Daniel Burgstahler, Christoph Peusens, Doreen Böhnstedt and Ralf Steinmetz: Horizon.KOM: A First Step Towards an Open Vehicular Horizon Provider. In: Proceedings of the 2nd International Conference on Vehicle Technology and Intelligent Transportation Systems (VEHITS 2016) April 2016.

Horizon.KOM: A First Step Towards an Open Vehicular Horizon Provider

Daniel Burgstahler, Christoph Peusens, Doreen Boehnstedt and Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Darmstadt, Germany
{*firstName.lastName*}*@KOM.tu-darmstadt.de*

Keywords: ADASIS, Electronic Horizon, Most Probable Path, Tree-Structure, Horizon Size Determination, Traffic Information, Open Street Map

Abstract: Modern vehicles are commonly equipped with several sensors to gather information about the direct environment. Due to the physical limitation of these sensors, the detection range and field of view are limited to the direct vicinity. This limits the functionality of driver assistance systems. A way to overcome this issue is to use digital road maps to generate a so called electronic horizon as virtual sensor about the environment ahead. All known electronic horizon providers are closed source and owned by companies. In the work at hand we present our concept of an electronic horizon provider that we have published as open source.

1 INTRODUCTION

Modern vehicles are equipped more and more with assistance systems to increase comfort, economy and safety. These systems are based on sensors that observe the vehicle's environment. However, all sensors are limited in terms of sensing range and view angle. The environment behind other objects, as well as in a larger distance is commonly not visible. Developing new applications, serving advanced tasks in a more complex environment, raises the desire of an extended view. The Adaptive Front Lighting (AFL) (Durekovic et al., 2011) can serve as example. It adjusts the headlight alignment to achieve an optimal illumination of the road. Assuming knowledge about the road geometry, algorithms can be applied to determine vehicle parameters and settings, which lead to a decreasing energy consumption while maintaining the same overall speed and comfort. A common technology to provide an extended view is to use a so-called electronic horizon (eHorizon), that provides information about the road network ahead of a vehicle. An according illustration is given in Figure 1. It enables applications to perceive the road geometry, including additional information, with an increased range and field of view. However, to the best of our knowledge all existing implementation for such a eHorizon provider are closed source and part of comercial products. We have developed a model that extracts relevant road information from a digital map and provides those to other applications. We name

Figure 1: Example illustrating the eHorizon.

our solution Horizon.KOM and provide the source code to the community. We will start with a short introduction of the ADASIS specification, which is used for the standardized distribution of road information via a vehicle on board data bus. Afterwards we will compose our model including several components such as the dynamic data storage, data processing, and most probable path prediction. The purpose and tasks of each module will be explained in detail supported by visual examples. At the end of this paper we will prove the feasibility of our model by introducing a prototype implementation using an android operated mobile device and road maps from the OpenStreetMap project.

2 RELATED WORK

As mentioned before we make use the ADASIS specification (Ress et al., 2005; Ress et al., 2008), in particular version 2. It was designed for Advanced Driver

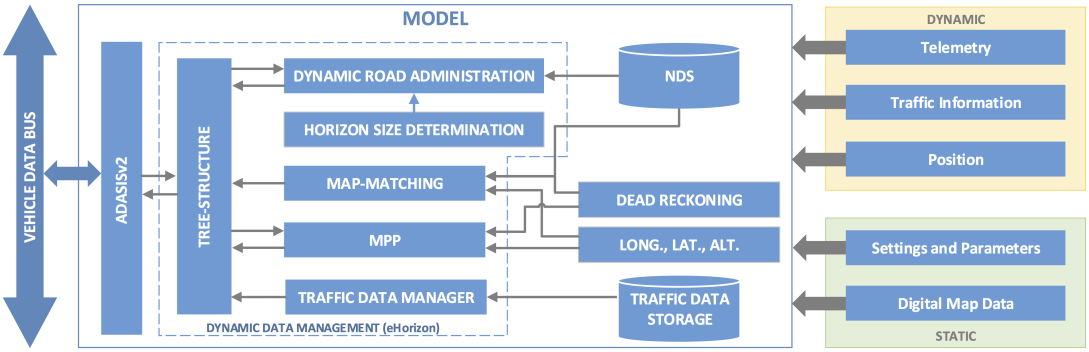

Figure 2: Schematic illustration of the dynamic data management structure.

Assistance Systems (ADAS) in order to transmit relevant road information via a vehicle data bus. In this manner our Horizon.KOM will be considered as an ADAS eHorizon provider, and connected clients will be denoted as ADAS application.

- The ADAS Horizon knows the vehicles current position, velocity and driving direction. It also has access to at least one digital road map. Using this information, the ADAS Horizon will construct a digital eHorizon holding all relevant information about the road network ahead of the vehicle.
- An ADAS Application can decode and reconstruct the transmitted eHorizon, by implementing a so called ADASIS Reconstructor. This enables the application to use information about the road network ahead of the vehicle without the constraint of implementing its own eHorizon based on a locally stored digital road map.

ADASISv2 supports 4 different modes of operation determining the amount of data which will be transfered. Mode *0* only transmits data about the *Most Probable Path*. Information regarding alternative routes or intersections along this path is excluded. Mode *1* adds additional data about the so-called *Stubs* along the *MPP*. Stubs denote intersections, including first curvature values. Mode *2* extends mode *1* by adding information about intersecting road segments along the *MPP*, excluding second level intersections. Mode *3* considers the entire road network and has no restrictions in therms of maximum levels and intersections. Each road segment is represented as a one dimensional line and is free of absolute geographical coordinates. Each segment representation has a fixed length and includes curvature values, denoted with offset values from zero to segment length, to preserve the road shape information. The maximum length of each segment is given by 8190m (13 Bit). A road segment from the digital map exceeding this maximum length has to be split. The current position of the vehicle is given by an offset value along the currently used road segment. Similar notation applies to all values that are bound to an fixed geographical position. On the provider side two parameters determine the eHorizon size. A value for the *Maximum Length of Transmitted Path* gives a maximum offset threshold, a value for *Current Length of Transmitted Path* describes the maximum offset of data that has already been sent. ADASISv2 incorporates 7 message types which are used to operate the data transmission. The *POSITION Message* is used to distribute the vehicles current position and velocity. It includes a reference to the currently used road segment. Using the *SEGMENT Message*, information about road segments are transmitted, including the road type, speed limits or other attributes. *STUB Messages* are sent to provide information about intersections and connecting road segments. The *PROFILE Message* is used to define the profile type of a segment, which can either be discrete, linear or non-linear. The *ATTACHMENT Message* contains information about special elements of the road network like traffic sings or traffic lights. Used settings like the operating mode, used units or version number of the ADAS Horizon Provider are distributed by using the *META-DATA Message*. Each message is only sent once by the ADAS Horizon Provider, except if an ADAS application requests a retransmission by sending a *RETRANSMISSION-REQUEST Message*. More detailed information about the ADASISv2 specification can be found in (Ress et al., 2005) (Ress et al., 2008). For data storage of digital road maps, different formates have been developed for different purposes. For plain data storage or archiving, formats like the Geographic Data File (GDF) (Volker, Walter, 1996) or the german *Amtliches Topologisches Kartographisches Informationssystem* (ATKI) (Volker, Walter, 1996) are used. The OpenStreetMap project has developed its own *OSM-XML* format, that is based on an XML-structure which has no restrictions in therms of extensions.The

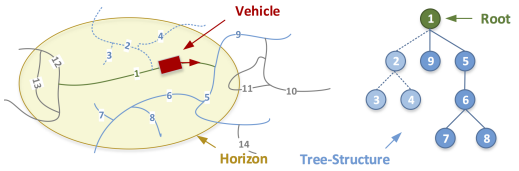

Figure 3: Movement within the current tree structure. Passed nodes are neglected.

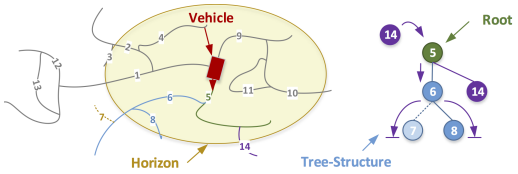

Figure 4: Further movement within the tree structure. Adding new nodes to the tree structure.

GDF, ATKI and OSM-XML formats all store data in a serial manner, which is very costly in terms of searching for a specific element. Therefore, road maps used for runtime applications are commonly compiled to a binary format to reduce storage requirements and increase data access speeds. However, binary formates are not designed for updates later on. Another important topic is *Map-Matching* that becomes important for every mobile application that makes uses of digital road map data. Map-Matching describes the action of mapping the vehicles geographical position onto the map and determining the currently used road segment. Several algorithms to solve this task have been published in literature. Kim et al. (Kim et al., 1996) presented already in 1996 a light weight approach of mapping positions to the closest point on the map. However, in order to achieve overall good mapping results, the algorithm has to compensate inaccuracies and variations in positioning data. Quddus et al. (Quddus et al., 2003) introduced in 2003 a more advanced and accurate approach, that uses the recent position history and dead reckoning to identify the position of the vehicle on a map. Dead reckoning uses other sensor data like steering angle and vehicle velocity to calculate deltas in the position. It is also used to keep up navigation, if for a short period of time no satellite position signal is received, e.g. while driving through a tunnel. Several other approaches, serving various requirements can be found in literature. For geographical location determination, we consider Global Positioning System (GPS), which can achieve an accuracy around 1 to 3 meter for civil services, by use of the European Geostationary Navigation Overlay Service (EGNOS) or Wide Area Augmentation System (WAAS). An important source for information about the current traffic situation are Traffic and Travel Information (TTI), which are also supported conceptually within Horizon.KOM.

3 MODEL

The model of Horizon.KOM is depicted in Figure 2. It is based on dynamic and static data provided by the vehicle's sensors and external sources. Dynamic data varies over time and includes the geographic position as well as traffic information. Additional data from the vehicle is also provided via the internal data bus. Digital map data is assumed to be mostly static. Horizon.KOM is basically composed of six components. The *Tree-Structure (I)* is used to store the road segments located ahead of the vehicle including their characteristics and additional information. This is a central element that combines all other components and further is used as interface to ADASIS. The size of the vehicles eHorizon, which effects the number of nodes (road segments) stored at the *Tree-Structure* is defined by the *eHorizon Size Determination (II)* and is a function of the vehicle's velocity. A predicted route, which the vehicle most likely will take, is defined by the *Most Probable Path (MPP)(III)*. It has an important influence on the shape and the size of the vehicles eHorizon area. Based on this eHorizon area and information provided by a digital map, the *Dynamic Road Administration (IV)* component defines the data that is stored in the *Tree-Structure*. The *Map Matching (V)* component keeps track and enriches the *Tree-Structure* of the vehicles current position. Using the *Traffic Data Manager (VI)* the *Tree-Structure* is extended by external Traffic Data Information along the eHorizon. These components are explained in more detail in the following.

(I) Tree-Structure: A basic component of the vehicle's eHorizon is the digital road map. In order to enable a cost efficient definition of the vehicles eHorizon in terms of computational power, the network of road segments that is mapped on this eHorizon has to be stored efficiently. Therefore we propose a tree-based data structure to store road segments that are ahead of the vehicle. Due to this *Tree-Structure*, the map is easily extensible and allows a fast adaption of new road segments or other entities. The currently used road segment is represented by the root of the tree. Each connecting road segment is linked to this root node and thus mapped as a second-tier node to the *Tree-Structure*. This is a multi-tier representation and in general road segments are accessible from the segment a tier above. Road intersections are represented by connections between nodes of different tier levels. The driving direction of the vehicle is used

to exclude road segments from the *Tree-Structure* that have already been passed. In order to limit the number of nodes within the *Tree-Structure*, we limit the number of maximum tiers along the *Most Probable Path (MPP)* and limit the size of the covered geographical area. Thus, leaves of the tree represent either a dead end or a road that continues beyond the eHorizon area. The *Tree-Structure* is updates each time the vehicle moves onto another road segment. Then this segment becomes the root node. If the vehicle passes an intersection, all passed segments are marked as not accessible but remain in the *Tree-Structure* until the next update. An example if given in figure 3, here the vehicle has passed road segment 2 and nodes 2, 3 and 4 remain in the *Tree-Structure* marked as not accessible. As soon as the vehicle moves onto a new segment, the *Tree-Structure* is updated and the new segment becomes the root node. The *Tree-Structure* has now to be reordered and all nodes that are only connected to the old root node are removed. New road segments are added accordingly to the eHorizon size as explained before. An example if given in figure 4, here segment 5 becomes the new root node and nodes 2, 3, 4, and 9 vanish by removing node 1 and, node 14 is added. Although node 7 has left the eHorizon area, it will remain in the *Tree-Structure*, due to the existing possibility of entering the eHorizon area again. To store road segments as node, we use a linear space representation, i.e., a linear line with a defined length, that provides an additional simplification and reduction of overhead. This notation is related to the segment representation of the ADASIS specification and provides an easy data exchange between ADASIS and the *Tree-Structure*. Therefore, the natural shape of a segment defined by its intermediate points (absolute coordinates) is removed. To compensate the information loss, the natural shape is translated into a list of curvature values that are distributed along the segment, denoted with offset values. All location depending information, including the current vehicle position, is defined by an offset value starting from the beginning of the road segment and is stored inside each node. Each node contains at minimum information about its length, road type, and curvature values. Additional information of any type can be stored in a similar manner.

(II) eHorizon Size Determination: The size of the eHorizon defines which nodes are stored in the *Tree-Structure*. We have to limit the size of this eHorizon due to limited resources in terms of processing power and storage space. Moreover, a dynamic size of the eHorizon is required in order to cope with different application scenarios. A vehicle driving with high speed, e.g., on a highway, needs an eHorizon of several kilometers ahead. However, considering urban areas, a smaler eHorizon is feasible. We propose two different approaches to define the size of the eHorizon based on the vehicles context. (I) By identifying the context of the vehicle, with respect of the road type (e.g. city, highway, ...) an appropriate eHorizon can be defined. (II) We can define the eHorizon as a function of the vehicle velocity. An increased velocity indicates road types such as a highway. However, we need to consider a temporary reduced speed due to characteristics of the route or traffic jams on the highway, leading to a very low velocity. In the following we consider the vehicle velocity to define the size of the eHorizon due to the fact that semantic description of the road types may not always be provided by digital maps. The maximum radius of the eHorizon is defined by the vehicle velocity (v) and the parameter d as shown in Equation 1. The parameter d can be used to adapt the overall size of the eHorizon.

$$Max_{eHorizon,d} = e^{log_2(v)*d} \tag{1}$$

However, the size of the eHorizon is further influenced by four adaptations. (I) As mentioned before, reducing the vehicle's speed would result in a reduced vehicles eHorizon. We will not truncate the tree-structure due to the reduced speed, but new nodes will only be added based on the new reduced size of the eHorizon. (II) The depth of the tree has to be limited, i.e., to limit the number n of tier-levels. On a highway a very large eHorizon is necessary due to vehicle velocity. On an exit toward a city the vehicle is still on a highway but the MPP goes toward an urban area that would cause in a high overhead, especially if the vehicle is not exiting exiting the highway. Thus the tier-levels have to be limited. (III) Moreover, we have to handle nodes that are located within the eHorizon, but are not accessible via a route stored in the tree-structure. We assume that these nodes will not be added to the tree-structure, due to the missing accessibility in terms of our current tree-structure state. Road segment 11 in Figure 3 illustrates such a scenario. (IV) The last adaptation is the influence of the MPP. By predicting the route with the highest probability the respective path nodes can be included with a higher priority. Therefore, the MPP is stored with an increased level of detail and with an increased number of tier-levels. This adaptation will result in an more elongated vehicle eHorizon.

(III) Most Probable Path: In the previous sections, we defined a system that is able to map all relevant road segments, including any related data, within a specific distance efficiently in a tree based structure. In the following the model is extended by a mechanism that predicts the MPP the vehicle will most likely use. In order to predict the MPP, knowledge

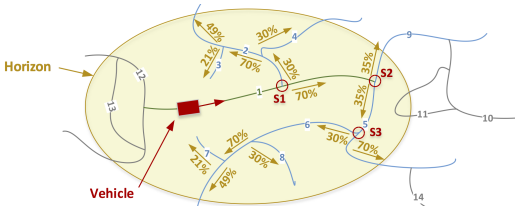

Figure 5: Example for the probability distribution of the previously introduced scenario.

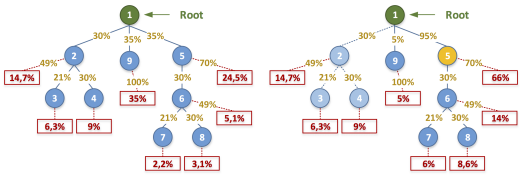

Figure 6: Example for the probability distribution according to introduced scenario.

about the vehicles final destination is not mandatory and can therefore be assumed as not given. We further assume an infinite route which is only limited by the size of the eHorizon. Thus there will always be a successor node, and therefore one of the child nodes will be the next root node. Moreover, we assume the absolute vehicle position as not necessary, since the root node and vehicle offset, stored in the *Tree-Structure*, provide sufficient information. As already defined, all child nodes of a parent node are located in driving direction of the vehicle and denote the set of available successor road segments. A probability distribution is assigned to each set of available successor nodes, associated the respective parent node. The probability value of each node is based on a predefined distribution table, which holds values for any kind of intersection, road type, and road class combination. The table can be adapted individually based on statistical data. An improvement of the table can be achieved by searching and evaluating regularly driven routes, e.g., the drivers route between home and workplace. Figure 5 shows an example probability distribution of the previously introduced scenario. The vehicle is approaching intersection *S1*. With a probability of 70% the vehicle will continue on road segment no. 1 and it will turn into the connecting segment no. 2 (side road) with a probability of 30%. At intersection *S2* two options exist (segments 5 & 9), both of the same road class and thus denoted with 50%. This leads to a value of 35% in the overall probability distribution of available successor nodes. Figure 6 depicts on the left side the probability distribution inside the *Tree-Structure*. The right side depicts the exclusion of passed nodes. When the vehicle is approaching an intersection, indicators like turning signals, the vehicles speed profile, steering wheel position, and as well the currently selected lane, are used to enhance the *MPP*. The vehicle is approaching intersection *S2* with turning signals set and decreasing velocity (v). We have to differentiate between ambiguous and unambiguous probability distribution. If the road segment the vehicle will turn into is vague, ambiguous probability distribution is used. Thus the MPP can only be determined by the values of the predefined distribution table. In contrast, an unambiguous probability distribution provides a high probability that the next road segment can be predicted correctly. This is the case if the previously mentioned indicators are used. The MPP can be extracted by traversing the *Tree-Structure* along the highest probabilities of each child node.

(IV) Dynamic Road Administration: The *Dynamic Road Administration* fills the *Tree-Structure* with data and maintains its up-to-dateness as iterative process. Furthermore, it is used to enrich the data with additional information that is not part of the digital map, if available. The basic concept is in a first step to determine all road segments that are located the vehicle driving direction and that are within the eHorizon area according to the *eHorizon Size Determination*. We assume that the minimum *Tree-Structure* always contains a valid root node. At the very first, the root node will be determined by the *Map Matching* in an initial process. Next, the *Dynamic Road Administration* loads all remaining road segments from the digital map and stores them in the *Tree-Structure*. During movement the component continuously traverses the *Tree-Structure* to determine and add missing road segments. The processing of each road segment contains a transformation into linear space and computation of additional information.

(V) Map Matching: The position is provided by a GNSS system, e.g. GPS, but with limited accuracy. A *Map Matching* component can compensate this by mapping the vehicle onto a known road network. Several algorithms are known in literature to minimize the chance of mismatching. For this the recent position history and dead reckoning is used. We recommend to use the algorithm introduced by Quddus et al. (Quddus et al., 2003), that is also used in our prototype implementation. The matching of geo position and the driven distance L is combined to determine the vehicles current offset. The *Map Matching* component fulfills several tasks. First of all it updates the root node with the vehicle offset, i.e., the translation of the geographical position into a segment related value. It is also responsible to find and define the first root node when the model is initialized and, therefore, the tree-structure is still empty. Furthermore it detects a vehicles road segment transition and

updates the current root node with the respective successor node.
(VI) Traffic Data Manager: The Traffic and Travel Information (TTI) is stored separately from the *Tree-Structure*. The main responsibility of the *Traffic Data Manager*, is to sort, filter and prioritize incommoding traffic information as well as to provide a link between the stored traffic records and the road segments of the *Tree-Structure*. A traffic record can be linked to more than one road segment and even effect an entire geographical area. The separation is intended to reduce a additional computation and storage overhead.

4 IMPLEMENTATION

The prototype implementation of our model is an Android project and makes use of maps from the OpenStreetMap project. The implementation has some abstractions to reduce complexity. Due to the implementation on a mobile device, we have to cope with the absence of data telemetry, provided by the vehicle. Thus we simulate turning signals by implementing two buttons indicating an according behavior. We have implemented an own road map database including a lightweight interface using SQLite. For *Map-Matching* we have implemented an approach according to Quddus et al. (Quddus et al., 2003). Due to the absence of vehicle telemetry, we have applied a simplified algorithm to determine the *MPP*. It is assumed, that the vehicle will continue traveling on the currently selected road segment with a probability of 100%, if no signal indicator is applied. The Android application consists of two views. The first view shows several information regarding the *Map Matching* and size of the eHorizon. The second view depicts, on top of a map visualization, the currently loaded eHorizon as blue paths, the predicted *MPP* marked as red paths, the current position marked by a red dot. The *eHorizon Size Determination*, *Dynamic Road Administration*, *Most Probable Path* and *Map-Matching* are implemented as Android *Service*. Each of these services operates in parallel with access to the *Tree-Structure*, that is incorporated into the Android *Application Layer*.

5 Summary and Sources

In this paper we have introduced our eHorizon software prototype. In summary, our model of the eHorizon provides a well suited data storage model in form of a *Tree-Structure* to natively and dynamically store the required road information. The presented *eHorizon Size Determination* provides the current eHorizon size with respect to several scenarios regarding the vehicles velocity v and driving environment, i.e., street type or urban area. We also have provisioned a method to predict and incorporate the *MPP*, including the learning of frequently used routes and runtime probability adaption. The choice of the Navigation Data Standard (NDS), accessing at least one digital map, ensures a future interoperability with other applications. To enrich the electronic eHorizon data, we also have incorporated Traffic and Travel Information (TTI) into our model. The holistic module-based approach provides the freedom of extending our eHorizon and apply adjustments to fulfill individual requirements. Furthermore, the majority of the introduced tasks are designed to be executed in parallel. We did several experiments to prove the functionality of our model. In a next step we plan a performance evaluation and a transfer into a universal Java library to be used easily on other platforms. The source code of our prototype is available under Apache 2.0 open source license on our website: `http://www.kom.tu-darmstadt.de/research-results/software-downloads/software/horizonkom/`. Furthermore we are currently working on extensions and improvements and will continuously update the available implementation.

REFERENCES

Durekovic et al. (2011). Architectures of Map-Supported ADAS. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 207–211. NAVTEQ.

Kim, J., JH, L., TH, K., WY, L., and YG, K. (1996). Node based map matching algorithm for car navigation system. In *International Symposium on Automotive Technology & Automation (29th: 1996: Florence, Italy)*.

Quddus, M., Ochieng, W. Y., Zhao, L., and Noland, R. (2003). A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7:157–167.

Ress, C., Balzer, D., Bracht, A., Durekovic, S., and Löwenau, J. (2008). ADASIS Protocol for Advanced in-vehicle Applications. In *ITS World Congress. New York*.

Ress, C., Etemad, A., Kuck, D., and Boerger, M. (2005). Electronic Horizon - Supporting ADAS Applications with Predictive Map Data. In *ITS European Congress, Hannover, Germany-TS*, volume 18.

Volker, Walter (1996). *Zuordnung von Rraumbezogenen Daten – am Beispiel der Datenmodelle ATKIS und GDF*. PhD thesis, Univerity Stuttgart.