A Knowledge-Spatial Architecture for Processing Dynamic Maps in Automated Driving

Haonan Qiu[1,2], Adel Ayara[1], and Birte Glimm[2]

[1] BMW Car IT GmbH, Ulm, Germany,
{haonan.qiu,adel.ayara}@bmw.de
[2] Institute of Artificial Intelligence, University of Ulm, Germany,
birte.glimm@uni-ulm.de

Abstract. An autonomous car needs a detailed, high-definition (HD) map to understand its surroundings. An HD map acts as a powerful virtual "sensor". Compared to traditional digital maps, high-definition maps require significantly more storage space, and a complete map cannot be stored in a navigation system. Furthermore, map data is provided in numerous heterogeneous formats. Hence, interoperability and scalability have become the main challenges of existing map processing solutions. We demonstrate how these challenges can be addressed using an interoperable knowledge-spatial architecture layer based on ontologies.

Demo Submission

1 Introduction

Motivation: Autonomous vehicles need to adhere to extremely high safety standards as failures can impact human lives. A high-definition (HD) map acts as a powerful virtual "sensor" [1]. However, as of now, there is no single, authoritative format or standard for HD maps [2]. As a result, map model development, maintenance, and integration, as well as map data exchange and sharing pose major challenges in practise. Furthermore, HD maps are extremely detailed and, therefore, require significantly more processing power and computation resources compared to standard-definition (SD) maps. The navigation system constantly requests map data streams while the car is progressing along a route and care has to be taken to provide any relevant information in time [3]. The heterogeneity and big volume characteristics of HD maps require a novel approach that allows for a generic representation of the road environment and a dynamic update mechanism. Ontologies have been used for representing road intersections for autonomous vehicles [8, 6], however, their work is neither generalized to different map formats nor is the spatial topology of lanes considered.

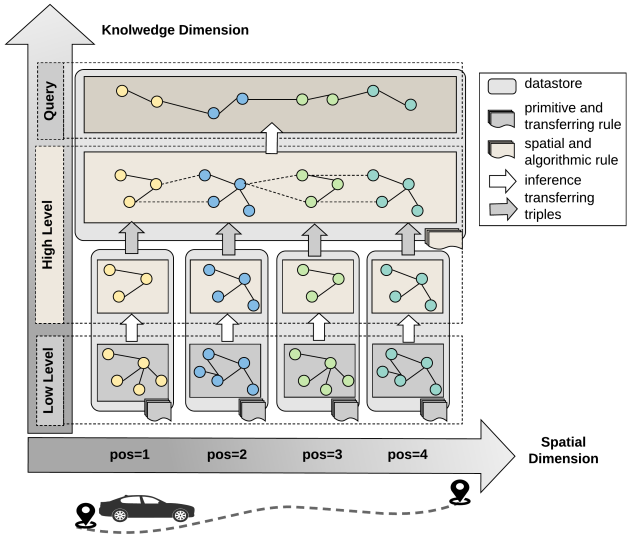


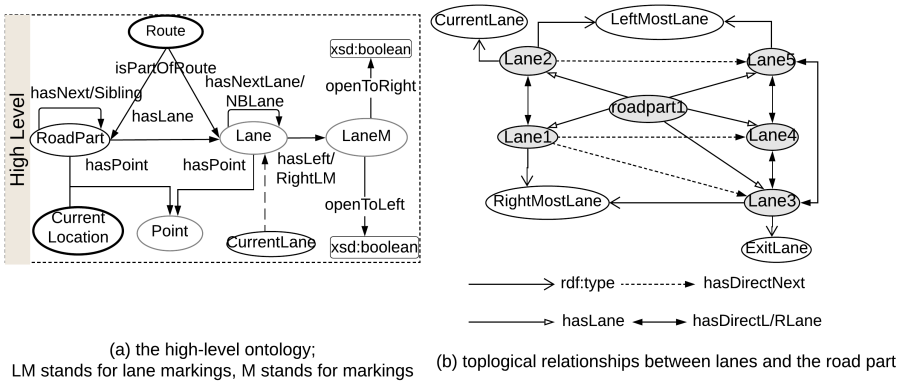
Fig. 1: The overview of knowledge-spatial architecture

Demo: We present the SmartMapApp prototype based on a knowledge-spatial architecture to deal with both knowledge abstraction and spatial reasoning on highly dynamic map data streams. We illustrate how the application works with a use-case of dynamic map updates in a highway scenario, but the approach generalizes to other scenarios. A demonstration video of the application is available at `https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/video/ISWCDemo.mp4`.

2 A Knowledge-Spatial Architecture

We present the knowledge-spatial architecture that enables autonomous vehicles to perceive their environment with a dynamic map in Figure 1.

The knowledge dimension addresses a knowledge abstraction process from the format-specific and detailed low-level ontologies to the generic high-level ontology (see the vertical axis of Fig. 1). The horizontal (time) axis represents road knowledge acquisition events, which trigger the knowledge abstraction process via spatial reasoning. Spatial reasoning considers updated vehicle motion events determined in the knowledge abstraction process and searches for spatial patterns to derive the relevant consequences of what is happening on the road. Different low-level ontologies for the different map formats can be used to feed the high-level ontology, which makes the proposed architecture very flexible as application-oriented queries, such as advanced driver assistance system (ADAS) functions, are posed over the generic high-level ontology.



(a) the high-level ontology; LM stands for lane markings, M stands for markings

(b) toplogical relationships between lanes and the road part

Fig. 2: The high-level ontology and an example of `Lane` and `RoadPart` spatial relations; hasDirectL/RLane stands for hasDirectLeft/RightLane.

The spatial dimension is orthogonal to the knowledge dimension and correlates facts that are true within a certain space. It describes the continues spatial reasoning process with respect to the updated vehicle position and dynamic road environmental knowledge. We adopt the notion of a *spatial window* with a fixed width or region in terms of geographic elements shift (slide) over a path line. Inspired by Mokbel et al. [4], we use the notion of *spatial expiration* depending on the spatial location of a moving object, e.g. a vehicle, and stored data expires only when the object leaves the spatial window. The spatial window is determined by the vehicle location and a forward and backward parameter.

Two levels of datastores are used to deal with the highly dynamic, location-aware environment, where the size of spatial streams is potentially infinite. Certain spacial events (e.g., the available map foresight of the vehicle reaches a threshold) trigger the initialisation of a new datastore for a low-level ontology. The next map data region is loaded (as defined by the specific map data format) and used to generate more abstract, high-level knowledge. While the high-level datastore uses spatial expiration for deletions, a low-level datastore is discarded once the high-level datatstore is populated.

Ontologies and rules are essential for road environmental knowledge representation and spatial reasoning. Figure 2(a) shows some of the concepts and spatial relations of the high-level ontology. One of the the low-level ontologies is presented in [7]. At the high level, a example of the spatial relations among `Lane` and `RoadPart` are shown in Fig. 2(b). The following rules are used to infer lateral relationships between lanes. The longitude relationships such as successor relationships are modeled in similar fashion.

$$\mathsf{hasRLane}(x, y) \leftarrow \mathsf{hasDirectRLane}(x, y)$$
$$\mathsf{hasRLane}(x, z) \leftarrow \mathsf{hasDirectRLane}(x, y),\ \mathsf{hasRLane}(y, z).$$
$$\mathsf{RightMostLane}(x) \leftarrow \mathsf{Lane}(x),\ \mathsf{Lane}(y),\ \mathtt{NOT}\ \mathsf{hasRLane}(x,\ y).$$

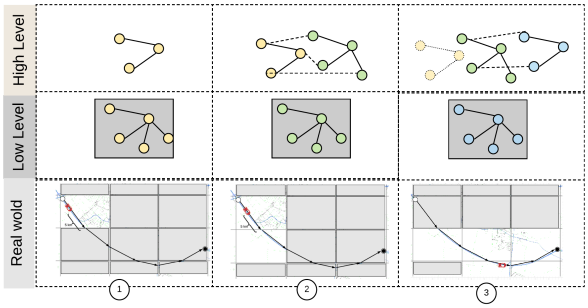


Fig. 3: An example of a route path over map tiles

3 Demo Application

To demonstrate the ontology-based approach for processing dynamic map data, we have implemented a prototype called *SmartMapApp* for the knowledge-spatial architecture described above and it is implemented using RDFox 3.0.1 with the provided Java APIs [5].

Dynamic map update scenario: We illustrate the demo scenario of a car progressing along a route on a highway using three snapshots (see Figure 3). In each snapshot, the vehicle's external and internal worlds are presented. In snapshot 1, the vehicle initialises its world view with some low-level map data, which results in a high-level road view. As soon as the car starts to move, it triggers a continuous pre-fetching query with a spatial window with the forward parameter set to 3 km. In snapshot 2, the system pre-loads data for a new map tile and extends the high-level road view. In snapshot 3, while the system incrementally updates the road environmental knowledge, it also continuously checks if any road parts are "out of window" based on the backward parameter (e.g., 3 km) of the spatial window and deletes them. In addition, the system also provides a lane change notification containing maneuver steps to the car for reaching the targeted lane based on the route.

SmartMapApp: Figure 4 shows a screenshot of the SmartMapApp GUI. The simulation is initialized with three types of input files: a JSON file containing a sequence of positions encoded in the World Geodetic System 1984, map data triples extracted from a map database, and a route represented as triples containing road parts and lanes. After the initialization, the system starts to simulate the progressing of the car by using the periodically updated position and the given route. The *Position Received* section shows the received position. The *Current Car Info* section shows the current car situation in the lane and the road (e.g the travel distance). Pre-loading and deletion information is also displayed whenever the spatial reasoning triggers the pre-loading and deletion processes, respectively. The *Maneuver Steps* shows the steps for changing lanes and the *Maneuver Explanation* displays the reasons for lane change notification.