3D maps distribution of self-driving vehicles using roadside edges

Masaya Mizutani, Manabu Tsukada, Yuki Iida, Hiroshi Esaki

3D maps distribution of self-driving vehicles using roadside edges

Masaya Mizutani
The University of Tokyo
Tokyo, Japan
mizu@hongo.wide.ad.jp

Manabu Tsukada
The University of Tokyo
Tokyo, japan
tsukada@hongo.wide.ad.jp

Yuki Iida
Tier IV, Inc.
Tokyo, Japan
yuki.iida@tier4.jp

Hiroshi Esaki
The University of Tokyo
Tokyo, Japan
hiroshi@wide.ad.jp

***Abstract*—Three-dimensional (3D) maps have become a shared digital infrastructure for autonomous vehicles, especially in urban areas. Point Cloud Data (PCD) maps are used for scan matching to enable self-localization. Autonomous vehicles need to maintain PCD maps along with the destination that is often decided on demand and to keep the PCD map updated. In this paper, we propose a system that delivers PCD maps cached at roadside edges in real time. We implement the system in Autoware, an open-source software for autonomous driving. Subsequently, we evaluate whether the autonomous vehicle can simultaneously download the PCD map from its edge and enable self-localization. Our results show that autonomous vehicles can perform self-localization while downloading the PCD map from the edge server. Additionally, we measure the download time with variable bandwidth and examine the bandwidth in which the self-localization normally operates. In our results, the download time of the PCD map at 60 Mbps was 1.16 s at maximum, and it is indicated that 60 Mbps is the deadline for this system to work properly.**

***Index Terms*—Roadside edge, PCD map, autonomous vehicles**

I. Introduction

Currently, autonomous driving is one of the most important areas in next-generation research. Most autonomous vehicles now use cameras, LIDAR, and other in-vehicle systems to recognize their surroundings, make decisions, and execute driving commands. However, these systems have various problems. For example, the range that can be detected by a vehicle is limited, blind spots exist, and there is a limit to the computing resources and storage that can be mounted on an autonomous vehicle.

Autonomous driving is being developed by a variety of organizations, but some software is being developed as open source. This includes Autoware [1] by the Autoware Foundation, apollo [2] by Baidu, and Nvidia Drive by Nvidia [3].

In autonomous driving software, it is important to accurately know the position of the vehicle. Methods to obtain the vehicle's position include Global Navigation Satellite System (GNSS) [4] applications represented by GPS, Simultaneous Localization and Mapping (SLAM) [5], etc. It is known that GNSS gives an error on the order of several meters, which provides insufficient safety margin. Therefore, SLAM is often used in self-driving software.

SLAM works by using point cloud data and Point Cloud Data Maps (PCD Maps) [6]. The point cloud data is generated by LIDAR. LIDAR illuminates the target with laser light and measures the reflection with a sensor. Subsequently, differences in laser return times and wavelengths can be used to make digital 3D representations of the target. The PCD map is a collection of this point cloud data. Its format consists of a header and a body. The header contains metadata that describes the data in terms of type and size, etc. The body contains binary data for each point, and the information is read based on the metadata defined in the header.

The size of the PCD map data is large due to the typical number of points collected by a LIDAR; for example, 120,000 points every 100ms. In the future, assuming that automated vehicles will travel long distances, it will be difficult to store all PCD maps on the route in a vehicle. In addition, it is necessary to update the map frequently because the surrounding environment changes each moment due to traffic accidents and construction work. For these reasons, it is necessary to support autonomous driving by transmitting PCD maps through inter-vehicle or vehicle-edge communication. Here, the system in which a vehicle solves these challenges by communicating with other vehicles and edges is called Cooperative Intelligent Transport System (CITS) [7].

In this paper, we propose a system for solving this problem. Additionally, we show how to extend Autoware, an open-source software, and conduct an experiment to download the map of the vehicle's surroundings from the edge. We also measure and evaluate the delay in downloading the map.

The structure of this paper is as follows. In section II, related technologies and related research, including standards related to vehicle-to-vehicle communication, are introduced; in section III, requirements for this research are set. Section IV describes our proposed method of PCD map distribution. The method used in field experiments for evaluating the proposed method is described in section V. Section VI describes our conclusions and future issues.

II. Related works

C-ITS uses vehicle-to-everything (V2X) communication, including vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. In [8], [9], roadside infrastructure support the cooperative perception of Autoware-based autonomous vehicles via V2X communication. V2V and V2I are communications between objects moving at very high speeds,

and therefore, vehicles and infrastructure do not communicate via base stations or fixed networks as in traditional networks. Vehicles and infrastructure communicate on the vehicular ad hoc network (VANET), which is realized through P2P communication. The communication standard used in VANETs is Dedicated Short Range Communications (DSRC) [10]. Recently, some research has been performed on using D2D communication over mobile networks such as 5G. DSRC and 5G communications are faster than conventional standards and communication over the Internet. To take advantage of this strength, there has been substantial research on edge computing. Edge computing brings computation and data storage closer to the devices that gather it, rather than relying on a central location that can be thousands of miles away.

There are many types of research on the efficient distribution of large-capacity data, such as maps using edges. Kim et al. [11] proposed an algorithm to efficiently distribute data in the cloud when the capacity of the storage at the edge is limited. It is shown that the suboptimal solution can be found in polynomial time by the verification experiment of these algorithms. Gangadharan et al. [12] proposed optimal map delivery using clouds and edges. This paper assumes a scenario in which all edge and vehicle data are aggregated into a cloud and analyzed, and the cloud provides instructions based on the results. In this study, the experiment was carried out by the simulation using MATLAB and the usefulness of considering the motion of the vehicle when optimizing the communication band was shown. Zhang et al. [13] proposed a method to optimize the communication load, assuming a scenario where an access point is located at the edge, and a car acquires data via the edge. In this study, by using Named Data Network (NDN) instead of IP as a communication protocol, it was shown that autonomously caching data at the edge helps to reduce communication time and communication bandwidth.

As described above, there have been no studies to verify the downloading of maps used in actual autonomous driving after constructing a practical network environment.

III. Issues and Requirements

In the future, we can consider autonomous driving over long distances. There is an issue with storing all PCD maps on the routes to perform autonomous driving. For example, a multi-purpose vehicle with no destination would require a huge amount of data, and a vehicle changing its destination would require more data than it had initially. To solve the above problems, we identify the following requirements.

1) Map distribution: PCD maps should be able to download from the edge or the cloud. This is because the vehicle cannot store all PCD maps on the route. Hence, if the vehicle does not have the necessary PCD map, it should be able to download it from a remote storage server.

2) Map delivery delay: The delay in downloading the PCD maps must be small. If the vehicle does not have the map of the point where it's heading, the vehicle must obtain the map before the vehicle reaches that point. Also, if the vehicle is downloading the map via an access point at the edge, the map must be downloaded while the vehicle is connected to the access point. So, to solve the issue, it is necessary to reduce the delay in downloading the PCD maps.

3) Freshness of maps: PCD maps should have near real-time update frequency. For example, if an accident occurs in the vehicle's path and the surrounding environment changes significantly, the vehicle must update the map to perform autonomous driving. So, to solve the issue, it is necessary to update the PCD map with near real-time frequency.

IV. PCD Map Distribution System

A. System Design

In this study, we propose a system that satisfies the above requirements. Fig. 1 shows the configuration of this system. As shown in the figure, there are three types of nodes in the system: cloud, edge, and vehicle. The cloud consists of a PCD map registry (which is an HTTP server for managing PCD maps), storage for PCD maps, and a database for storing the file paths of the PCD maps. The edge consists of a sync server that checks for PCD file updates in the cloud and storage for PCD maps. The vehicle consists of the self-driving software Autoware and a client for sending HTTP requests to the PCD Map Registry.

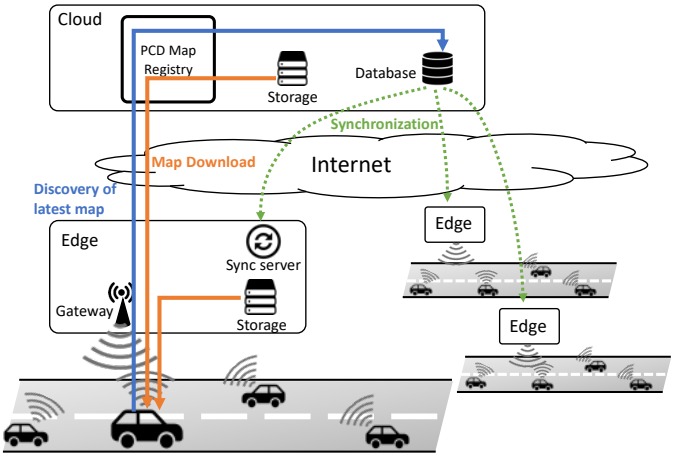


Fig. 1: System configuration and process of system

Fig. 1 shows the flow of the system. As shown in the figure, the edge is arranged between the vehicle and the cloud server, and the map is distributed. First, before downloading the map, the map is uploaded to cloud storage, and its file path is registered in the database via the PCD map registry. The download flow for the PCD map is as follows.

Map Synchronization

As this system is gradually spreading, it is necessary not to break the existing architecture when the edge network is newly installed. Therefore, caching PCD maps to the edge network is accomplished via a push mechanism. The edge network determines an area to be managed in advance. The edge synchronization server periodically checks the map for updates, stores the map in edge storage and registers its file path.

Map update check

The vehicle detects a change in your location, and HTTP Client requests the PCD map file paths from the PCD Map Registry via Gateway.

Discovery of latest map

The PCD Map Registry refers to the database and searches for the file path of the appropriate map. If the map exists in the edge storage, the PCD Map Registry sends the file path of the PCD map in the edge storage to the HTTP client; otherwise, the PCD Map Registry sends the file path of the map in the cloud to the HTTP client.

Map Download

The HTTP client requests the PCD map from the edge storage or the cloud storage via the gateway. Then, the edge storage or the cloud storage sends the PCD map to the client.

B. Implementation

In this study, we extended Autoware and created a system, so we describe Autoware before explaining the implementation of the proposed method.

1) Autoware: Autoware is based on the Robot Operating System (ROS). ROS is a distributed computing platform involving ***nodes*** and `topics`. The ***nodes*** represent the processing module of tasks, and the ***nodes*** communicate with one another via `topics`. Furthermore, ROS provides a powerful tool named *ROSBAG* for recording and replaying the messages in `topics`. Moreover, ROS includes a 3D visualization tool named *RViz*. The processes for self-driving in Autoware consist of sensing, localization, perception, decision, planning, and actuation. The input information is the initial position obtained from GNSS, a point group data obtained from the LIDAR and the PCD map. The point group data obtained from the LIDAR is compared with the PCD map, and the position with the highest coincidence probability is estimated as the self-position.

In addition, Autoware uses the Universal Transverse Mercator (UTM) coordinate system [14], which can specify the coordinates of the entire world by the geographic identifier. Because it is derived from a projected coordinate system, the area can be accurately identified. Meanwhile, the PCD map files are maintained in such a way that they can be uniquely identified by a coordinate in the Military Grid Reference System (MGRS) coordinate system [15]. MGRS is an extension of the UTM coordinate system. Its level of detail can be changed to fit scales such as 100 m square or 1000 m square. Since the amount of data in PCD maps is very large, they are stored in 100 m squares (the file size of each square is about 100 MB at most, depending on compression). Therefore, the PCD maps are managed in the MGRS coordinate system, which is scalable.

2) Map Downloader: The detailed architecture of the vehicle-side system is shown in Fig. 2.

The diagram shows only the relevant parts of Autoware's nodes and topics, showing the relationship to the edge and the cloud. `/filtered_points` is the point cloud data, filtered to reduce the amount of data. In Autoware, ***points_map_loader*** loads the PCD map in the vehicle and publishes to `/points_map`. ***Ndt_matching*** subscribes to `/filtered_points`. Subsequently, it implements the normal distributions transform (NDT) scan-matching [16], assumes the current position of the vehicle, at 10Hz, and subscribes to `/ndt_pose` which is the current position and orientation of the vehicle. In this study, we implemented a new node, ***remote_map_loader***, and replaced ***points_map_loader***. This node downloads the PCD maps of surrounding areas according to the position of the vehicle. To enable this, it subscribes to `/current_pose`, which is the current position and orientation of the vehicle.

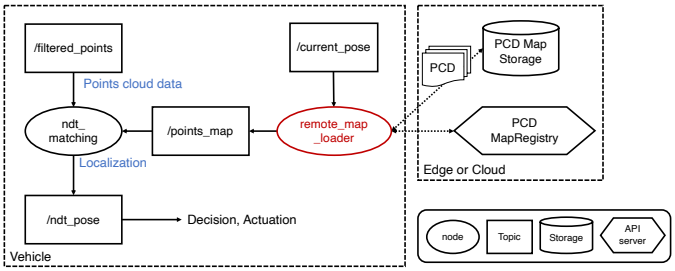


Fig. 2: Detailed system architecture on the vehicle side

The detailed flow of processing is shown in Fig. 3. First, ***remote_map_loader*** subscribes to `/current_pose`, indicating the current coordinates of the vehicle. Here, `/current_pose` represents a self-position in the UTM coordinate system. From there, ***remote_map_loader*** converts the self-position into the MGRS coordinate system and searches for the PCD maps including, in our study, $5 \times 5 = 25$ maps around the grid in the position of the vehicle in the local storage. If there is no map, it requests the file paths of PCD maps from the PCD map registry. When it receives the file paths, it requests the PCD maps from the storage. Finally, when it receives all maps, it publishes to `/points_map`. This process allows Autoware to load a PCD map of the surrounding area according to the location of the vehicle.

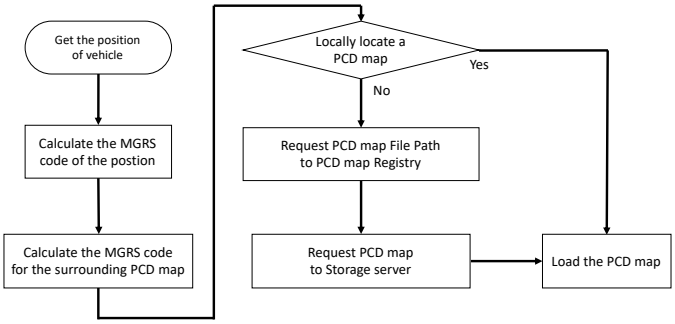


Fig. 3: Process on ***remote_map_loader***

V. EVALUATION

A. Experimental environment

The purpose of this experiment was to evaluate whether the vehicle can download maps in real time using wireless communication while driving. The experiment was conducted by reproducing previously recorded travel data from the Hongo campus. To perform this experiment, an experimental environment was used. The environment used AWS EC2 instances (Northern Virginia Region) for the PCD map registry and Amazon S3 for cloud storage. The edge contained a PC running MinIO [17] for the storage server and running the sync server. To represent the vehicle, a PC running Autoware to reproduce the running data was wirelessly connected to a router.

The specifications of the communication parts used in this experiment are shown in Table I.

TABLE I: Communication method within the edge

Place	Communication system	Bandwidth
PC1 · router	Wi-Fi IEEE 802.11ac	638Mbps
PC2 · router	Ethernet 1000Base-T	942Mbps
PC1 · EC2	-	592Kbps

We scanned the 4km road of the Hongo campus at the University of Tokyo and created the PCD map published on our website[1]. The original data size is about 1.35 GB. We perform downsampling by the voxel grid filter with a leaf size of 20 cm because the NDT scan matching works efficiently with the downsampled PCD map. The voxel grid filter reduced the data size to 90.6 MB. We conduct the experiments with the downsampled PCD map.

B. Validation

First, the Autoware software running on the vehicle reproduces the data (ROSBAG) from a vehicle that has traveled along the route. In ROSBAG, Autoware processes point cloud data obtained by LIDAR while driving and stores selected topics among published topics. Therefore, by reproducing the ROSBAG, an experiment is performed by running the vehicle in a simulated manner. In the ROSBAG, the average vehicle speed was 13.1 km/h , and the maximum vehicle speed was 27.3 km/h . Then, the ROSBAG was played back, and the newly implemented ***remote_map_loader*** node was run. In this experiment, the vehicle downloaded $5 \times 5 = 25$ maps surrounding the position of the vehicle.

Fig. 4 shows the actual running result in RViz. This figure shows the 3D space visualization of each topic published by Autoware. The white dots represent the point cloud data that form the PCD map, the red dots represent the point cloud data retrieved from LIDAR, and the yellow lines show the vector map that represents the geological information. The vehicle is located near the center of the point cloud data shown in red. The figure shows the map in the vehicle when the vehicle is stopped at the initial position, the one when the vehicle has been running for some time, and the one when the PCD map where the vehicle is located is switched, in the order of (a) to (c).

The results show that it is possible to download 25 PCD maps in real-time from edge storage using Wi-Fi while driving.

C. Download time measurement

First, we analyze the deadline for the download time of a map tile, then we measured the time required to download the map using the configuration in the previous section.

The deadline T is given by

$$T = \frac{l}{v \times n} \tag{1}$$

,where l is the length of a tile, v is the vehicle's speed, and n is the number of tiles required to download in a new map area. In our scenario, $l = 0.1km$, and $n = 5$ at maximum, assuming that the vehicle download all the neighboring map tiles in a new area. Assuming that the vehicle is traveling at 20 km/h, the deadline $T = 3.6$ seconds ($\frac{0.1}{20 \times 5} \times 60 \times 60$). Meanwhile, if the vehicle is traveling at 60 km/h, $T = 1.2$ seconds.

We conducted three experimental evaluations. In the first trial, the vehicle downloaded all PCD maps at the Hongo campus and measured the time required to download them from a cloud server. The results are shown in Fig. 5. Fig. 6 also shows the download time from the edge server.

Downloading from the cloud storage took an average of 37.0 s to download from the cloud, with a maximum of 121 s. This result indicates that it does not fill the deadline.

In comparison, it took an average of 109 ms and a maximum of 282 ms to download from the edge storage. The results show that a PCD map on the Hongo campus can be downloaded in no more than 290 ms. This indicates that it is possible to download all maps at 60 km/h.

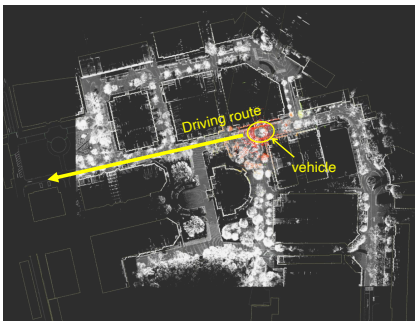
In the second experiment, the vehicle downloaded the PCD maps with variable bandwidth for transmitting data by using TC command [18]. This result is shown in Fig. 7. For example, as shown in the figure, at 60 Mbps, the maximum time to download a map is 1.16 s. Meanwhile, at 55 Mbps, the maximum time to download a map is 1.27 s. From this result, the bandwidth to fill the deadline of the bandwidth is 60 Mbps.

In the third experiment, we verify whether self-localization can normally be performed for a vehicle with variable bandwidth. It was found that self-localization was normally possible with up to 7 Mbps bandwidth between the vehicle and the router but became impossible at 6 Mbps.

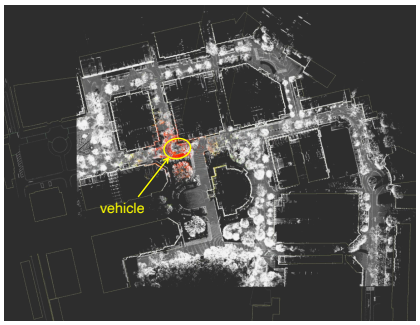
VI. CONCLUSIONS

Autonomous driving suffers from the issue that a vehicle cannot store all maps on its route. Therefore, in this work, we propose a prototype system that allows vehicles to download maps from the edge or cloud storage. We also implemented and evaluated this system by reproducing the ROSBAG recording. It was found that by using edge storage,

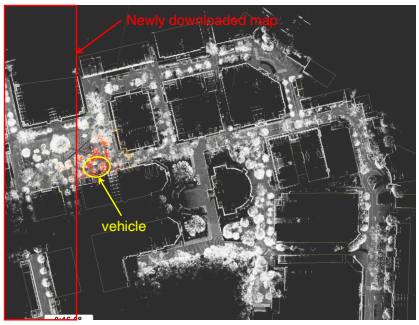
[1]https://tlab.hongo.wide.ad.jp/maps/hongo_pointcloud



(a) Stopped at the initial position



(b) State awhile after starting



(c) State where the PCD map has changed

Fig. 4: View on RViz

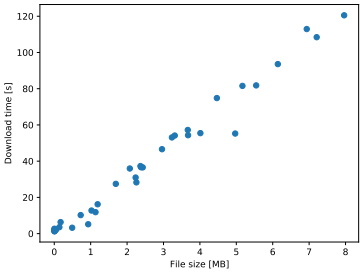


Fig. 5: Time taken to download maps from cloud storage server to vehicle

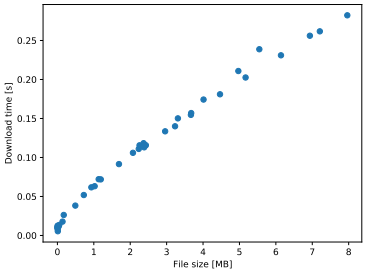


Fig. 6: Time taken to download maps from edge storage server to vehicle

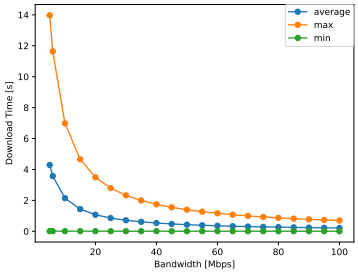


Fig. 7: Time taken to download the PCD map with variable bandwidth between the vehicle and the router

it is possible to perform self-localization while downloading the PCD maps. In addition, we measured the download time with variable bandwidth and examined the bandwidth in which self-localization normally operates. The download time of the PCD map at 60 Mbps was found to be 1.16 s at most, and it was indicated that 60 Mbps is the deadline for this system to work properly. As a future prospect, we would like to examine a system to verify this operation when the number of vehicles increases and consider a system to upload the PCD maps.

References

[1] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, ieeexplore.ieee.org, Apr. 2018, pp. 287–296.

[2] *Apollo*, http://apollo.auto/, Accessed: 2020-4-25.

[3] *Autonomous car development platform from NVIDIA DRIVE AGX*, https://www.nvidia.com/en-us/self-driving-cars/drive-platform/, Accessed: 2020-4-25.

[4] D. garcia-yarnoz, *International committee on global navigation satellite systems (ICG)*, Accessed: 2020-4-24. [Online]. Available: http://www.unoosa.org/oosa/en/ourwork/icg/icg.html.

[5] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot.," in *IROS*, vol. 3, 1991, pp. 1442–1447.

[6] *PCL - point cloud library (PCL)*, http://pointclouds.org/, Accessed: 2020-4-24.

[7] ETSI, "Intelligent Transport Systems (ITS); Framework for Public Mobile Networks in Cooperative ITS (C-ITS)," 2012, ETSI TR 102 962 V1.1.1 (2012-02).

[8] M. Tsukada, T. Oi, A. Ito, M. Hirata, and H. Esaki, "AutoC2X: Open-source software to realize V2X cooperative perception among autonomous vehicles," in *The 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, Victoria, B.C., Canada, Nov. 18, 2020.

[9] M. Tsukada, T. Oi, M. Kitazawa, and H. Esaki, "Networked roadside perception units for autonomous driving," *MDPI Sensors*, vol. 20, no. 18, Sep. 17, 2020. DOI: 10.3390/s20185320.

[10] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[11] R. Kim, H. Lim, and B. Krishnamachari, "Prefetching-Based data dissemination in vehicular cloud systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 292–306, Jan. 2016.

[12] D. Gangadharan, O. Sokolsky, I. Lee, B. Kim, C. Lin, and S. Shiraishi, "Bandwidth optimal Data/Service delivery for connected vehicles via edges," in *2018 IEEE*

11th International Conference on Cloud Computing (CLOUD), Jul. 2018, pp. 106–113.

[13] Z. Zhang, T. Li, J. Dellaverson, and L. Zhang, "Rapid-VFetch: Rapid downloading of named data via distributed V2V communication,"

[14] J. P. Snyder, *Map projections–A working manual*. US Government Printing Office, 1987, vol. 1395.

[15] P. H. Stott, "The UTM grid reference system," *IA. The Journal of the Society for Industrial Archeology*, vol. 3, no. 1, pp. 1–14, 1977.

[16] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3, Oct. 2003, 2743–2748 vol.3.

[17] MinIO, Inc, *MinIO — high performance, kubernetes native object storage*, https://min.io, Accessed: 2020-5-31.

[18] B. Hubert, *Tc(8): Show/change traffic control settings - linux man page*, https://linux.die.net/man/8/tc, (Accessed on 08/10/2020).