International Journal of Intelligent Transportation Systems Research (2020) 18:77–89
https://doi.org/10.1007/s13177-018-0173-7

DynamicMap 2.0: A Traffic Data Management Platform Leveraging Clouds, Edges and Embedded Systems

Yousuke Watanabe[1] · Kenya Sato[2] · Hiroaki Takada[1,3]

Received: 4 June 2018 / Revised: 17 October 2018 / Accepted: 25 October 2018 / Published online: 12 November 2018

Abstract
Various embedded sensors such as LiDAR, cameras and GPS are mounted on vehicles today. And, vehicles with communication equipment can exchange sensor data between not only surrounding vehicles but also road-side-units. To achieve traffic safety and comfort driving, utilization of these traffic data are important for autonomous driving and advanced driving assistance systems. Our research group has been developing a city-scale traffic data management platform named DynamicMap 2.0. This paper proposes a software architecture of our platform. Our platform uniformly treats road maps, static information, dynamic information and predicted information. Road maps consist of three-level granularities: links, lanes and physical shapes. Especially, a data representation of lane-level road maps is newly proposed. Static information includes fixed properties about objects on the map. Dynamic information includes streaming data from sensors. And, predicted information is computed by prediction algorithms and simulations. Our platform provides query language for searching these traffic data. A developer of traffic applications writes a query (continuous query or one-shot query) which specifies integration and filtering of data. As a use case of our platform, we developed a prototype of a cooperative merging assistance system at intersections, and verify that our platform operates correctly.

Keywords Dynamic map · Traffic data management · High-precision road map · Query language

1 Introduction

In recent years, advanced driver assistance systems and autonomous driving systems have been rapidly studied and developed. Embedded sensors enable vehicles to recognize surrounding environment. A vehicle notifies dangerous situations, or avoids conflicting objects automatically. Cooperative intelligent transport systems, which communicate with vehicles and road-side units, are also developed to optimize traffic flows and to reduce environmental burdens.

Individual sensor has pros and cons. A recognition area covered by each sensor is quite limited (range of LiDAR is about 120 m [1]). Some optical sensors (cameras) cannot detect back objects hidden by another front object. Therefore, sharing sensor data obtained from multiple vehicles and road-side-units become important to improve quality of traffic services.

Demands from advanced in-vehicle-systems encourage advancement of digital road maps. Conventionally, road maps are used for navigation, which provides a route from the origin to the destination. In addition to that, new roles are expected to road maps. One is providing detailed physical shapes about surrounding geographical objects. Because, autonomous driving systems need them to estimate the current position of ego vehicle. Another one is giving meaning interpretation to sensor data. Since raw sensor data are just a set of numeric values, meaning based on traffic rules is to be attached (which lanes a detected object moves on, what kind of relationships exists between my lane and object's lane, and so on). Thus, many organizations discuss and experimentally produce high-precision road maps which satisfy new demands from advanced in-vehicle-systems.

Dynamic Map [2, 3] is proposed as a notion of the next-generation road maps. It is a logical data set which enables to overlay sensor data (dynamic information, transient-dynamic

✉ Yousuke Watanabe
watanabe@coi.nagoya-u.ac.jp

[1] Institute of Innovation for Future Society, Nagoya University, Furocho, Chikusa, Nagoya, Aichi, Japan

[2] Mobility Research Center, Doshisha University, Kyoto, Japan

[3] Graduate School of Informatics, Nagoya University, Nagoya, Japan

information and transient-static information) onto a high-precision road map by using a location reference method. Originally, it was proposed as Local Dynamic Map (LDM) [4–6] in Europe. Although LDM mainly considers sensor data from surrounding environment, Dynamic Map is extended to treat city-scale sensor data. Now, dynamic map is regarded as important information infrastructure for supporting advanced in-vehicle-systems. In Japan, Dynamic Map is one of importance research areas of SIP-adus project [2]. In Europe, some element technologies for Dynamic Map are discussed. NDS [7] proposes a data format of high-precision road maps [8], and ERTICO – ITS Europe [9] considers a specification for gathering data from vehicles to a cloud [10].

Information exchange among vehicles and road-side-units in local area has been already studied and known as Cooperative ITS. But, local information exchange is not enough to support city-scale traffic data management. We need an information platform to gather/manage/utilize city-scale sensor data and high-precision road maps. For this purpose, we have to treat large volume of real-time sensor data. A centralized cloud server is not suitable to handle real-time sensor data.

because communication delays become unacceptable. It is difficult to rapidly reply a response message by the temporal deadline. A centralized architecture should be reconsidered.

To avoid this problem, we employ edge computing. Here, "edge computing" means utilizing computational resources at base stations for mobile phones and road-side units. We propose a geographically-decentralized architecture, which consists of embedded devices, edge servers and cloud servers. Parallel and distributed data processing and exchange are performed among the distributed environment. To implement such software platform, we built up a research organization "Dynamic Map 2.0 Consortium" [3] in 2016. Several companies and universities belong to the consortium.

In this paper, we introduce DynamicMap2.0 platform (shortly, DM2.0PF), which is a distributed information platform developed by our consortium. We summarize our challenges and contributions:

- **A distributed architecture for processing large volume of real-time data**: Data processing nodes in DM2.0PF are clouds/edges/embedded systems. They collaboratively work and simultaneously process traffic data. Most of traditional traffic services are based on standalone or client/server architectures. On the other hand, we focus on utilizing edge computing. Thanks to edge nodes, we can satisfy both of real-time data management and city-scale data management.
- **A common data model for heterogeneous traffic data**: DM2.0PF treats four information-types: road maps and static/dynamic/predicted information about objects on roads. Road maps are essential information in Dynamic Map. Static information includes fixed properties about objects on the map. Dynamic information includes streaming data from sensors. Predicted information is computed by prediction algorithms and simulations. Empty areas observed by sensors are treated as virtual objects, and they are used for short-term prediction. Since four information types are heterogeneous, integration of them is not a trivial problem. For example, road map and static information are stored in HDDs, but dynamic and predicted information are produced continuously (streaming). We employ the relational model as a common data model in DM2.0PF. All types of information are logically represented as relations (tables).
- **High precision road map format for integrating traffic data**: This paper proposes our road map format. Since traffic applications have different demands to road maps, our road maps consists of three-level granularities: links, lanes and physical shapes. Especially, a data representation of lane-level road maps is newly proposed. Our lane-level road map is an extension of GDF (Geographic Data Files) [11] which is the international standard of link-level road maps. To be specialized for integration of other traffic data, it is represented as the common data model in DM2.0PF.
- **Common access method (query language) for heterogeneous traffic data**: DM2.0PF provides a common query language to handle four types of information. Since the relational model is used in DM2.0PF, our query language is an extension of SQL. SQL is the standard query language in RDBMSs (Relational DataBase Management Systems). Our query language covers not only one-shot query for stored data, but also continuous query for streaming data. A developer of traffic applications can write a query, which specifies integration and filtering of traffic data, without considering heterogeneity of information types.
- **A development of an example application on Dynamic Map**: We implemented a cooperative merging assistance service as a use case of DM2.0PF. This service uses DM2.0PF to monitor the specific merging point in real time. An edge node continuously gathers data from vehicles, and it provides assist messages to approaching vehicles. We use traffic simulator and driving simulator to generate input traffic data.

The remaining parts are composed as follows: Section 2 introduces the objective of this paper and demands to dynamic maps. Section 3 mentions related work. Section 4 proposes our DM2.0PF. Section 5 explains information treated in DM2.0PF. Section 6 shows our query language for Dynamic Maps. Section 7 presents a traffic application service using our platform. Finally, Section 8 includes conclusion and future research issues.

2 Objective and Requirement Analysis

The objective of our research is to develop a distributed information infrastructure for city-scale traffic data management. It supports utilization of high-precision road maps and real-time sensor data.

Use cases of our platform are achievement of traffic safety and comfort driving provided by in-vehicle-systems (autonomous driving systems and advanced driver assistance systems). Optimization of city-scale traffic flow is also included. Concretely, the following traffic applications are considered.

- **Personalized navigation**: When a navigation system provides a route from the origin to the destination, it considers driver's unfavorite situations. For example, driving a road with many blind areas, driving against the sun, and driving unfamiliar roads are often hated by drivers. A personalized navigation system utilizes high-precision road maps, and recommends a customized route which enables the driver to avoid unfavorite situations. This service is expected to reduce driver's stresses.
- **Smart traffic flow**: DM2.0PF is used to support cooperative merging assistance. Fig. 1 illustrates a merging point of highway ramp. An elder driver on a red vehicle is approaching to a merging point. In-vehicle-system predicts arrival timing of the red vehicle, and it sends a request message to ask another driver (on a light-blue vehicle) to yield up the lane. When the request is accepted, the elder driver can smoothly join the highway.
- **Extending recognition ranges of in-vehicle-systems**: Advanced in-vehicle-systems need wider views for safety and smart driving. Short sensor-range and blind angles of sensors are to be complemented by information exchange in DM2.0PF. Shared sensor data are used to judge whether a space is occupied or not occupied. A space not occupied by anybody is suitable for safety driving.

2.1 Requirement Analysis

To achieve traffic applications in the previous section, a Dynamic Map platform should have the following functions.

- **Data processing capability for large volume of real-time data**: Sensor data are gathered from all sensors in a city. We have to consider limited communication bandwidth of cloud servers. For autonomous driving systems, response messages should be returned by the deadline. Centralized systems are insufficient for large volume of real-time data, thus we need distributed processing with employing edge computing.
- **High-precision road maps**: In Fig. 1, a lane-change request is sent to a light-blue vehicle. To achieve this, lanes are distinguished in the road map. Each lane should be represented as a geographical object. Thus, Dynamic Map must support lane-level road maps. Similarly, details are also needed at intersection areas. Trajectory lines within an intersection are to be included. They are useful to predict movement of other conflicting vehicles.
- **Common access method for traffic data**: Road maps and static information are stored in long term and searched on demand. On the other hand, dynamic/predicted information are streaming data continuously provided from sensors. We need a common data model and a common access method to uniformly handle heterogeneous information.
- **Explicitly sharing information about empty areas and unknown areas**: A space on a road is to be classified into occupied or not occupied. But, just sharing information about detected objects is not enough. We have to consider outside of sensor-range and blind angles. In Dynamic Map, empty and unknown areas should be shared explicitly.

2.2 Assumption for Penetration Ratio of Connected Vehicles

Though small sensors and wireless communication equipment are becoming popular, it is difficult to assume all vehicles and pedestrians completely equip them in near future. Thus, we must still consider vehicles and pedestrians without any sensors and communication equipment. Sensors on road-side-units are expected to detect them and to report their positions.

3 Related Work

Here, we introduce some research projects related to traffic data management.

SIP-adus [2] is a special project in Japan, because it is led by Cabinet Office, Government of Japan. Dynamic Map is also studied in the SIP-adus. But, our research area is completely different from one of the SIP-adus project (Fig. 2). In the SIP-adus project, it focuses on data representation of high-precision maps. Especially, they are interested in how to create the map data from measured point cloud data, and how to deliver the map data from a map center to data centers of car manufacturers. However, integration of road maps and dynamic information is not included in their scope. It is regarded as competition research area. Thus, the SIP-adus project never provide a common data model and a common access method for traffic data management.

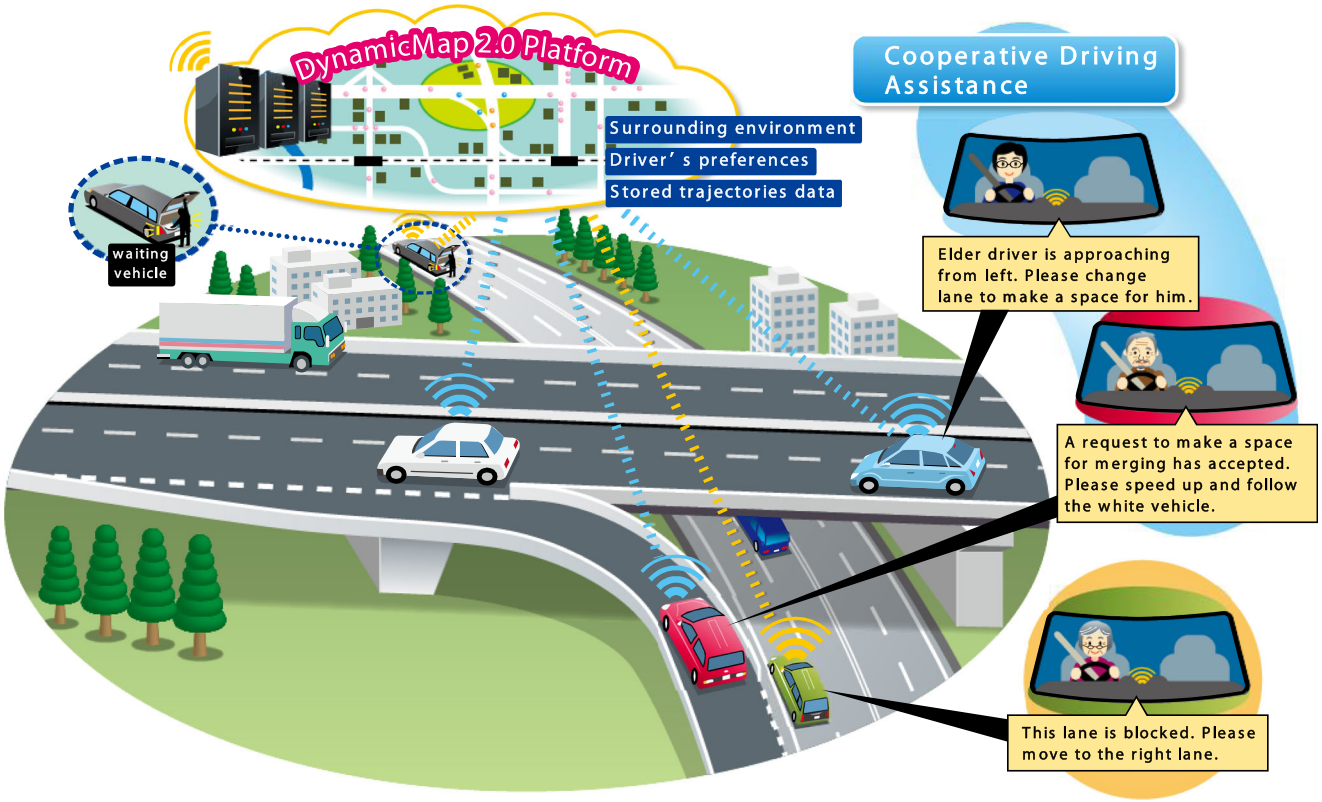


Fig. 1 Use case of DM2.0PF

AEDSMS [12] focuses a traffic data management in embedded systems. It is a distributed system within a vehicle. In their environment, multiple ECUs (Electronic Control Units) are connected via CAN (Controller-Area-Network). It regards data from on-board-sensors as streams, and provides a query processing framework for streams. Its main purpose is to improve development cost of in-vehicle systems. Information sharing like Dynamic Map is not considered.

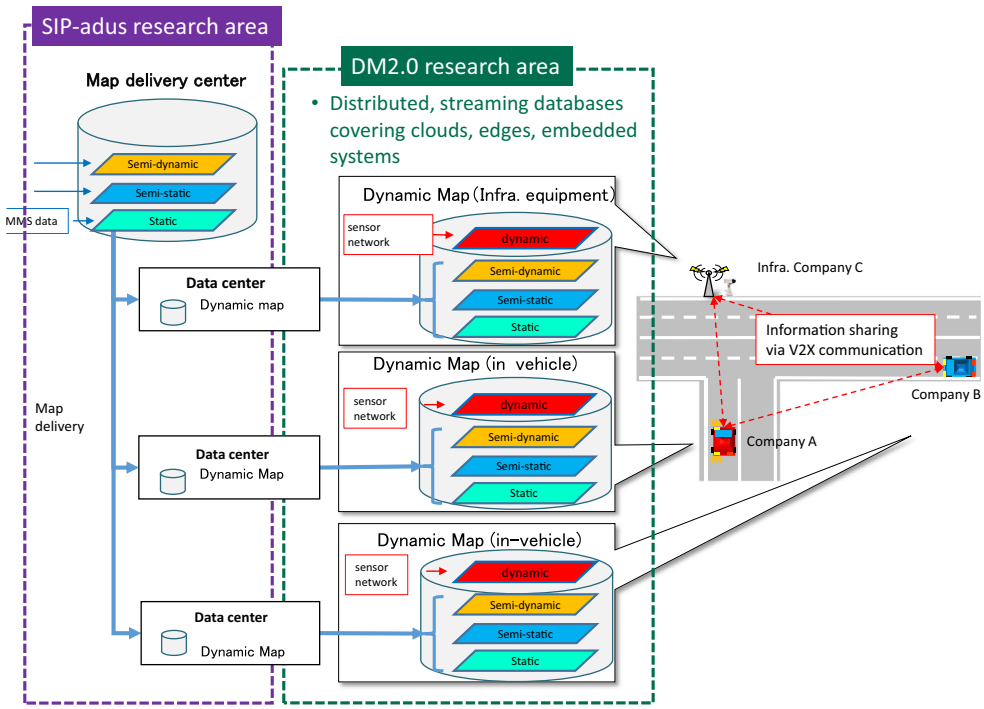


Fig. 2 Research area of DM2.0PF

SAFESPOT [4] is one of implementations of Local Dynamic Map. "Local" means that their main target is data management between road-side unit and vehicles. Thus, city-scale data management is out of their scope. They use a conventional RDBMS to manage traffic data. The relational model is also used in SAFESPOT. RDBMS can treats queries for stored data, but cannot treat queries for streaming data directly.

Processing traffic data in cloud is studies in many projects. Most of them are based on the client/server architecture. CarStream [13] is a system to analyze big data extracted from vehicles. It can process city-scale large data. CarStream is a distributed system in data center, but it is not "geographically" distributed system. It does not consider a problem of network delay between cloud and vehicle. Thus, it cannot be applicable to real-time traffic services such as information provision to autonomous driving systems.

DM2.0PF mainly focuses on information integration in a geographically distributed environment. Especially, edge computing is a key technology. And, a query processing scheme is considered. The relational model is employed as a common data model, and SQL-like query language for both stored data and streaming data is available in our platform. By using queries, we can integrate road maps, static/dynamic/predicted information uniformly.

4 Dynamic Map 2.0 Platform

In this section, we introduce an architecture of our distributed information platform. Figure 3 presents an architecture of DM2.0PF.

4.1 Distributed Environment Composed of Cloud/Edge/Embedded Nodes

DM2.0PF works in a distributed environment. A node belonging to DM2.0PF works on each element of the distributed environment. Cloud servers are located at data centers, and they are regarded as final destination of traffic data. Embedded devices are in-vehicle-systems and smart phones. They continuously produce sensor data. Edge servers are base stations of mobile phones and road-side-units. We assume wired-network between cloud server and edge server and wireless-network between edge server and embedded device. We may use them separately, or make collaboration of them. Cloud servers are good for treating large volume of data, but not suitable to satisfy real-time constraints (deadlines). Embedded devices mainly focus on real-time operations, but their own sensor covers limited areas. Edge servers are used for real-time operations on middle range.

4.2 Query Processing for Traffic Data

Developing a program code for data processing from a scratch needs an expensive cost. DM2.0PF provides a query processing scheme for large volume of data. Instead of writing a program code, a query is submitted to our platform. To reduce development cost, our platform supports a common data model and a common access method. The relational model is employed as a common data model, and SQL-like query language is provided as a common access method. Two-styles of queries are available:

- **One-shot query**: One-shot query is a query-processing style for stored data. Once a query is submitted, the system immediately generates the query results from stored data. It is almost same as one supported in traditional RDBMS.
- **Continuous query**: Continuous query is another query processing style for streaming data. Since sensor data are repeatedly generated, a continuous query is evaluated repeatedly. It is almost same as one supported in DSMS (Data Stream Management System) [12].

A traffic application submits a query with filter conditions and join conditions to DM2.0PF. The query is analyzed, and converted into an internal data processing flow among cloud/edge/embedded nodes. Our query language is explained in Section 6.

For the proof of concept, we implemented a prototype of DM2.0PF. It is written in C++ programming language. The core part of DM2.0PF is composed of 81 C++ modules (and classes). It can work on Linux-based computers. This prototype was used to develop a traffic application described in Section 7.

5 Information Managed by DM2.0PF

This section mentions information-types in our platform. They are road maps, static information, dynamic information, and predicted information. Each of them are explained in the remaining subsections.

5.1 High-Precision Road Maps

Since demands for road maps are different for each traffic application, we have to consider several granularities of road maps.

For example, route-search finds paths from the origin to the destination, and estimates travel time of the paths. Road network is necessary for this application, but detailed shapes of roads and lanes are not needed. On the other hand, cooperative driving assistance needs geographical relationships between lanes and intersections, because it must advise timing for lane-change and merging. Autonomous driving systems often need detailed physical shapes of building and road surfaces to apply self-position estimation.

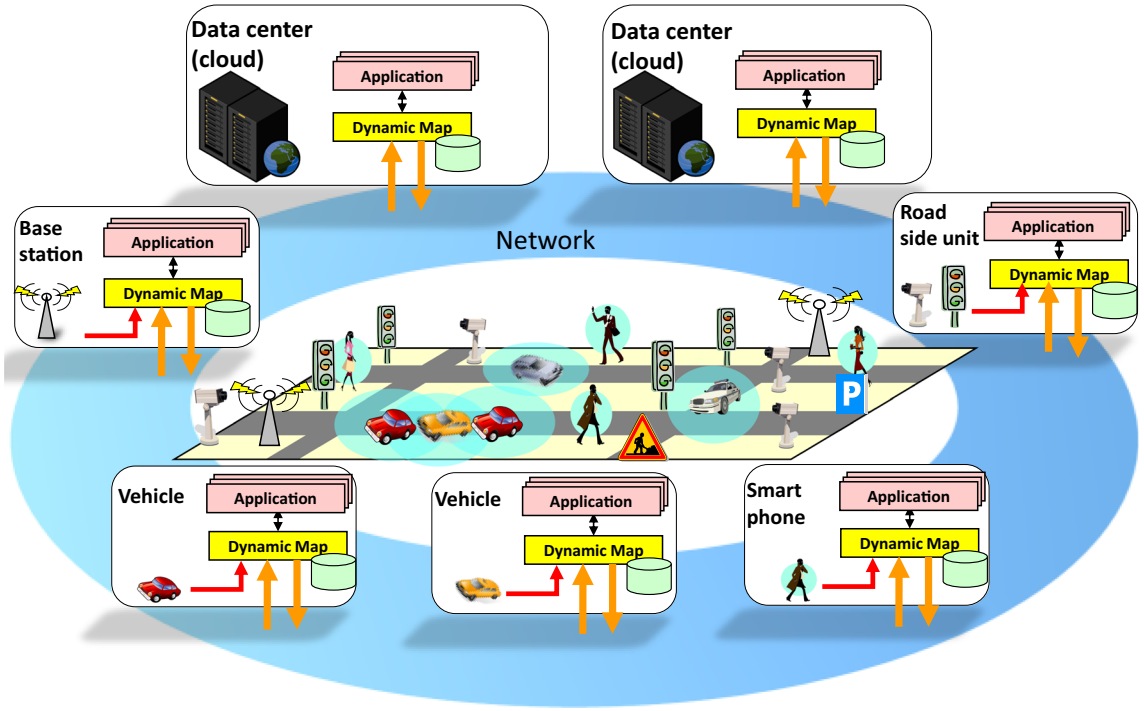


Fig. 3 Distributed environment of DM2.0PF

As described above, we need multiple levels of details in road maps. So we consider three granularity-levels (Fig. 4):

- **Link-level**: It is represented as a graph. An intersection is regarded as a vertex of the graph. A road is as an edge of the graph. This level corresponds to an existing road map used in navigation.
- **Lane-level**: It is also a graph, but each lane in a road is separately represented as a graph element. Lanes are necessary for traffic applications for lane-change and

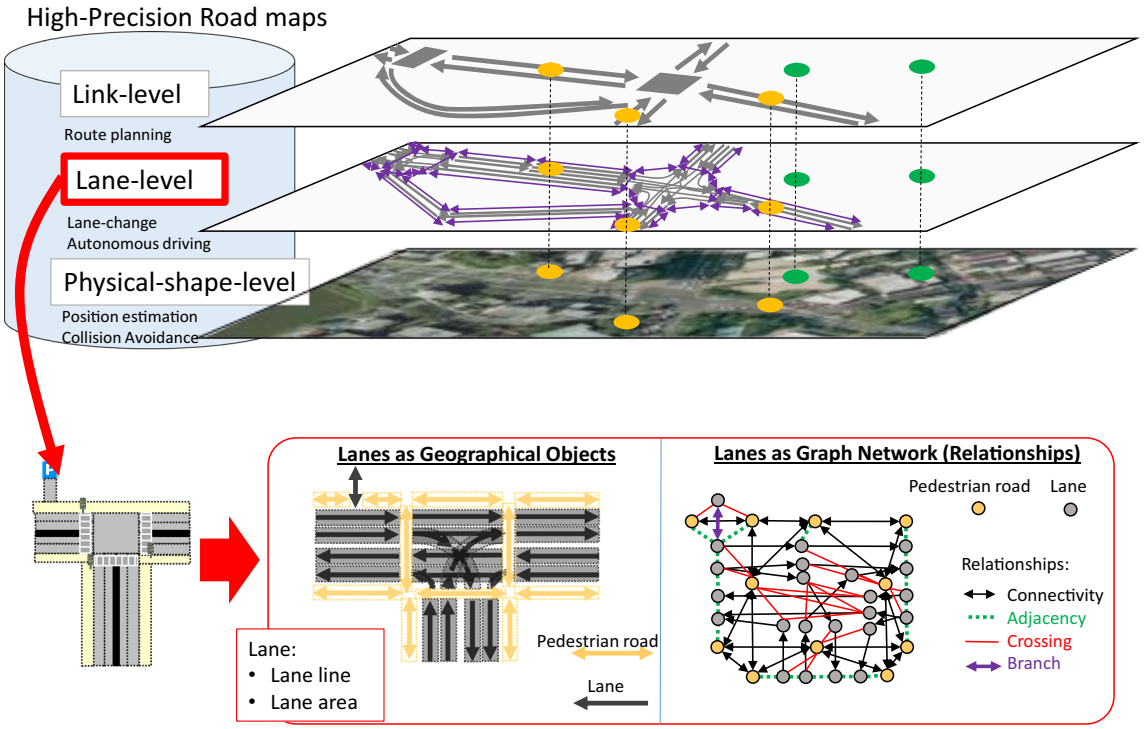


Fig. 4 High-Precision Road Maps

merging. Trajectories within intersections are also necessary as a rough guide to estimate paths of other vehicles. A data representation of lane-level road maps will be explained in Section 5.1.1.

- **Physical-shape-level**: It is a set of raw data obtained from cameras and LiDARs. Examples are point cloud, images, and videos. Traffic applications such as self-position estimation and precise collision avoidance need highly precise physical shape.

5.1.1 Data Representation of Lane-Level Road Maps

In our platform, a link-level road map is based on GDF 5.0 (Geographic Data Files 5.0) [11], which is the international standard of conventional road maps. Since physical shape-level is a raw data set, their data format highly depend on types of sensors. We cannot assume the standard data format.

Lane-level road maps are currently studies in the world. In the original GDF5.0, lanes are described as a property of road (the number of lanes on the road), thus lanes are not individual geographical objects. To fix this problem, we have extended GDF5.0 to represent lanes as geographical objects and graph data.

GDF: Geographic Data Files 5.0 At the first, we explain a foundation of GDF5.0. Major data types in GDF5.0 are Feature, Attribute, and Relationship.

- **Feature**: Geographical objects on a map in GDF5.0 are called Features. Road and Intersection are defined as Features. Features are classified into four categories: point, line, area and complex (combination of multiple Feature). Road and Intersection are complex feature, they consist of multiple points and lines. GDF5.0 allows users to add user-defined Features.
- **Attribute**: Each Feature has some Attributes to describe the Feature. Examples of Attributes are width and cant of Road. User-defined Attributes are also allowed.
- **Relationship**: It is used to express a special relation between multiple Features. Fork is a relationship to express a branch of a road. Relationship can have some attributes. User-defined Relationships are also allowed.

Extension of GDF (1) "Lane Feature" In this research, we have extended GDF5.0 to express lanes. Lane is a newly defined complex Feature. It has the following two sub-features:

- **Lane line**: It is a center line of the lane, or a rough guide line for vehicles on the lane. This feature belongs to a line-category.
- **Lane area**: It is an area in which a vehicle can legally run on the lane. This feature belongs to an area-category.

The bottom part of Fig. 4 presents lanes as geographical Features. Lane lines and lane areas are drawn, and an intersection also includes lanes. Similarly, pedestrian roads and pedestrian crossings are represented as user-defined Features in the same manner.

Extension of GDF (2) "Relationships for lanes" In addition, we have newly defined Relationships for lanes. They are treated as user-defined relationships. The following four relationships express relations from one lane to another lane.

- **Connectivity**: A pair of lanes has a connectivity relationship if a vehicle can move from one lane to another without special driving actions (e.g. lane changes).
- **Crossing**: A pair of lanes has a crossing relationship if their lane lines are crossing or their lane areas overlap. This relationship is useful to detect lanes which possibly conflict each other. Collision avoidance application utilizes this relationship.
- **Adjacency**: A pair of lanes has an adjacency relationship if they share the same border line and do not have a connectivity relationship. A pair of one lane and opposite lane also has this relationship. And, a pair of one lane and a pedestrian road also have it if they share the same border line. Divider between lanes (e.g. center divider, white line, and curb stone) is represented as Attribute.
- **Branch**: Since the number of entry points to parking is too large, regarding all of them as intersections is impractical. This relationship represents an entry point to parking from an outer lane.

The bottom part of Fig. 4 illustrates relationships between lanes. Like this figure, relationships are representable as a graph.

We are creating lane-level road maps in the real world. Figure 5 is a road map near Nagoya University.

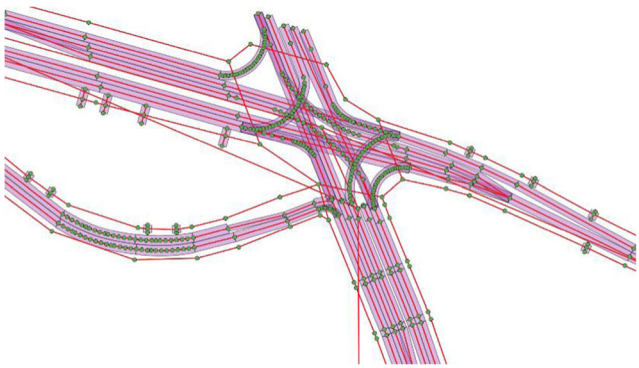


Fig. 5 Lane-level Road Map in the Real World

5.2 Static Information

Static information includes fixed properties about vehicles, pedestrians, traffic signals and other objects appearing on the map. Table 1 shows examples of static information. Of course, we can add data according to demands of traffic applications. Static information is stored in a node of DM2.0PF. IDs used in DM2.0PF are two meanings: static ID and dynamic ID. Static IDs are assigned to vehicles and pedestrians which are registered to DM2.0PF in advance. Dynamic IDs are temporary assigned by road-side units.

5.3 Dynamic Information

Dynamic information includes sensor data from vehicles, pedestrian, traffic signals, and other objects on the map. Table 2 shows examples of dynamic information. They are extendable according to demands of traffic applications. Health condition, weather, occupancy state of parking are possibly included in dynamic information.

Dynamic information is represented as a relational data stream. A relational data stream is a sequence of table rows (tuples). Its new row data is continuously produced. Each row data has a timestamp when the data was produced.

Data in dynamic information may contain static ID or dynamic ID. When a vehicle (pedestrian) not registered is detected by a third-person's sensor, dynamic ID is temporary assigned to the data. Dynamic ID is used together with physical appearance (e.g. color, size). Physical appearance is useful to merge duplicated data from different sensors.

5.4 Predicted Information

Predicated information is produced by computation. Each predicted data has two timestamps. The first timestamp is the latest timestamp of input data read by the prediction algorithm. The second timestamp is the future time at which the predicted data indicates. We assume two types of prediction algorithms:

- **Prediction based on statistics**: By the big data stored in a long term, we can predict periodic events. For example, "traffic jam on every eight o'clock" is predictable by this approach. It is applicable to long term (several minutes or longer) prediction for periodic events, but location accuracy is not so precise. Link-level road maps are often used.
- **Prediction based on dynamic information**: It uses dynamic information as input data, and predicts by using physical law and traffic rules. Although location accuracy is more precise than above, prediction time range is short (several seconds). Lane-level road maps are often used.

Table 1 Static information

	Data Name
Vehicle	Static ID, Type, Model, Number, Size, Color, Information terminal device, Automotive insurance, …
Pedestrian	Static ID, Name, Gender, Age, Phone number, insurance, …
Traffic Signal	Signal ID, Cycle pattern, Type, Related lane ID, Height, …

Table 2 Dynamic information

	Data Name
Vehicle	Timestamp, Static ID or Dynamic ID, Current Lane ID, 3D Position, Speed, Headway, Destination, Driver ID, Wiper, Light, Winker, Fuel,… (for Dynamic ID) Model, Size, Color
Pedestrian	Timestamp, Static ID or Dynamic ID, Current Lane ID, 3D Position, Speed, Headway, Destination, … (for Dynamic ID) Height, Body shape, Color of clothes, …
Traffic Signal	Timestamp, Signal ID, State (color), Mode, …
Road	Timestamp, Lane ID, Surface condition, Recent average travel time, …

5.5 Representation of Empty and Unknown Areas

Even if autonomous driving systems become popular in the future, moving objects which do not have any sensors and communication equipment may be still active in the traffic society. Position data of such moving objects are to be reported by sensors on a third person (Fig. 6).

An area not covered by any sensors is straightforwardly regarded as unknown area. On the other hand, an area covered by one or more sensors is classified into occupied area, empty area and unknown area. We need to share the result of three-class area classification.

Unknown area is chosen when an area-classification method returns the result with insufficient confidence. Occupied area with low confidence may contain non-existent objects (noise) caused by a false recognition. Empty area with low confidence may contain undetected objects.

5.5.1 State Propagation of Empty and Unknown Areas

Occupancy grid map [14] is an existing method to treat the result of area-classification. Occupancy grid map divides a space into small cells. Each cell maintains a state. A state is occupied, empty and unknown. If we want just sharing the latest result of area-classification, occupancy grid map is sufficient. However, when we would like to predict future transition of areas, occupancy grid map is insufficient. Generally,

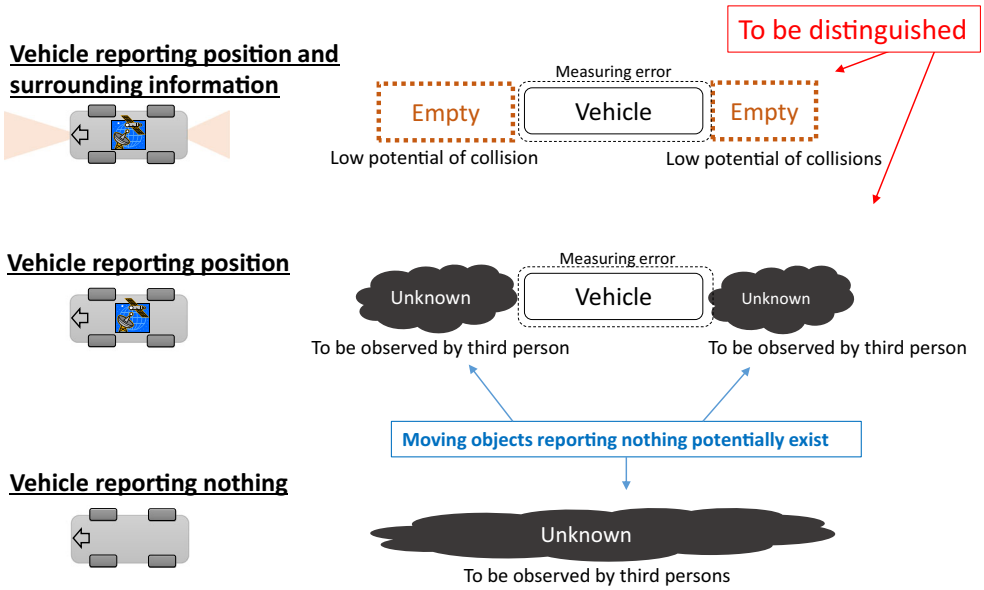


Fig. 6 Empty areas and unknown areas

prediction is done by computing future movement of each object and propagating empty and unknown areas.

For example, suppose that one cell is currently empty. We would like to predict states of surrounding cells in the next time interval. The current empty area is propagated to surrounding cells, but the propagation is not even. We have to consider physical law and traffic rules in the propagation.

For future prediction, we model an empty area **as empty object**, which is a virtual object moving on the map. An empty object has its shape, propagation velocity and existence probability. The initial shape of empty object is formed by unoccupied area from sensors. If unoccupied area is crossing multiple lanes and pedestrian roads, the area is divided into subareas for each lane and pedestrian road. Each subarea is

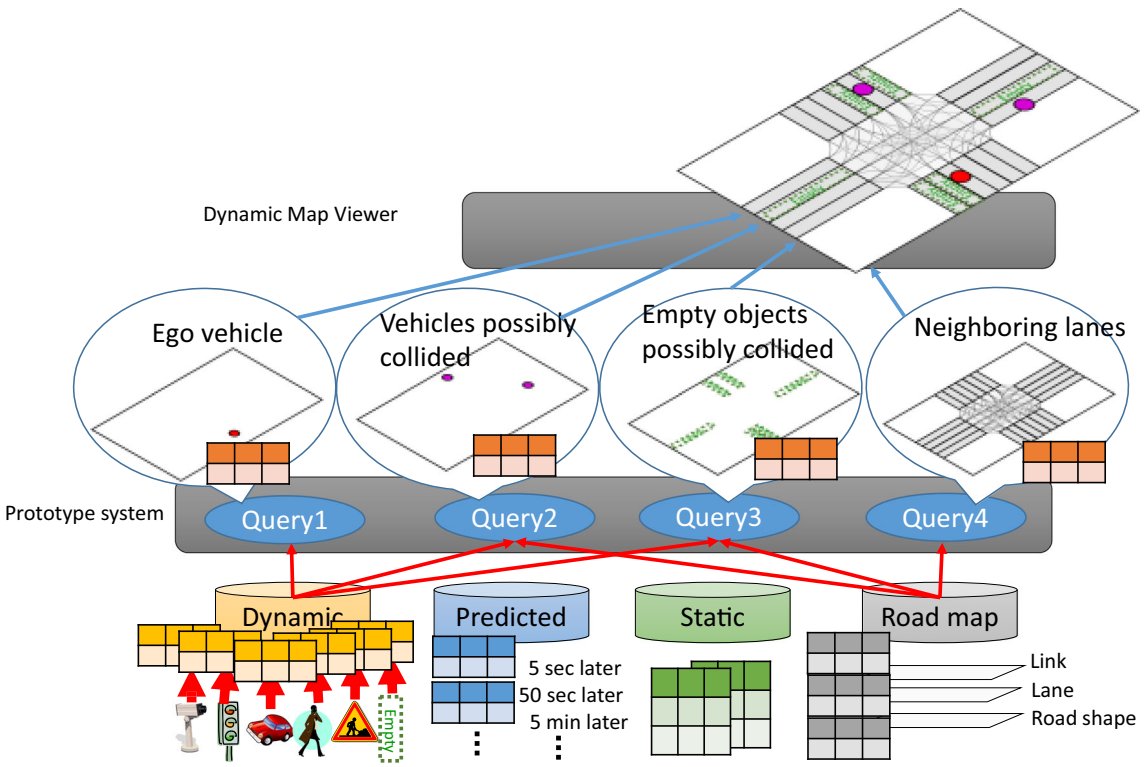


Fig. 7 Querying on DM2.0PF

```
MASTER master_1, …
SELECT attr_1, …
FROM source_1 [windowsize_1], …
WHERE conditions
GROUP BY key_1,…
```

Fig. 8 Syntax of query

regarded as one empty object. The propagation velocity of empty object is given by the following rules:

- **Lanes on highway**: An empty object moves according to lane. Its speed is estimated within an interval [minimum legal speed, maximum legal speed].
- **Lanes on general road**: An empty object moves according to lane. Its speed is estimated within an interval [0, maximum legal speed].
- **Pedestrian roads**: An empty object shrinks over time (without movement).

Existence probability of empty object decreases over time. If the probability becomes lower than a threshold, the empty object is deleted. And, the corresponding area becomes unknown.

6 Query Language for Dynamic Maps

As mentioned in Section 4.2, DM2.0PF provides a SQL-like query language. Like Fig. 7, all types of information are represented as relations (tables). A developer of traffic applications specifies a query with filter conditions and join conditions for underlying relations.

Error! Reference source not found. Figure 8 shows a syntax of our query language. MASTER clause gives event names to trigger the query. When a new data is delivered from the event source, the query is repeatedly evaluated and the query result is generated. A query with a MASTER clause is regarded as a continuous query. If a MASTER clause is omitted, it is regarded as one-shot query. SELECT, FROM, WHERE and GROUP BY clauses have almost same meanings in SQL. A list of relation names referred in the query is written in a FROM clause. And, we can specify sliding-window on streaming data in FROM clause.

Our query language makes application development more flexible. For example, when we would like to change different level of road maps, we just change a relation name referred in a query.

7 Cooperative Merging Assistance Service on the DM2.0PF Prototype

To confirm applicability to traffic services, we have implemented cooperative merging assistance service on our platform (Fig. 9). This service is one of examples described in Section 2. And, it is an example of real-time applications utilizing edge computing.

Here, we assume a merging point in T-junction. This service optimizes traffic flow at the merging point. It sends an assist message to approaching vehicles. According to the

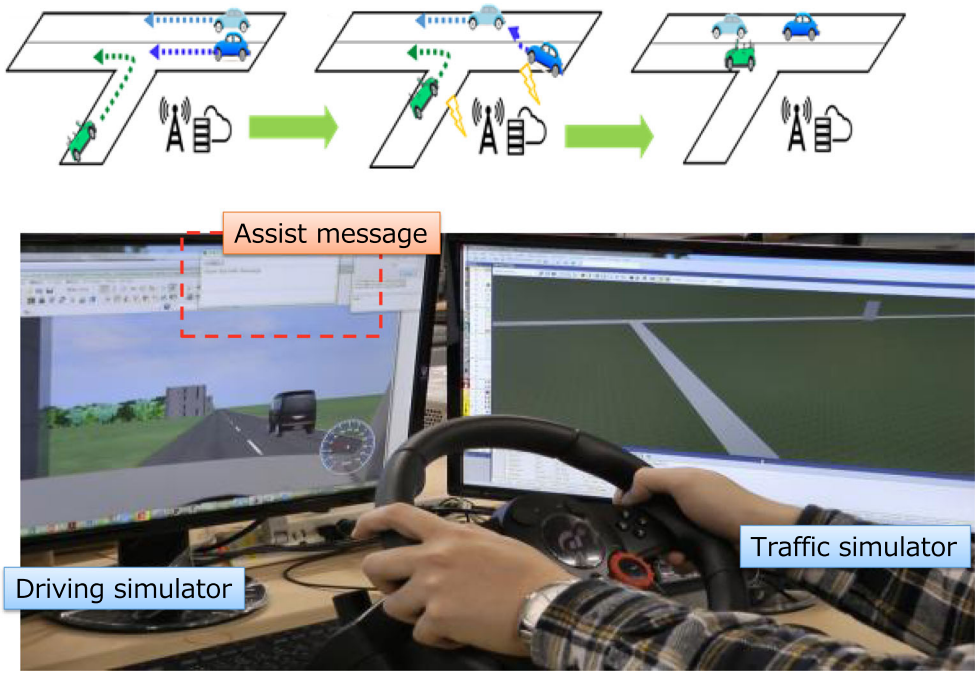


Fig. 9 Cooperative merging assistance service

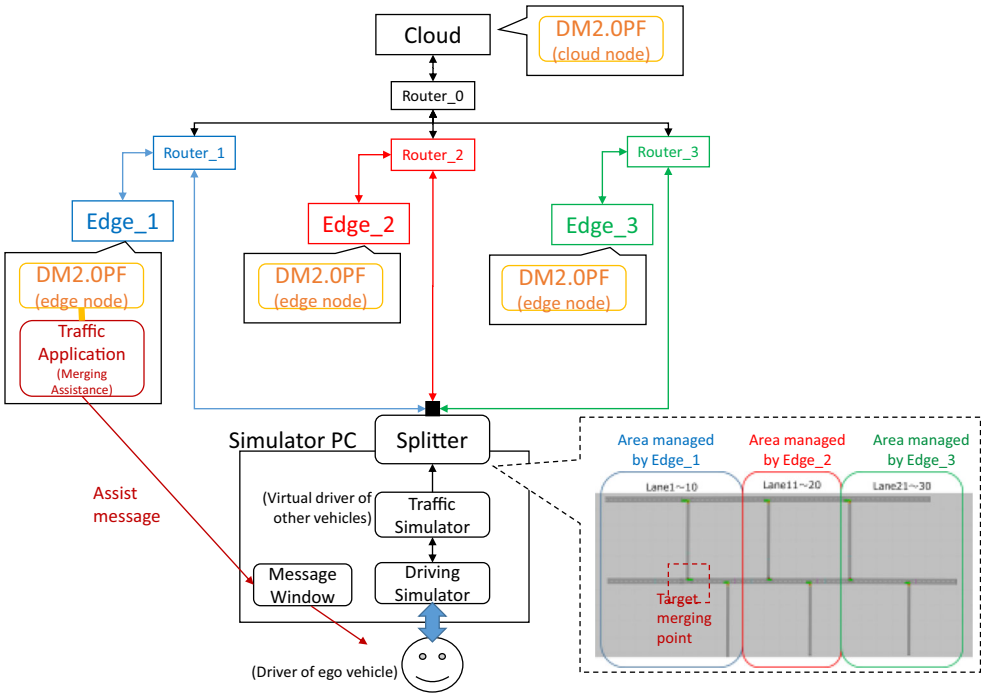


Fig. 10 Experiment environment with simulator

message, each vehicle performs speed-up/speed-down/lane-change. In Fig. 9, the blue car moves to another lane for making a space for the green car. In this service, DM2.0PF is used for processing queries to retrieve lanes, and to detect positions of vehicles approaching to the merging point.

Currently, real vehicles are not available in our project (an experimental environment in the real world is under development). Instead, we used simulators to generate input sensor data. Our platform is able to connect to a driving simulator and a traffic simulator. As shown in Fig. 10, our simulation environment combines both driving simulator and traffic simulator simultaneously. Ego vehicle is controlled by driving simulator (UC-win/Road). Other vehicles are controlled by traffic simulator (PTV Vissim). Generated data are sent to edge nodes of DM2.0PF. In Fig. 10, there are one cloud node and three edge nodes. Each edge node is assigned to its management area. Each edge node receives sensor data from vehicles in the corresponding management area. Cloud node manages a special table which contains a pair of vehicles and the corresponding edge node.

An application process for cooperative merging assistance service is deployed on an edge node (e.g. Edge_1 in Fig. 10). The application process submits queries to local DM2.0PF. In this use case, we need to monitor the target merging point continuously. Thus, it produces queries to get lane-level road maps and dynamic information (vehicle's position and velocity) near the merging point. When the application process finds a set of conflicting vehicles, it generates an assist message. We also implemented simple merging assistance algorithm, but it is out of scope of this paper.

Figure 9 also includes a screenshot of our simulation environment. Although sensor data in this section are not real data from real vehicles, our DM2.0PF prototype successfully processes data from our simulation environment. We have confirmed that it has enough capabilities for traffic applications shown in Fig. 1.

Table 3 Scope of data management

	Scope of data management		
	In-vehicle information management	Local area information management	City-scale information management
AEDSMS [12]	Continuous query	–	–
SAFE SPOT [4]	One-shot query	One-shot query	–
CarStream [13]	–	–	One-shot query Continuous query
DM2.0PF prototype	One-shot query Continuous query	One-shot query Continuous query	One-shot query Continuous query

Finally, we present a comparison result of several systems from the view point of the scope of data management (Table 3). Thanks to the architecture leveraging clouds/edges/embedded systems, our DM2.0PF prototype covers the widest scope.

8 Conclusion and Future Research Issues

This paper proposes DynamicMap2.0 platform, which is a distributed information platform for city-scale traffic data management. Its distributed architecture is composed of cloud servers, edge servers and embedded devices. The platform deals with dynamic, static, predicted information and road maps with three levels. And, it provides query processing scheme for these data. A notion of empty objects is newly introduced to explicitly share current state of lanes. Our platform contributes to support development of advanced traffic applications (e.g. personalized navigation, smooth traffic flow, advanced driver assistance systems). We developed a cooperative merging assistance system on the prototype of DM2.0PF and verified that our DM2.0PF operated correctly.

There are some future research issues. The first one is doing field operation tests in the real road. The second one is large-scale simulation. The real road is difficult to increase the number of vehicles joining the experiment. We will evaluate our platform in large simulation settings.

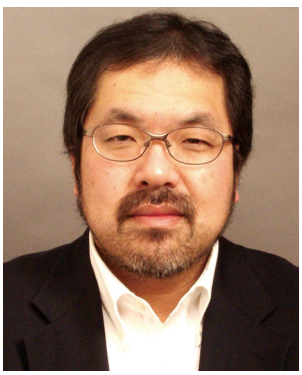
Acknowledgments This research is partially supported by JST COI STREAM and JST OPERA.

The authors would like to thank all members of Dynamic Map 2.0 Consortium.

References

1. "High Definition Lidar HDL-64E", http://velodynelidar.com/lidar/products/brochure/HDL-64E%20Data%20Sheet.pdf
2. "SIP-adus", http://www.sip-adus.jp/
3. "Dynamic Map 2.0 Consortium", http://www.nces.i.nagoya-u.ac.jp/dm2/
4. "SP7 - SCORE - SAFESPOT Core Architecture, LDM API and Usage Reference", http://www.safespot-eu.org/documents/SF_D7.3.1_Annex2_LDM_API_and_Usage_Reference_v0.7.pdf
5. "Intelligent Transport Systems - Extension of map database specification for Local Dynamic Map for Applications for Cooperative ITS"', ISO/TS 17931:2013(E)
6. "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM)", ETSI EN 302 895 V1.1.1 (2014–09)
7. "Navigation Data Standard", NDS Association, https://www.nds-association.org/
8. "'HERE HD Live Map", http://360.here.com/tag/hd-live-map/
9. "ERTICO | ITS Europe", http://ertico.com/
10. "SENSORIS", http://sensor-is.org/
11. "Intelligent Transport Systems - Geographic Data Files (GDF) - GDF5.0", ISO 14825:2011(E)
12. Yamaguchi, A., Nakamoto, Y., Sato, K., Ishikawa, Y., Watanabe, Y., Honda, S., Takada, H.: AEDSMS: automotive embedded data stream management system. In: Proceeding of the International Conference on Data Engineering, pp. 1292–1303 (2015)
13. Zhang, M., Wo, T., Xie, T., Lin, X., Liu, Y.: CarStream: an industrial system of big data processing for internet-of-vehicles. The Proceedings of the VLDB Endowment. **10**(12), 1766–1777 (2017)
14. Elfes, A.: Using occupancy grids for Mobile robot perception and navigation. Computer. **22**(6), 46–57 (1989)

Yousuke WATANABE received M.E. and Dr.E. degrees in University of Tsukuba in 2003 and 2006. In 2014, he joined Institute of Innovation for Future Society, Nagoya University, as a designated associate professor. His research interests include data stream processing and information integration. He is a member of the Database Society of Japan, IEICE, and ACM.



Kenya SATO is a professor of Doshisha University, Kyoto, Japan, where he has been since 2004. He also currently leads the Mobility Research Center of the university, and serves as a designated professor of Nagoya University. He received the BE and ME degree from Osaka University, and also received the Ph.D. degree from Nara Institute of Science and Technology. During 1991–1994 Dr. Sato was a visiting researcher at Computer Science Department, Stanford University, and he was a chief technologies of Automotive Multimedia Interface Collaboration in Michigan, U.S. in 2001–2003. His research interests include network architecture, distributed systems, and ITS. Professor Sato is a member of Japan head of delegation to ISO ITS technical committee.

Hiroaki TAKADA is a professor at Institute of Innovation for Future Society, Nagoya University. He is also a professor and the Executive Director of the Center for Embedded Computing Systems (NCES), the Graduate School of Informatics, Nagoya University. He received his Ph.D. degree in Information Science from University of Tokyo in 1996. He was a Research Associate at University of Tokyo from 1989 to 1997, and was a Lecturer and then an Associate Professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IEICE, JSSST, and JSAE.