Application Driven Environment Representation

André Dietrich, Jörg Kaiser, Sebastian Zug, and Sasanka Potluri
Department of Distributed Systems
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
Email: {dietrich, kaiser, zug, sasanka}@ivs.cs.uni-magdeburg.de

***Abstract*—Autonomous systems have to sustain in environments of growing complexity and under dynamically changing conditions. In addition, the number and the complexity of the tasks these systems have to fulfill increases constantly. To cope with these emerging problems, we present a general concept of constructing environment models. From these models, we can derive any kind of information in an application specific abstraction, which we call views. Furthermore, we describe the main problem in the generation of views as well as a possible solution to this.**

***Keywords*—Smart Environments, Environment Model, Robotics**

I. Introduction

One of the main characteristics of smart environments is the spontaneous use of ambient information, obtained from remote sensors and servers. These sources of information (mobile or stationary) can be a part of the infrastructure of a smart space itself or gathered from other systems that exploit these location-dependent information and services. The ultimate goal is to perceive the environment with the best possible quality, including as many aspects as desired. For autonomous systems, such as mobile robots or even for the emerging field of cooperating cars, the benefits of exploiting environment knowledge and available sensory information are obvious; reaching from a substantial increase of sensing range to the integration of more and more detailed aspects into the environment perception. Literally, it is possible to look around the corner or to recognize obstacles and to plan to avoid them, long way before they come into sight of a local sensor.

Perception requires two things: the input from sensors that provide some basic physical data and a model that allows to interpret this information correctly. Designers of such applications have to deal with a couple of problems. Firstly, the heterogeneity of sensors may require intimate knowledge of sensor characteristics and complex adaptation procedures of the application. Secondly, they have to solve the problem of interpreting data coming from complex mobile remote directed sensors like Kinects, laser scanners or cameras. Solving both problems requires a huge amount of specific context information, as depicted in Figure 1.

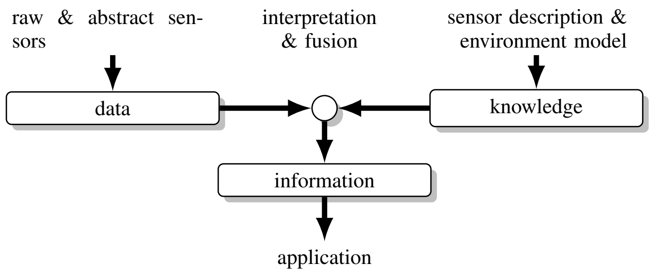

Fig. 1: Relation between sensor data and application specific information.

On the lowest level, sensor data interpretation needs some form of knowledge about the sensor type and the physical nature of observed real world entities, which can be provided by metadata. We therefore had developed expressive description formats for sensors, to support their spontaneous use [1]. While these sensor descriptions cover low level sensor interpretation and fusion, this paper focuses on an application-oriented abstraction of perception on a higher level. We propose the use of environment models as explicit knowledge for putting remote sensor data into an adequate context. We argue that cooperative systems, which exploit information from remote sensors, will need an explicit environment model to cope with the requirements of mobility. The information, generated with the help of an environment model and available data, can be represented in different flavors, such as an occupancy grid map, a relative position, or a set of objects or entities that fulfill some certain conditions.

In the related work, an environment model is rarely understood as a powerful filter and fusion mechanism for the heterogeneous and distributed sensor system, providing a well-defined and stable view of the physical surroundings. Additionally, as a major difference, the presented work does not consider cooperative functions of different systems based on an easy and efficient exchange of environment knowledge. In this paper we strive for a more general approach, which will support sensor interpretation, serve multiple applications, and also allows easily to exchange environment information between different mobile or immobile entities. This will be crucial for generating large scale and global environment perception. In contrast to our approach, other solutions are to a large extent rather application-specific and just tailored to a single dedicated control task. Furthermore, in most cases the environment models interpreting information coming from (local) sensors are intimately integrated into the application.

Before describing our approach and the problems we have to deal with, we will give detailed overview on related research areas within the next section and finish with a conclusion and an outlook on future work.

II. Related Work

There is a huge amount of proposals to model the environment from various areas in mobile robotics. An early overview on state-of-the-art world models is given in [2] but as stated by the authors themselves, most of these models are designed for specific (robotic) applications only. Examples are "Constructive Solid Geometry" [3], a geometric world model based on primitive geometrical objects, occupancy grid maps [4], octrees [4], and many more. Nearly all of these examples for models are also directly linked to the available sensor equipment and the respective data formats. In contrast to the pure geometrical representations, there are also application specific models, like in [5] that shows an example for trajectory-planning of car like vehicles, which require an abstract 2D model of the environment.

Recently there is a shift to more general and complex environment models, which results from the fact that also the robotic applications and the environments itself become more and more complex. Models should serve different purposes like surveillance, decision making, trajectory planning, obstacle avoidance, etc. Also, the authors of [6] tackle the problem of specified and customized world models. They present a system that incorporates a 3D occupancy grid map, other abstractions with more than just spatial semantics are not considered (but it allows to adapt update rates and accuracy to serve different needs). A more general approach, with more information about the environment and the objects within, is presented by Hsiao [7]. It bases on the Open Dynamics Engine (ODE) [8], a high performance library for simulating rigid body dynamics. Next to the geometric description of the environment, it also allows to label included 3D objects with additional information, such as mass, velocity, color, etc.

The idea that an environment model can be constructed from multiple sources is presented in [9]. It describes a vehicle's road environment conceptually as an (very specialized) object-oriented model. It uses a-priori information and information obtained from on-board sensors or through car2car communication. This means that, for example, the awareness of another car in front can be obtained from the local sensor system or from the communicated position of the front car itself. It makes no difference for the application, because all information is taken from the model representation only. This type of modeling is ideal for situation assessment, because all required sensor data of the environment is already translated into a simplified data-structure. This type of modeling is specialized on road environments, which restricts the type of entities (vehicles, pedestrians, and traffic signs), their representations (2D points), and also the application. Nevertheless, it shows that the application can be separated from the sensor interpretation and work only with information derived from a general model.

We discussed this separation of concerns firstly in [10] and presented a step by step approach of extracting functionality from an application that deals with environmental perception in [11]. Furthermore, we demonstrated the applicability of our concept in [12], see also our YouTube-channel [13]. Comparable to our approach, which is presented in the next section, Belkin and Kuwertz discuss their concept for a holistic environment model in [14] and [15]. It bases also on a scene-graph, representing the spatial relations between different geometrical entities/concepts. But in contrast to their solution, we do not believe that it is desirable in general to build a global environment model. Instead, every application should be able to generate its own local environment model for its own purposes (based on the available data within a smart environment) and abstract any kind of information from it. We had developed an architecture, based on a distributed database, which allows each entity to construct and update its own local environment model. Furthermore, all required data whether it is actual sensor data (using our Cassandra_ROS package [16]) or metadata describing sensors, actuators, or other objects of the environment, can be stored with arbitrary complexity as required.

Models based on logic predicates can be seen as another type of environment modeling, which also separates sensor interpretation and the use of the derived information. In a first step, all data has to be transformed into logic predicates, which subsequently can be queried easily according to various aspects. This approach is mainly used to determine complex action sequences as well as to describe complex situations and formally bases on the situation calculus [17]. Examples are "alGOl in LOGic" better known as GOLOG [18] with its specialized dialects, such as Con(current)GOLOG [19], which includes concurrency and the influence of external events. KnowRob [20] is also a knowledge processing system already integrated in the Robot Operating System (ROS) [21]. Logic-based approaches require huge knowledge bases and are slow in response, but due to the used concepts of predicates and rules, queries and situations can be expressed very simple (mostly in Prolog-syntax).

III. Environment and Views

It is a challenging task for a system to interpret varying and heterogeneous sets of networked remote sensors and also information obtained from other systems. It has to have the ability to "understand" the environment. On the highest level all information about the environment is captured in an environment model. The concept of an environment model can be described as a simplified co-simulation of the surrounding, which is continuously updated by local and remote sensors. It consists of elements and relations that represent all relevant contexts for perception and control.

Similar to the concept described in [14], [15], a simple scene graph may be sufficient enough, if only spatial information (e. g., distances, geometries, relative positions, etc.) is required. More sophisticated scenarios like cooperating cars require next to a geometric representation also data about masses, friction, velocities and forces. This type of data can be easily incorporated into the model by extending the prior scene graph with capabilities of the ODE (equal to the approach of by Hsiao [7]). The model as we use it for a robotic applications are even more complex, including dedicated information about the surroundings, its inhabitants (other robots and sensors), their capabilities, as well as on different objects and tools. To incorporate also such kind of information, we apply the OpenRAVE [22] (Open Robotics Automation Virtual Environment), an environment for testing, developing, and deploying real-world robotics applications.

As depicted in Figure 2, the environment model fulfills three tasks. First of all, sensor data is interpreted and filtered using its information. Secondly, it stores a history of states of the environment and the present state that is continuously updated by sensor data. Thirdly, the model provides access to specific aspects of the environment. We call the respective information, depending on the application requirements, a "view". Figure 2 illustrates these relations.

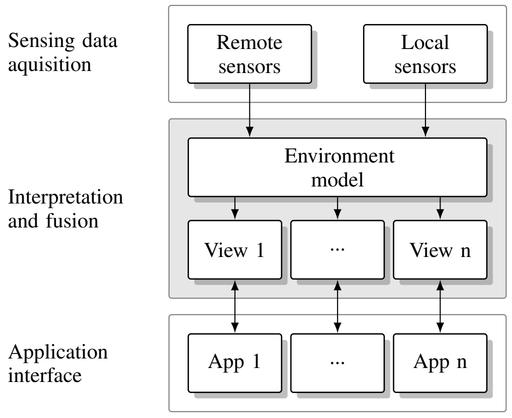

Fig. 2: View generation related to application requirements and based on a common environment representation.

A view is a well-defined and specialized abstraction of the external environment, while the environment model is a general representation and will serve multiple applications. This separation of concerns will ease the design of the application and support adaptation, because all the assumptions about the environment are made explicit in the description and can be changed. A specialized representation may be partly based on a-priori knowledge like a map but it is updated by the information derived from the sensors to include elements that are dynamically perceived, for instance, other vehicles and obstacles. As already mentioned, a 3D geometric environment model will comprise a coordinate system, geometric elements, the relations between them and the own position of the respective entity. If other elements were classified as mobile, their positions and relations will continuously be updated by sensor information.

Figure 3 shows the dataflow of a simple robotic scenario. In this example, a mobile platform operates in the aisle of our laboratory. It is equipped with a Kinect sensor and has access to virtual remote sensors that provide the static geometry of the building. It should be noted that the correct interpretation of sensor data from the Kinect already needs the environment model, i. e., the description about the position and the field of view. If we enrich the geometric model by the description of motion, we will be able to observe trajectories and use the environment model to detect and reason about jams and collisions. However, for our example we assume two functions, collision avoidance and short range path planning. Both applications utilize different aspects of the 3D environment model obtained from the view generator. The path planning algorithm receives an occupancy grid map of the vicinity. The collision avoidance algorithm does not need the sophisticated representation of the general environment model. The view defines a needed subset of the environment model, describing this as metadata to interpret sensor data. The view thus provides just the information needed by the application functions. A further important benefit of a view is that it shields the application from the details of the sensor system. In the example, the view would provide the unit "distance" independently of what sensors are used to derive this information. As indicated earlier, a varying set of sensors can participate in generating a view, without the need that the application must be aware of this.

IV. The Problem of Arbitrary Views

As shortly described in Section II, our previous work is based on developing an infrastructure for distributed environment modeling. According to the required view complexity, different models can be applied. All information about the surrounding are obtained from the environment model. But currently we have to apply specialized functions and filters to create views, specialized in terms of the type of environment model. Values such as the distance between two objects, their velocities, colors, or their relative positions can be easily extracted from the environment, while the generation of an occupancy grid map might be more difficult. And the definition of functions can vary according to the applied scene graph, physics simulation, etc.

It is easy to see that this solution is less satisfactory because complex queries or the definition of situations (based on the scene graph) have to be defined within the program code. Although, we started out with the undertaking of reducing source code complexity. Furthermore, calling such functions from the program code does not simplify the programmation and it does not meet the requirements of a dynamic access and changing demands. Queries and situations are hard coded and we have to parameterize our code to include dynamics. As also stated in [14], it is essential to have some kind of symbolic representation of the environment to solve these problems. Currently there is no system or solution, which can convert geometric models into logic-based and vice versa.

If we interpret environment models as an implicit and dynamically changing knowledge base, it should also be possible to query it in the same way as we query databases. In contrast to logic-based systems, we do not want to query the environment with predicates and rules, instead we want to query it with simple SELECT-queries, similar to ordinary SQL. In fact, querying with SQL or with Prolog is quite the same, since SQL is already an implementation of the Relational Calculus (form of Predicate Logic). While SQL is primarily intended to derive facts and relations (as we want to use it), Prolog is primarily a rules and inference engine. The result of an SQL-query, as we intend it, can be a single distance between two objects, a list of object having a special property, a specialized map of the surrounding, containing only objects bigger than a certain value, or a simplified model of the environment model itself. Everything can be put into SELECT statement using the WHERE clause to define additional conditions. Furthermore, it allows also to define situations, which occur, if the SELECT query returns a non-empty result. Therefore, we are currently developing a new kind of programming semantics/language that should allow to mix SELECT statements, running on our environment models and normal programming languages.

V. Conclusion and Future Work

As stated in [2], an environment model is the key component of any intelligent system, which must be able to describe and represent the environment as well as incorporated entities

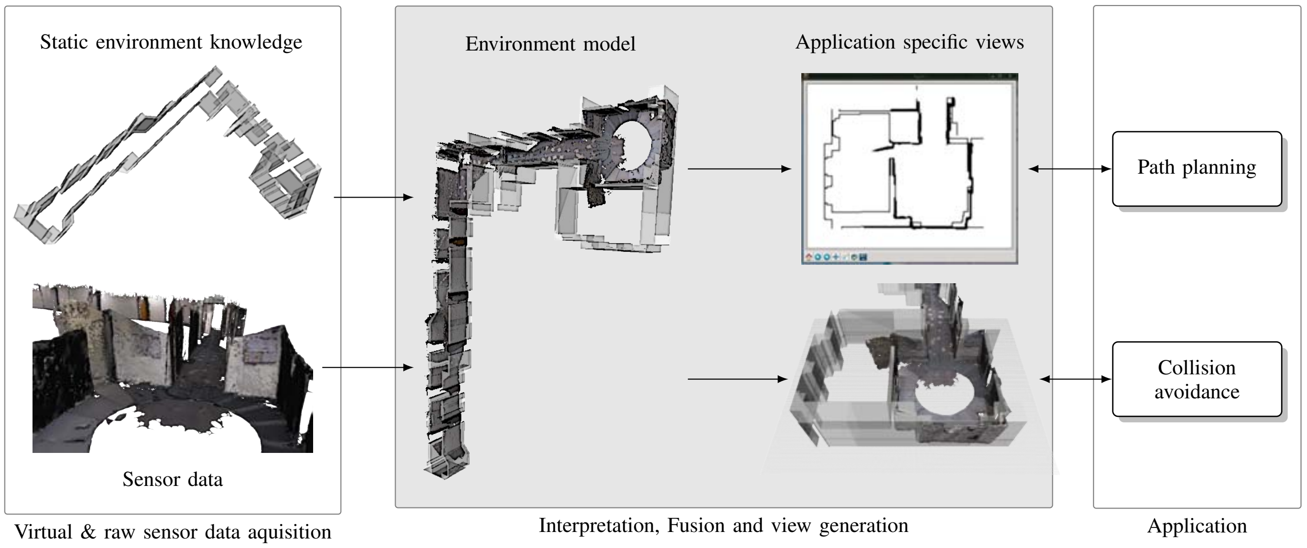
Static environment knowledge
Sensor data
Virtual & raw sensor data aquisition
Environment model
Application specific views
Interpretation, Fusion and view generation
Path planning
Collision avoidance
Application

Fig. 3: Composing an environment representation from static & dynamic data and generate of application specific views from it.

in an adequate structure and complexity. We presented our general approach of building environment models and the concept of deducing views from it. To be able to cope with the complexity and diversity of application specific requirements, we will have to develop also new programming concepts and paradigms. Or at least combine well known concepts in a new ways, which we will do by applying SQL on environment models.

Acknowledgment

This work is funded and supported by the German Ministry of Education and research within the project ViERforES-II (grant no. 01IM10002B) and by the EU FP7-ICT program under the contract number 288195 “Kernel-based ARchitecture for safetY-critical cONtrol” (KARYON).

References

[1] S. Zug, M. Schulze, A. Dietrich, and J. Kaiser, “Programming abstractions and middleware for building control systems as networks of smart sensors and actuators,” in *Proceedings of Emerging Technologies in Factory Automation (ETFA ’10)*, Bilbao, Spain, 9 2010.

[2] E. Angelopoulou, T.-H. Hong, and A. Y. Wu, “World model representations for mobile robots,” in *Proc. of the Intelligent Vehicles Symposium*. IEEE, 1992, pp. 293–297.

[3] A. A. G. Requicha and R. B. Tilove, “Mathematical foundations of constructive solid geometry: General topology of closed regular sets,” Production Automation Project, Univ. Rochester, Tech. Rep., 1978.

[4] S. Thrun, “Learning occupancy grid maps with forward sensor models,” *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.

[5] T. Fraichard, “Smooth trajectory planning for a car in a structured world,” in *Proc. of the International Conference Robotics and Automation*. IEEE, 1991, pp. 318–323.

[6] R. K. Harle and A. Hopper, “Dynamic world models from ray-tracing,” in *Proc. of the 2. Annual Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2004, pp. 55–64.

[7] K. Hsiao, N. Mavridis, and D. Roy, “Coupling perception and simulation: Steps towards conversational robotics,” in *Proc. to the International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. IEEE/RSJ, 2003, pp. 928–933.

[8] R. Smith, “Open Dynamics Engine project website,” http://www.ode.org, [(online), as at: 29.03.2011].

[9] A. Furda and L. Vlacic, “An object-oriented design of a world model for autonomous city vehicles,” in *Proc. of the Intelligent Vehicles Symposium (IV)*. IEEE, 2010.

[10] A. Dietrich, S. Zug, and J. Kaiser, “Model based Decoupling of Perception and Processing,” in *ERCIM/EWICS/Cyberphysical Systems Workshop, Resilient Systems, Robotics, Systems-of-Systems Challenges in Design, Validation & Verification and Certification*, Naples, Italy, 9 2011.

[11] ——, “Towards Artificial Perception,” in *SAFECOMP 2012 Workshops*, F. Ortmeier and P. Daniel, Eds. Springer-Verlag Berlin, 2012, pp. 466–476.

[12] ——, “Geometric Environment Modeling System,” in *IFAC Conference on Manufacturing Modelling, Management and Control*, Saint Petersburg, 6 2013, pp. 1445–1450.

[13] A. Dietrich, “Youtube-channel: Ivs magdeburg,” http://www.youtube.com/ivsmagdeburg, [(online), as at: 29.09.2013].

[14] A. Belkin, A. Kuwertz, Y. Fischer, and J. Beyerer, “World modeling for autonomous systems,” *Innovative Information Systems Modelling Techniques*, vol. 1, pp. 135–158, 2012.

[15] A. Kuwertz, “Towards adaptive open-world modeling,” Vision and Fusion Laboratory, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Tech. Rep., 2012.

[16] A. Dietrich and S. Zug, “Cassandra_ros project website,” http://www.ros.org/wiki/cassandra_ros, [(online), as at: 29.09.2013].

[17] J. McCarthy, “Situations, Actions, and Causal Laws,” Stanford University Artificial Intelligence Project, Tech. Rep., 1963.

[18] H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl, “Golog: A logic programming language for dynamic domains,” *The Journal of Logic Programming*, vol. 19, no. 1-3, pp. 59–83, 1994.

[19] G. De Giacomo, Y. Lespérance, and H. Levesque, “Congolog, a concurrent programming language based on the situation calculus,” *Artificial Intelligence*, vol. 121, no. 1, pp. 109–169, 2000.

[20] M. Tenorth and M. Beetz, “Knowrob – knowledge processing for autonomous personal robots,” in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 4261–4266.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[22] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, 8 2010.