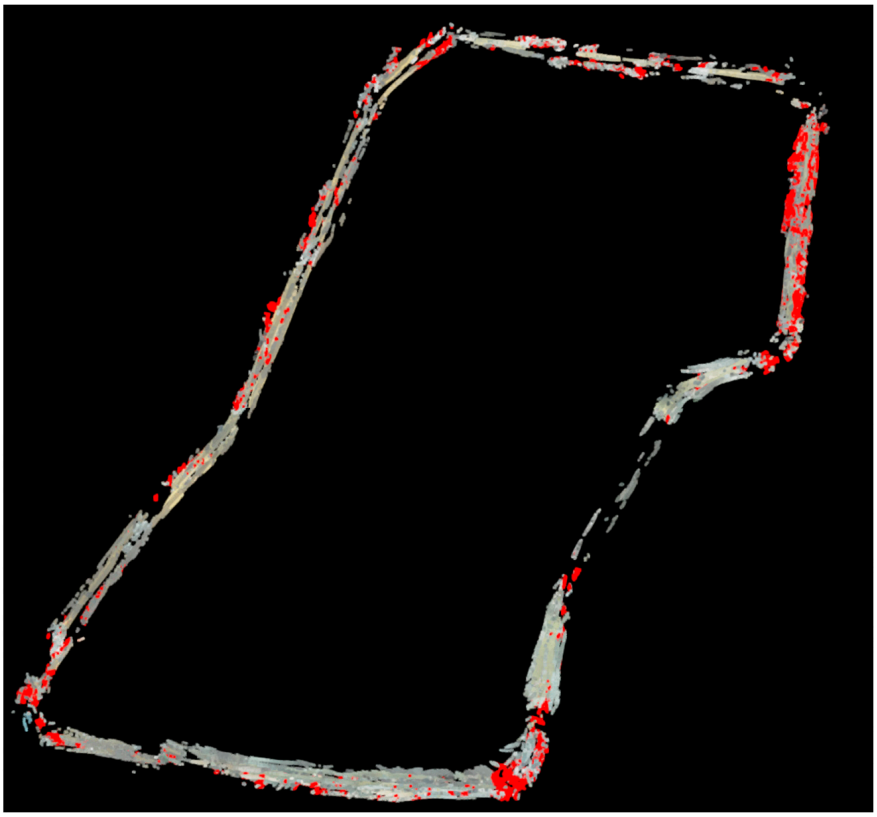


Figure 10. Result of outlier filtering.

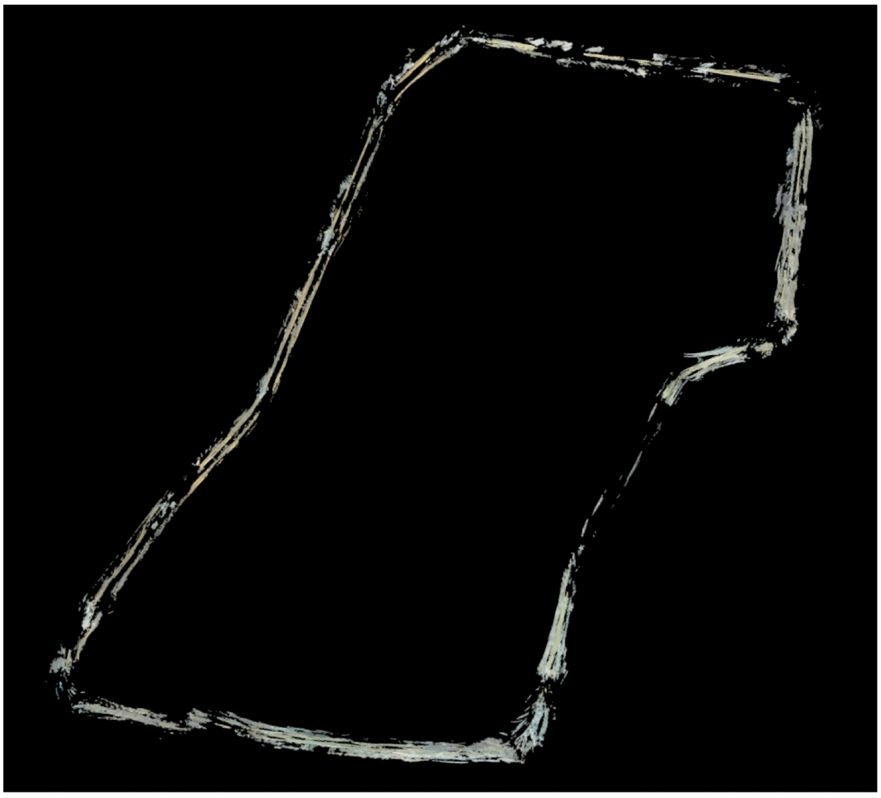


Figure 11. Result of yaw correction.

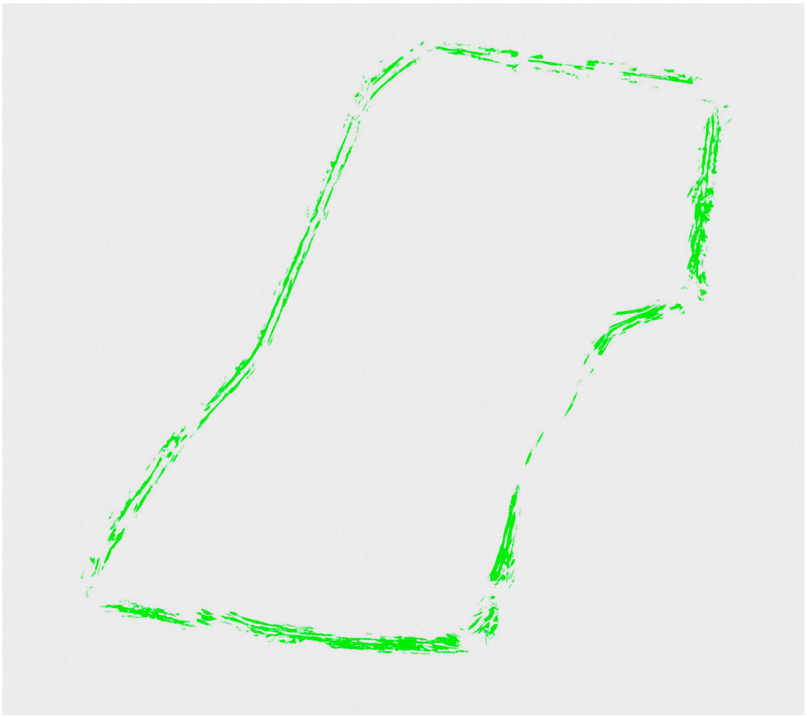


Figure 12. The point cloud of matched element.

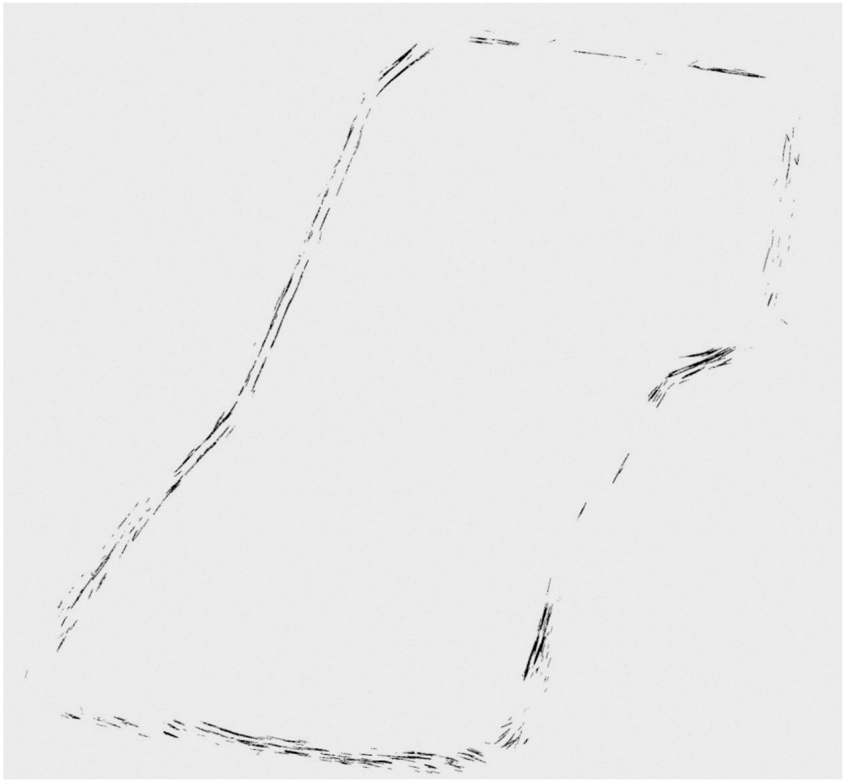


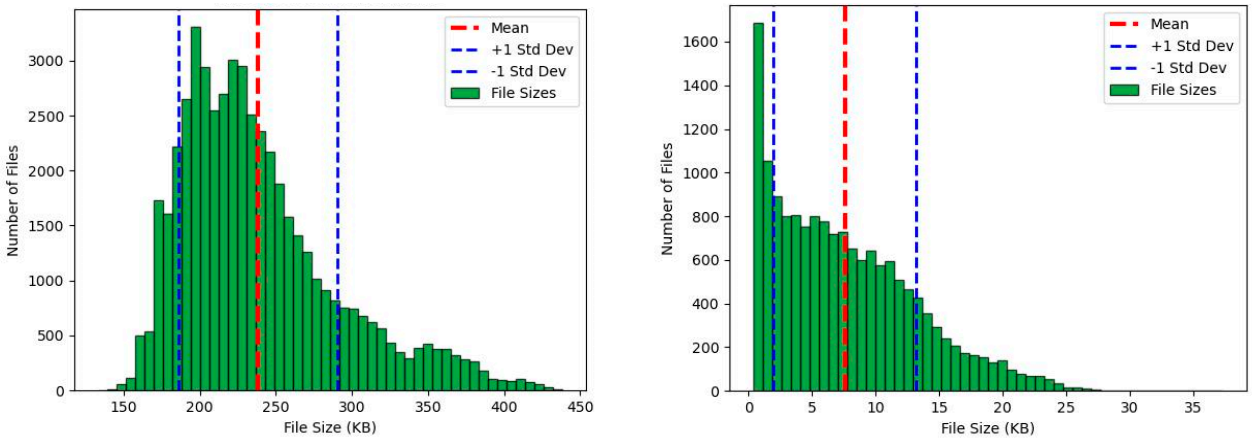
Figure 13. The result of regressed elements.

The performance metrics are described in Table 6: the size of image data transmitted from the vehicle, the size of extracted elements, and the processing time for each step were selected as indicators.

Table 6. Performance metrics.

Type of Performance Metric		Average	Maximum	Deviation
Data size of transmitted image		238.42 KB	438.51 KB	52.2 KB
Size of each Extracted Element		7.58 KB	37.36 KB	5.61 KB
Processing time to generate Extracted Element	Overall time	0.306 s	0.585 s	0.028 s
	Inference	0.028 s	0.173 s	0.004 s
	Cropping	0.001 s	0.004 s	<0.001 s
	Reconstruction	0.217 s	0.285 s	0.024 s
Generation processing time of each partial map	Overall time	2.858 s	53.78 s	6.777 s
	Element Loading	0.123 s	0.166 s	0.170 s
	Filtering and Yaw Correction	0.338 s	2.772 s	0.391 s
	matching	2.390 s	52.400 s	6.422 s

Figures 14 and 15 are graphs plotted based on the metrics in Table 6. Figure 14a is a graph of the "data size of transmitted image", and Figure 14b is a graph of "size of each extracted element". Figure 15a,b shows the plotted result of the processing time of the extracted element and the generation processing time of each partial map, respectively.



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

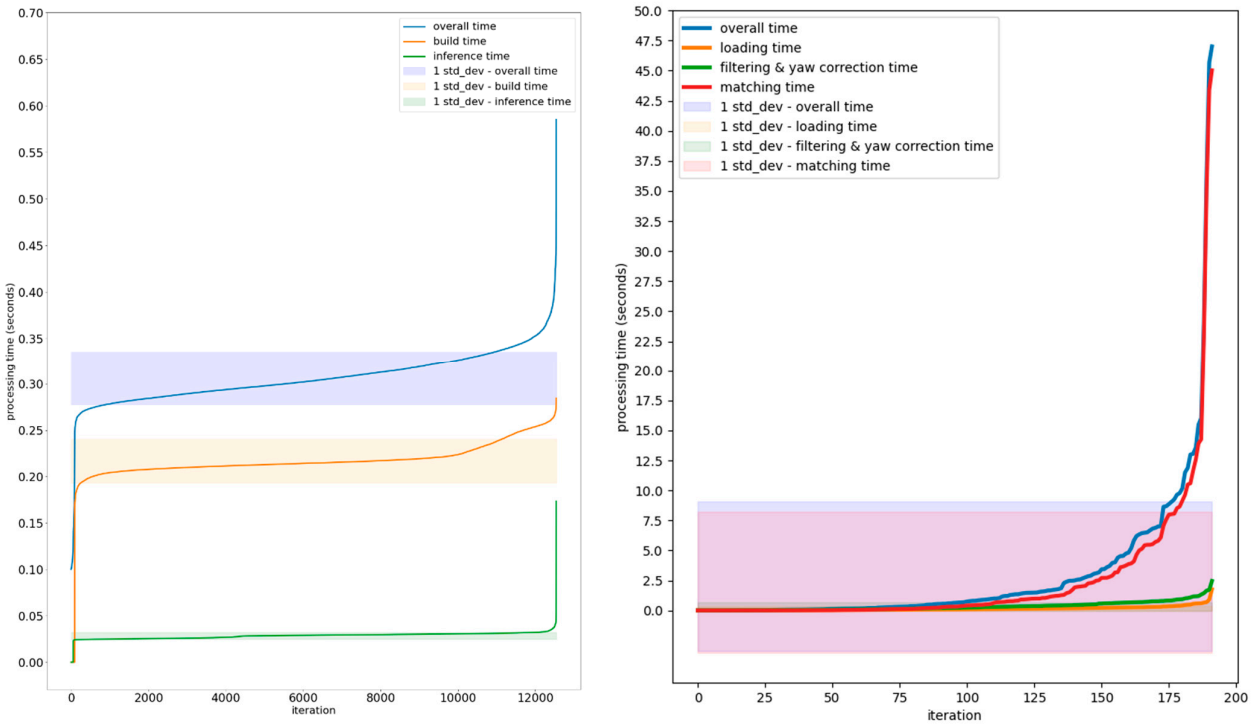
Figure 14. (**a**) Data size of the transmitted image. (**b**) Size of each extracted element.

7.3. Discussion

The following points were discussed regarding the results.

- Data Transmission and Extraction

The average upload speed of the 5G Cellular Network is 44.1 Mbps [35], which is sufficiently fast to receive stereo image capture, requiring an average bandwidth of 37.3 Mbps at 10 Hz transmission. By cropping the area where the lane is located for transmission and receiving streaming based on H.264 [57], it is expected that the required bandwidth can be further reduced. The data extraction time averaged 0.306 s, with the reconstruction process taking the longest. This process utilized the stereo matching function process of Opencv [58], which does not benefit from GPU hardware acceleration. In the future, by using Nvidia's NVM stereo matching function [59], processing time could be drastically reduced.



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

Figure 15. (**a**) The processing time to generate extracted element. (**b**) Generation processing time of each partial map.

- Accuracy of the Map

When position errors occurred due to the performance limitations of GNSS and vibrations from the IMU, two major factors significantly affected the accuracy of the map. The first was the inaccuracy of the uncertainty matrix, making it difficult to assess the accuracy of its elements' position. The second was that during the update of the Kalman filter, smoothing was not applied to the previous position, resulting in no error correction of the predicted location. This can potentially be overcome by using smoothing algorithms like unscented Kalman smoothing (UKS) [60]. Additionally, accuracy was compromised by linear outliers such as road signs, pavements, curbs, and parking lots. This issue can be improved by applying constraints regarding the shape of the lane to the filtering.

8. Conclusions

This study presented a framework that collects sensor data to generate lane-level maps in the cloud, extracts road elements from the collected data, and uses these extracted elements for map generation and management throughout the entire process. The framework was designed considering the data collection situation during sensor data acquisition, collected data structure, and data flow. We analyzed the size and the frequency of the sensor data collected from the vehicles to determine whether it can be applied at the current level of the mobile communication technology. The size of image data is 350 KB on average, which is sufficiently small to upload stereo image data at 10 Hz using the commercially available 5G upload link bandwidth (Average of 5G bandwidth: 44.1 MB, required bandwidth: 37.3 Mbps). The data are received via cellular communication; hence, their quality can be degraded. Accordingly, we designed an algorithm for the noise and outlier processing of 3D restored elements and provided quantitative metrics on the time required to compose the road elements and the data size of the extracted elements. We managed the map by logically partitioning the space to form a "partial map", which is a smaller map unit, and established its state transition. This simplified the task of selecting vehicles that would collect the sensor data necessary for map creation and management. Our devised mapping

method consisted of filtering considering the road lane characteristics, pose correction of the collecting vehicle, and a matching phase for generating the lane-level map. We validated the method performance by utilizing the directly collected data to create a map, presenting qualitative results. Using the proposed technique, we were able to generate 651 map elements with a position accuracy of less than 0.25 m and regression result's RMSE of less than 0.25 m from over 12,000 extracted elements in 192 partial maps. To achieve this, an average of 0.31 s per element was taken for the creation of extracted elements, with a maximum time of 0.585 s. To generate map elements from each partial map, it took an average of 2.88 s and a maximum of 53.8 s. This performance doesn't guarantee real-time map generation as data is collected. However, considering that GPU acceleration was not utilized for the stereo matching of images and every process was conducted on a single workstation, we anticipate that future advancements in the system architecture could reduce processing time. Finally, by storing statistics on the situation in which the element was collected, we proposed a technique for determining from which vehicle the data can be collected to update the map elements.

However, there are several limitations of our approach to be advanced in the future:

- Limitations of map generation methods using the geometric properties of the elements.

We were unable to determine whether to perform matching based on the similarity between elements or to treat some as outliers and remove them from the matching pool. It's necessary to enhance the element matching techniques and find ways to reduce the impact of incorrect element matching. Methods include checking the similarity of elements and introducing techniques such as non-maximum suppression.

- The map area strictly divided using a grid.

The approach of strictly dividing the map area into grid shapes made it difficult to restore elements collected at the boundaries of the grids. It is necessary to divide the map region in a fuzzy manner so that a single element can be managed across multiple grid maps. More meaningful element groups should be created than simply dividing the map with grids.

- Unfiltered outlier elements.

The lane detection DNN can recognize outliers with a linear shape, such as continuous lines (pavement markings), crosswalks, and stop lines. By utilizing contexts like road types and vehicle behavior, it is possible to reduce incorrect element detection or mismatches in element matching.

Author Contributions: Conceptualization, J.K. and C.M.; methodology, J.K. and J.M.; software, J.K. and J.M.; validation, J.K. and J.M.; formal analysis, J.K.; investigation, J.K. and J.M.; resources, C.M.; data curation, J.K. and J.M.; writing—original draft preparation, J.K.; writing—review and editing, J.K., J.M. and C.M.; visualization, J.K.; supervision, C.M.; project administration, J.K.; funding acquisition, C.M. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0020536, HRD Program for Industrial Innovation).

Data Availability Statement: Data is available from the authors.

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

References

1. Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.N.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robot.* **2008**, *25*, 425–466. [CrossRef]
2. Diaz-Diaz, A.; Ocaña, M.; Llamazares, Á.; Gómez-Huélamo, C.; Revenga, P.; Bergasa, L.M. HD maps: Exploiting OpenDRIVE potential for Path Planning and Map Monitoring. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany, 5–9 June 2022; pp. 1211–1217. [CrossRef]

3. Elhashash, M.; Albanwan, H.; Qin, R. A Review of Mobile Mapping Systems: From Sensors to Applications. *Sensors* **2022**, *22*, 4262. [CrossRef] [PubMed]
4. Zhang, P.; Zhang, M.; Liu, J. Real-time HD map change detection for crowdsourcing update based on mid-to-high-end sensors. *Sensors* **2021**, *21*, 2477. [CrossRef]
5. Dabeer, O.; Ding, W.; Gowaiker, R.; Grzechnik, S.K.; Lakshman, M.J.; Lee, S.; Reitmayr, G.; Sharma, A.; Somasundaram, K.; Sukhavasi, R.T.; et al. An End-To-End System for Crowdsourced 3D Maps for Autonomous Vehicles: The Mapping Component. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 21 September 2017; pp. 634–641. [CrossRef]
6. Massow, K.; Kwella, B.; Pfeifer, N.; Häusler, F.; Pontow, J.; Radusch, I.; Hipp, J.; Dölitzscher, F.; Haueis, M. Deriving HD maps For Highly Automated Driving from Vehicular Probe Data. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1 November 2016; pp. 1745–1752. [CrossRef]
7. Doer, C.; Henzler, M.; Messner, H.; Trommer, G.F. HD Map Generation from Vehicle Fleet Data for Highly Automated Driving on Highways. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 2014–2020. [CrossRef]
8. Kim, K.; Cho, S.; Chung, W. HD Map Update for Autonomous Driving With Crowdsourced Data. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1895–1901. [CrossRef]
9. Use-Case-Implementations-for-Sensor-Data-Sharing. Available online: https://5gaa.org/content/uploads/2023/02/5gaa-t-220019-use-case-implementations-for-sensor-data-sharing.pdf (accessed on 14 August 2023).
10. ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and Guidance on Standardization*; ETSI TR 102 863, V1.1.1; ETSI: Sophia-Antipolis, France, 2011.
11. Almheiri, F.; Alyileili, M.; Alneyadi, R.; Alahbabi, B.; Khan, M.; El-Sayed, H. Evolved Local Dynamic Map (eLDM) for Vehicles of Future. In Proceedings of the 2021 6th International Conference on Computational Intelligence and Applications (ICCIA), Xiamen, China, 11–13 June 2021; pp. 292–297. [CrossRef]
12. Al Mojamed, M. On the Use of LoRaWAN for Mobile Internet of Things: The Impact of Mobility. *Appl. Syst. Innov.* **2022**, *5*, 5. [CrossRef]
13. Di Renzone, G.; Parrino, S.; Peruzzi, G.; Pozzebon, A. LoRaWAN in Motion: Preliminary Tests for Real Time Low Power Data Gathering from Vehicles. In Proceedings of the 2021 IEEE International Workshop on Metrology for Automotive (MetroAutomotive), Bologna, Italy, 1–2 July 2021; pp. 232–236. [CrossRef]
14. IEEE 802.11p. Available online: https://standards.ieee.org/ieee/802.11p/3953/ (accessed on 30 August 2023).
15. LTE C-V2X, 3GPP TS36.300. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2430 (accessed on 30 August 2023).
16. Sutdy on NR C-V2X, 3GPP TS.38.885. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3497 (accessed on 30 August 2023).
17. Lien, S.Y.; Deng, D.J.; Lin, C.C.; Tsai, H.L.; Chen, T.; Guo, C.; Cheng, S.M. 3GPP NR Sidelink Transmissions Toward 5G V2X. *IEEE Access* **2020**, *8*, 35368–35382. [CrossRef]
18. Dhinesh, K.R.; Rammohan, A. Revolutionizing Intelligent Transportation Systems with Cellular Vehicle-to-Everything (C-V2X) Technology: Current Trends, Use Cases, Emerging Technologies, Standardization Bodies, Industry Analytics and Future Directions. *Veh. Commun.* **2023**, *43*, 100638. [CrossRef]
19. Cheng, N.; Lyu, F.; Chen, J.; Xu, W.; Zhou, H.; Zhang, S.; Shen, X. Big Data Driven Vehicular Networks. *IEEE Netw.* **2018**, *32*, 160–167. [CrossRef]
20. Marosi, A.C.; Lovas, R.; Kisari, Á.; Simonyi, E. A novel IoT platform for the era of connected cars. In Proceedings of the 2018 IEEE International Conference on Future IoT Technologies (Future IoT), Eger, Hungary, 18–19 January 2018; pp. 1–11. [CrossRef]
21. Yoo, A.; Shin, S.; Lee, J.; Moon, C. Implementation of a Sensor Big Data Processing System for Autonomous Vehicles in the C-ITS Environment. *Appl. Sci.* **2020**, *10*, 7858. [CrossRef]
22. Alexakis, T.; Peppes, N.; Demestichas, K.; Adamopoulou, E. A Distributed Big Data Analytics Architecture for Vehicle Sensor Data. *Sensors* **2023**, *23*, 357. [CrossRef]
23. Rocha, D.; Teixeira, G.; Vieira, E.; Almeida, J.; Ferreira, J. A Modular In-Vehicle C-ITS Architecture for Sensor Data Collection, Vehicular Communications and Cloud Connectivity. *Sensors* **2023**, *23*, 1724. [CrossRef]
24. Lee, K.; Hong, D.; Kim, J.; Cha, D.; Choi, H.; Moon, J.; Moon, C. Road-Network-Based Event Information System in a Cooperative ITS Environment. *Electronics* **2023**, *12*, 2448. [CrossRef]
25. Kim, C.; Cho, S.; Sunwoo, M.; Jo, K. Crowd-Sourced Mapping of New Feature Layer for High-Definition Map. *Sensors* **2018**, *18*, 4172. [CrossRef] [PubMed]
26. Li, Q.; Wang, Y.; Wang, Y.; Zhao, H. HDMapNet: An Online HD Map Construction and Evaluation Framework. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 4628–4634. [CrossRef]
27. Lee, J.; Lee, K.; Yoo, A.; Moon, C. Design and Implementation of Edge-Fog-Cloud System through HD Map Generation from LiDAR Data of Autonomous Vehicles. *Electronics* **2020**, *9*, 2084. [CrossRef]

28. Shin, S.; Kim, J.; Moon, C. Road Dynamic Object Mapping System Based on Edge-Fog-Cloud Computing. *Electronics* **2021**, *10*, 2825. [CrossRef]
29. Apache Hadoop. Available online: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (accessed on 14 August 2023).
30. UML Activity Diagram. Available online: https://www.lucidchart.com/pages/uml-activity-diagram (accessed on 14 August 2023).
31. Apache Kafka. Available online: https://kafka.apache.org/ (accessed on 14 August 2023).
32. Wan, E.A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), Lake Louise, AB, Canada, 4 October 2000; pp. 153–158. [CrossRef]
33. Brossard, M.; Barrau, A.; Bonnabel, S. A Code for Unscented Kalman Filtering on Manifolds (UKF-M). In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 5701–5708. [CrossRef]
34. lz4 Compression Algorithm. Available online: https://lz4.org/ (accessed on 14 August 2023).
35. Survey of 5G Upload Speed. Available online: https://www.opensignal.com/reports/2023/06/southkorea/mobile-network-experience (accessed on 14 August 2023).
36. Docker. Available online: https://www.docker.com/ (accessed on 30 August 2023).
37. Hirschmuller, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 2, pp. 807–814. [CrossRef]
38. Lastilla, L.; Ravanelli, R.; Fratarcangeli, F.; Di Rita, M.; Nascetti, A.; Crespi, M. Foss4g Date For Dsm Generation: Sensitivity Analysis Of The Semi-Global Block Matching Parameters. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 67–72. [CrossRef]
39. 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]
40. Nezhadarya, E.; Taghavi, E.; Razani, R.; Liu, B.; Luo, J. Adaptive Hierarchical Down-Sampling for Point Cloud Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 12956–12964. [CrossRef]
41. Balta, H.; Velagic, J.; Bosschaerts, W.; De Cubber, G.; Siciliano, B. Fast statistical outlier removal based method for large 3D point clouds of outdoor environments. *IFAC-PapersOnLine* **2018**, *51*, 348–353. [CrossRef]
42. Standard of HD Maps in Korea. Available online: https://www.ngii.go.kr/kor/contents/view.do?sq=1251&board_code=contents_data (accessed on 14 August 2023).
43. Jekeli, C. *Inertial Navigation Systems with Geodetic Applications*; Walter de Gruyter GmbH & Co. KG: Berlin, Germany, 2023.
44. UTM-K Coordinate System. Available online: https://epsg.io/5179 (accessed on 14 August 2023).
45. Gamby, A.N.; Katajainen, J. Convex-Hull Algorithms: Implementation, Testing, and Experimentation. *Algorithms* **2018**, *11*, 195. [CrossRef]
46. Li, L.; Yang, F.; Zhu, H.; Li, D.; Li, Y.; Tang, L. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote Sens.* **2017**, *9*, 433. [CrossRef]
47. Zand, M.; Etemad, A.; Greenspan, M. Oriented Bounding Boxes for Small and Freely Rotated Objects. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 4701715. [CrossRef]
48. Plonsky, L.; Ghanbar, H. Multiple regression in L2 research: A methodological synthesis and guide to interpreting R2 values. *Mod. Lang. J.* **2018**, *102*, 713–731. [CrossRef]
49. Lanelet2, Map Handling Framework for Automated Driving. Available online: https://github.com/fzi-forschungszentrum-informatik/Lanelet2 (accessed on 27 August 2023).
50. Arcos-García, Á.; Álvarez-García, J.A.; Soria-Morillo, L.M. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing* **2018**, *316*, 332–344. [CrossRef]
51. Wang, J.; Ma, Y.; Huang, S.; Hui, T.; Wang, F.; Qian, C.; Zhang, T. A keypoint-based global association network for lane detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 1392–1401. [CrossRef]
52. Zhou, Q.-Y.; Park, J.; Koltun, V. Open3D: A modern library for 3D data processing. *arXiv* **2018**, arXiv:1801.09847. [CrossRef]
53. German Traffic Sign Detection Benchmark. Available online: https://benchmark.ini.rub.de/?section=gtsdb (accessed on 30 August 2023).
54. Microsoft COCO Dataset. Available online: https://cocodataset.org/#home (accessed on 30 August 2023).
55. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
56. Tusimple Benchmark Dataset. Available online: https://github.com/TuSimple/tusimple-benchmark (accessed on 30 August 2023).
57. Kalampogia, A.; Koutsakis, P.H. 264 and H.265 Video Bandwidth Prediction. *IEEE Trans. Multimed.* **2018**, *20*, 171–182. [CrossRef]

58. Opencv. Available online: https://opencv.org/ (accessed on 14 August 2023).
59. Nvidia Stereo Disparity Estimator. Available online: https://docs.nvidia.com/vpi/algo_stereo_disparity.html (accessed on 14 August 2023).
60. Xu, Z.; Yang, S.X.; Gadsden, S.A. Enhanced Bioinspired Backstepping Control for a Mobile Robot with Unscented Kalman Filter. *IEEE Access* **2020**, *8*, 125899–125908. [CrossRef]