 DOI: 10.1109/IROS.2014.6942700

Distributed Management and Representation of Data and Context in Robotic Applications*

André Dietrich[1], Sebastian Zug[1], Siba Mohammad[2], and Jörg Kaiser[1]

***Abstract*— The traditional, isolated data handling in sensor-actuator systems does not fulfill the requirements of robots that need to interact with their smart environment. Consequently, we have to develop new mechanisms for adaptive data and context handling.**

We firstly investigate what types of data are present within smart environments and how they can be classified and organized. Only if the available data can be structured, it can be queried and thus put into context. This is important because the variety of data and possible interpretations is tremendous, ranging from measurement values, sensor and robot descriptions/states/commands, to environmental data, such as positions, maps, spatial relations, etc. To cope with this diversity, we developed a solution capable of storing and accessing data within a distributed environment by providing additional context information. Furthermore, we describe how this information can be assembled in a task-oriented manner. This enables robots to dynamically generate environmental abstractions by using data from different sources and also enables them to incorporate external sensor measurements.

I. INTRODUCTION

If we think of future smart environments (a good overview is given in [1]), whether in industrial manufacturing, building automation, logistic processes, or health-care scenarios, we think of various different smart entities, capable of functioning in dynamic and changing environments, interacting with each other and solving tasks in cooperation. These entities are capable of sharing their knowledge, experiences, and environmental perception. New and evolving technologies and paradigms like the "Internet of Things" (cf. [2]), "Cyber-Physical Systems" (cf. [3]), or "Pervasive/Ubiquitous Computing" (cf. [4]) adopt these ideas and intend a flexible communication between different intelligent components. However, simply transferring data is by far not enough. It is rather the basis for flexible cooperations, or from another perspective just the "tip of the iceberg" of what needs to be done. In the end, there is a tremendous difference between a value and a meaning. What does a change of a single distance measurement stand for? Thus, meaning can only be derived by transforming, interpreting, and reinterpreting data according to the current context. Whereby, a context can be defined by everything that is relevant to fulfill a certain task, like other sensor measurements, location and infrastructure, safety and security requirements, time, physical conditions, etc. (cf. [5]). This opens up two rarely asked questions:

1) *How can data be stored and accessed in conjunction with context?*
2) *How can data, gathered from various sources, and context be dynamically put together, to deduce any kind of application specific information?*

We construe a smart environment with appearing and disappearing entities, with overlapping workspaces, changing tasks, etc., as some kind of distributed mind that is continuously producing new data, information, and knowledge. Systems in such environments will have to continuously adapt to these changes, and therefore will require holistic access to this distributed mind. Thus, there is a myriad of data (measurements, commands, states) and descriptions (meta-data) resident within smart and distributed environments, which is somehow related (spatial, temporal, etc.), but it is not organized, not directly accessible, and cannot be queried (other problems are discussed in [6]).

Imagine a robotic platform that enters a (manufacturing) hall for the first time to deliver some cargo to a certain location. This platform would establish a connection to the smart entities of the environment and simply request all needed information, like a map of the hall with a sufficient level of detail, including restricted areas, present robotic systems, available sensors, human positions, etc. According to the determined route, it would then just filter out irrelevant information and possibly try to get access to external sensors along its way. The flexible usage of external sensor allows to extend the local view on the environment. The robot is able to navigate more efficiently, by preventing sudden and unpredictable dangerous situations and can increase its traveling speeds.

The common benefits of such a flexible exchange and integration of information are obvious. It provides a higher coverage and higher degree of fault tolerance compared to individual perception. At the same time, systems can interact more efficiently and coordinate their behavior. It should be noted that such future cooperations will happen between previously unknown systems and therefore have to tolerate dynamic integration and segregation of heterogeneous entities. Unlike today where cooperation or even the integration of external sensor measurements is only possible if, and only if, it was previously intended by the programmer/system-designer.

*This work is (partly) funded 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).

[1]A. Dietrich, S. Zug and J. Kaiser are with the Department of Distributed Systems, at the Otto-von-Guericke-Universität Magdeburg in Germany {dietrich,zug,kaiser}@ivs.cs.uni-magdeburg.de

[2]S. Mohammad is with the Database and Information Systems Group at the Otto-von-Guericke-Universität Magdeburg in Germany siba.mohammad@iti.cs.uni-magdeburg.de

Structure and Overview

To tackle the above mentioned problems more efficiently, we subdivided our approach into three conceptual parts. Starting bottom up, we first identify what data is available in such environments and how it can be classified. Based on this preliminary investigation, we present a solution to *question 1*, which allows us to organize this data (with its spatial and temporal contexts) so that it can be dynamically accessed and queried. The third part is based mainly on our previous research efforts and describes how this data, coming from different sources, can be combined freely and how to deduce any kind of application specific information (*question 2*). Additionally, we demonstrate the applicability of our approach by replaying the introductory example on the floors of our faculty building. To illustrate important aspects of this paper in a more convenient way, we uploaded additional video material to our YouTube-channel at `htttp://www.youtube.com/ivsmagdeburg` and denote its appearance in some figures with "Fig.$^{\text{Ani.}}$".

Because the presented concept touches different research ideas, we present a comparison to the related work at the end of this paper, followed by the conclusion.

II. DIFFERENT FLAVOURS OF DATA

There is a tremendous difference between the often erroneously mixed terms knowledge, information, and data, which is caricatured in Fig. 1. As described in [7] (in comparison to different research areas), data is mostly assumed as simple, discrete, and isolated facts without any explicit interpretation (e. g., the positions and trajectories of robots and humans). Combining data within a structure or putting it into a certain context generates information (e. g., combining the trajectories and current positions within a map). Giving information a meaning by interpreting it, produces knowledge (e. g., the prediction of a collision, due to intersecting trajectories). While intelligence comes into play by choosing alternatives or by deciding on strategies based on knowledge (e. g., preventing the collision by choosing between a full-stop, a change of speed, by recalculating the trajectory, etc.).

In the context of smart environments, we can subdivide data into the following four general categories (cf. Fig. 1):

1) Metadata: it can be translated as "data about data" and refers to the static descriptions of data-formats (e. g., TEDS[3], ROS-msg[4]), entities (e. g., MOSAIC[5], SensorML[6]), or complete systems (e. g., AutomationML[7]) and interfaces. It allows to access external systems, describes their configurations, and enables the decoding of raw data. Literally, metadata is used to describe every sensory system of a robot, the robot itself, and external sensors in detail as well as how to access these systems and to decode their message streams.
2) Raw Data: it is the amount of directly measurable physical quantities that change over time, such as distance (laser scans), temperature, light (camera frames). This group furthermore comprises status messages gathered sensors or actuators.
3) Virtual Data: it can be described as indirect measurements that are derived from raw data by applying physical or mathematical laws (cf. [9]). This type is required where physical modalities are not directly measurable, such as the temperature within a combustion engine, or simply to reduce the amount of raw data and to deduce more expressive values, like a maximum temperature or the average income.
4) Abstract Data: it is higher-level and can be generated by abstracting from raw and virtual data. A wall for example or a map, which are abstracted from multiple laser scans, a detected human within an individual camera frame or deducing the human's trajectory by analyzing multiple frames.

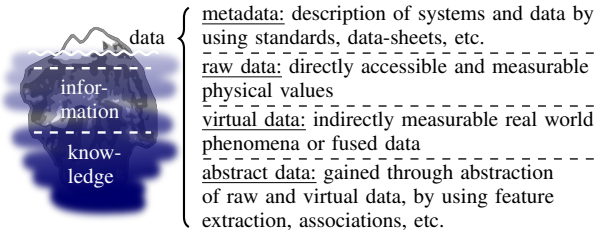


Fig. 1: Differences of data as well as information and knowledge in smart environments.

Furthermore, raw and virtual data are useful in two ways: They have a real-time value as well as historical value. While access to real-time data is necessary for most systems that directly manipulate or interact with their environment, or bodiless applications for controlling or surveillance; a wide range of applications also require further access to historical data. Examples are the mining of sensor logs to detect unusual patterns, analysis of historical trends, post-mortem analysis of particular events, or simply for later data abstraction (mapping). Thus, the archival storage of past sensor data is becoming more important.

We therefore tried to create a minimalist solution that covers all the above mentioned types of data as well as their specific real-time/historical value and spatial and temporal context. This requires the combination of two diverse areas: Distributed robotic applications (as a part of smart environments) with distributed or nowadays cloud-based database systems. This conceptual approach is presented within the following section.

[3]"Transducer Electronic Data Sheet", which is part of the smart transducer interface standards set IEEE 1451 [8].

[4]A messages description language used to define various message formats for the ROS publish/subscribe system: `www.ros.org/wiki/msg`

[5]The "fraMework for fault-tOlerant Sensor dAta processIng in dynamiC environments" provides several XML descriptions for accessing sensors as well as actuators: `www.code.google.com/p/mosaicframework`

[6]"Sensor Model Language" offers models and encodings to describe sensors and measurement processes: `www.ogcnetwork.net/SensorML`

[7]Automation Markup Language is a description format for storing and exchanging plant engineering information. `www.automationml.org`

III. DATA ORGANIZATION AND STORAGE

The key idea is to use databases as a kind of holistic storage, where every entity hosts a local database instance to memorize any kind of data that may be usable to carry out a certain task. The usage of distributed databases should therefore allow the entity to seamlessly access and query data from all other connected entities. We decided to apply Cassandra as our prior database system in combination with ROS (Robot Operating System [10]) and its communication infrastructure and message description formats.

Within this section, we describe how data is stored and organized by keeping spatial and temporal relations. A simplified diagram revealing the basic database structure and the organization of data is depicted in Fig. 2. But firstly, we discuss some benefits of Cassandra as well as the reasons for choosing this particular database system.

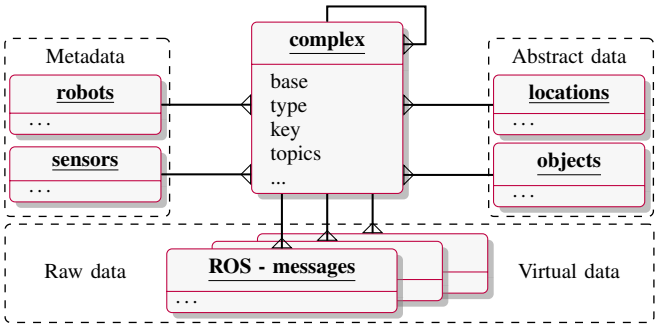


Fig. 2: Organization of data with column-family complex as the global link.

A. Why Cassandra?

Although Cassandra was initially developed as a NoSQL (Not only SQL (Structured Query Language)) database for Facebook [11] and received only little attention in the robotics community (cf. [12]), its concept as well as some of its features make it an ideal storage system for distributed applications (especially for smart environments).

Cassandra provides a distributed key-value store. Keys map to rows, which can contain a multitude of columns (values), while rows by themselves are stored in column-families (tables). See also Fig. 2 to get a better impression about the data structure in Cassandra. Further columns can be added dynamically or removed at any time, so that different rows can store different types and different amounts of values. That means that the structure of the database can change over time and adapt to varying requirements, unlike a classical relational database system. As revealed in [13] with a performance evaluation, it also seems to provide an ideal storage for sensor data. In [14] we describe a generic ROS-Cassandra interface and present an evaluation that compares our solution with the ROS MongoDB[8] implementation and the standard ROS data container rosbag[9]. This comparison reveals that Cassandra memory consumption is very close to rosbag, by keeping the ability for fast and complex querying by using CQL (Cassandra Query Language). Additionally, it showed the best overall performance.

Cassandra instances can be grouped into clusters and keyspaces, allowing an entity to host and update its own database instance. By querying local data, it also queries data from other connected entities. Different strategies for replication between instances provide a high availability of data with no single point of failure. An entity can thus leave the cluster while its data remains on other nodes. Every value is marked with a timestamp, which allows to define TTLs (Time to Live), so that data can be forgotten after a certain period of time. This usage of timestamps enables eventual consistency (see also [15]), a weaker consistency level than strict or immediate consistency, which are commonly used in relational databases, but more appropriate for distributed systems. That means that due to the distribution of data and the availability of nodes, it is guaranteed to receive a valid value but maybe not the most recent. Tunable consistency level enables application dependent refinements.

B. Metadata

There are currently two types of system described with metadata: Sensors and robots. Whereas the description of sensor and actuator messages is left out to ROS and its message description language. Metadata for robots and sensors are stored within two different column-families.

1) Sensors: We used the OpenRAVE[10] XML-description format for sensors, capable of defining various kinds of sensors (such as laser scanners, odometry, etc.). This description format was combined with the MOSAIC sensor description capabilities, which allows the definition of more realistic sensors, by including various fault models (see also [16]).

2) Robots: To characterize different robots in terms of a kinematic and dynamic description, a visual representation, and a collision model, we apply common formats such as URDF (Unified Robot Description Format [17]) and COLLADA (COLLAborative Design Activity [18]).

C. Raw & Virtual Data

Our generic cassandra_ros[11] interface, introduced in [14], allows to store, query, and request historic ROS-messages (e. g., laser scans, camera streams, etc.) of any kind and source. Data can be stored either in a binary format (for fast access), in a ROS similar manner (slower due to conversation of messages, but it allows querying and analyzing data more conveniently with Cassandra's CQL-capabilities), or in other formats, like yaml or string (having their specific advantages and disadvantages). In contrast to other data, these types of data are stored in multiple column-families, one topic per column-family. This is necessary to be able to cope with the large amount of produced data, its diversity, and to reduce replication efforts. As mentioned in Sec. III-A, every stored message is automatically labeled with a timestamp by Cassadra, which allows it to keep temporal relations.

[8] `wiki.ros.org/warehousewg`, [9] `wiki.ros.org/rosbag`

[10] OPEN Robotics Automation Virtual Environment: `openrave.org`

[11] Project website: `www.ros.org/wiki/cassandra_ros`

D. Abstract Data

Actually, we identified two basic types of abstract data that should be sufficient to allow smart entities to sustain even in complex 3D environments. These are locations and objects. Therefore, both of these data types are stored within their own column-family (cf. Fig. 2).

In our case, a location always represents a certain area, which is in general static and does not change over time. It can be represented by any kind of 2D map or any kind of 3D structure. Whereas objects can define any "thing" in the environment and come up with higher dynamics than locations, ranging from rarely moved objects such as tables or wardrobes to objects with frequently changing positions (e. g., mugs, chairs, etc.). As depicted in Lis. 1, the same object or location can be represented in different formats and with different levels of precision. A mug for example can be defined as a detailed 3D model, such as vrml (Virtual Reality Modeling Language), stl (Surface Tessellation Language), etc., or as point cloud data (pcd). The value for precision is currently a subjective chosen value. An application shall later be enabled to decide upon the required amount of precision and the appropriate format, simply by parsing column names.

Listing 1: Extract of column-family objects, with row-keys (bold), columns (blue), and values.

```
1   objects : {  ...
115   1sx34s : {
116     comment : "mug ... ",
117     stl_85 : binay-data,
118     pcd_85 : ascii-data },
232   6yfr48 : {
233     wrl_90 : xml-data,
234     comment : "standard phd student table",
235     ... },
```

E. Complex

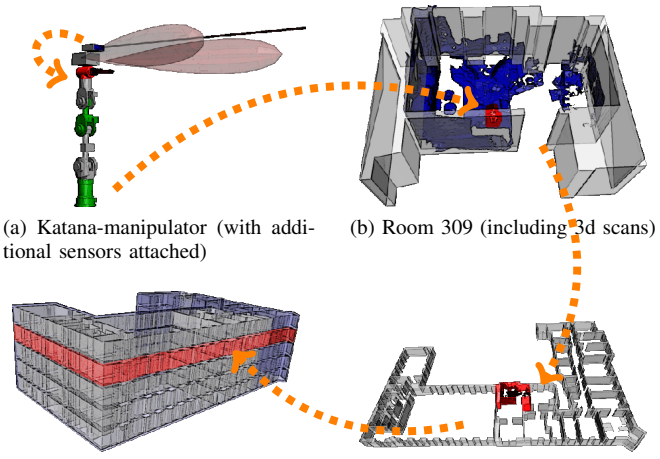
So far we have only described how data is stored, but not how this data is interconnected. To put all entities into a global context, we use a further data structure, which we call "complex". This column-family combines all previous data (cf. Fig. 2) and organizes it within a hierarchical structure.

As listed in the code example below, a complex entry always points to a certain entity of a certain type (i. e., robot, location, sensor, object) with a certain position, identifier, etc., whereas metadata or objects are used to describe a certain class of entities. For example, there can be two mugs placed within an environment that are derived from the same (abstract-) object. A complex entry can further be used as a bodiless placeholder (to simplify some structures) by pointing onto another complex entry. Next to a position (with its specific uncertainty) and orientation relative to a base, complex entities are further labeled with communication specific information. Since we are using ROS, we store additional information about masters, topics, and services. Thus, by assigning communication details to complex entities, it is also possible to access to any kind of raw&virtual real-time data (the correct interpretation is left out to the next section) as well as to historic data, which is assigned to this topic (cf. Sec. III-C).

Listing 2: Showing the complex entry `katana_62x` of type robot, whose description is stored in row `katana_2012` in column-family robots, whereby position and orientation are defined relative to the complex base entry `room_309`.

```
1   complex : { ...
255   katana_62x : {
256     key : "katana_2012",
257     type : "robots",
258     base : "room_309",
259     position : [2.3, 3.2, 0.0],
260     quaternion : [0.92, 0.0, 0.0, 32.2],
261     covariance : [0.04, 0.2, 0.01, 0.14, 0...],
262     ros-master : "http://moritz:11311",
263     topics : ... },
517   room_309 : {
518     key : "room_309",
519     type : "locations",
520     base : "floor_4", ... },
```

The order of entities, in our case, is defined by the their positions, which are commonly defined as relative to a basis, like a robot's position relative to a room, sensor positions and orientations mounted to a robot, or a room's position within a floor of a building, etc. We do the same, by assigning a base to every complex entry. All positions are then defined relative to that base. Knowing the id of a single entity (bootstrapping), like the katana's id in Fig. 3a, allows to query in two directions: Downwards, by identifying entities whose base the katana is, and upwards, by querying for the katana's base and for entities that are related to the same base (cf. Fig. 3b).



(a) Katana-manipulator (with additional sensors attached)

(b) Room 309 (including 3d scans)

(c) Building (global base)

(d) 4th floor (assembly of multiple rooms)

Fig.[Ani.12] 3: Hierarchy of entities, starting from the sensors whose positions are relative to the robot, while the robot is located within room 309, which is part of the fourth floor in building 29.

To sum up, it can be said that the organization of data, as we propose it, constructs a large and distributed scene graph, where every entity updates its own local Cassandra instance with positions, identified objects, sensor measurements, etc. But it is exactly what we required to solve *question 1*.

[12]Animation: `www.youtube.com/watch?v=kvoC5yxdzsw`

IV. DATA INTERPRETATION

Within this section, we present a solution to *question 2*, where we try to make sense of all of these different types of data and relations by transferring them into a useful representation, which we call an "environment model".

A. The Concept of an Environment Model

Environment models can be considered as an adoption of "mental models" (cf. [19]), which are widely used in cognitive science to (partially) explain how humans perceive, reason, assess, learn, and make decisions for a variety of domains. Additionally, these models are essential for integrating new information correctly. In the simplest way, they can be interpreted as "mental simulations" of the real situations or problems, with reduced complexity according to reality. Quantitative relations are mapped onto qualitative, and thus allow to store and handle elements of the world within the working memory (cf. [20]).

The term environment model is used, because we only consider information and data that might be relevant for a system (robot), to sustain in a spatial environment. While mental models are used in a much broader sense, including also sociocultural standards or knowledge and reasoning about complex action sequences. We firstly described the idea of using environment models for environmental perception and modeling in [21] and demonstrated its applicability in [22]. As depicted in Fig. 4, all data is put into a "co-simulation" of the surrounding. We currently apply OpenRAVE[10] as our prior simulation environment. It is already integrated into ROS and open source, which allows to develop own extensions[13].

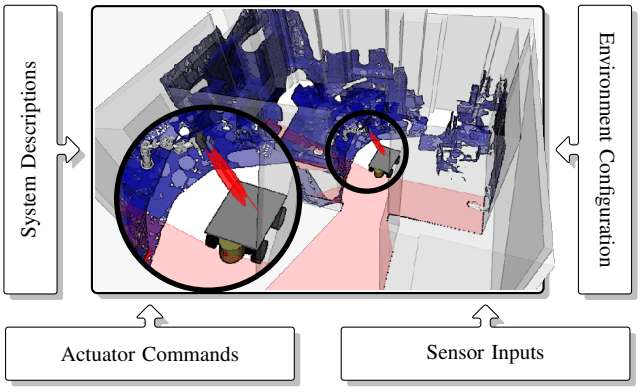


Fig. 4: Schematic structure of an environment model as an assembly of available and selected data, such as robots and sensors (metadata), real-time values including sensor measurements (red), current robot positions and configurations (raw data), as well as (abstract data) geometries of the room and depth scans (blue).

The bases for building such an environment co-simulation are the data and the relations of the last recent environmental configuration obtained from the distributed database. However, by associating the reconstructed scene and its entities with real-time values through subscribing for topics, it is possible to repeat every action and measurement within the virtual world. It allows to integrate external sensors and actuators and to interpret their in- and outputs as well as their effect on the configuration of the environment. It can be further used to validate sensor measurements by comparing their values with measurements obtained from their virtual counterparts.

All information required for applications can then be taken directly from the model, such as relative positions and distances, available sensors/robots and their location or monitoring/operational area, etc., which can be considered as abstractions of an environment model.

B. The Concept of Abstracting Views

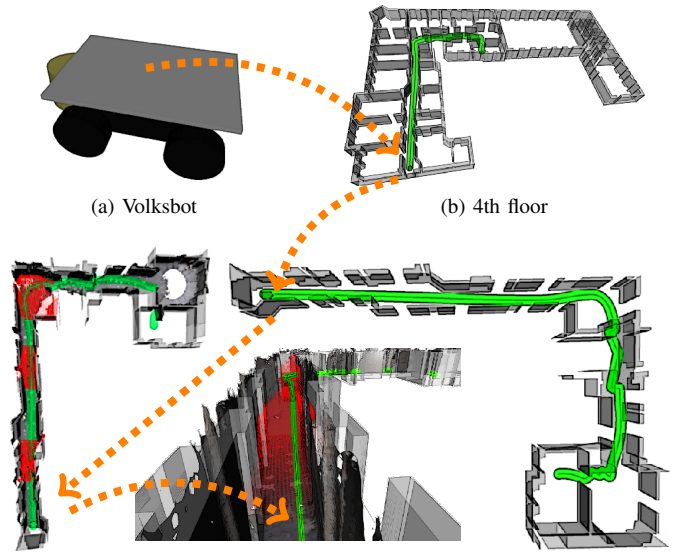
A view is a well-defined and application specific abstraction of the external environment, while the environment model is the most general representation and serves multiple applications. A view can be anything such as complex 3D models used for grasp planning or to switch perspectives (to be able to follow human orders [23]), 3D or 2D maps used for navigation and localization, or simple state vectors including distances, velocities, or even sets of objects and sensors that fulfill certain properties. Deducing such application specific information from the environment model is a quite challenging task, due to the multitude of requirements and opportunities. In [24] we discussed this problem and proposed a solution, which interprets an environment model as an implicit and dynamically changing knowledge base that can be queried in the same way as a database. We therefore had developed a new scripting semantics/language, based on simple SELECT-statements, which is mainly inspired by SQL. It even allows to define complex situations (combining time and space), which occur if the SELECT query returns a non-empty result. This query language is called SELECTSCRIPT and it exists a prototype implementation for OpenRAVE, to get a first impression on the language and its concepts see the current project web-site: `http://pythonhosted.org/SelectScript_OpenRAVE`

C. Proof of Concept

Because the benefits of our approach are hard to measure and a sequence of queries would be meaningless, we present the results of those queries and describe the conceptual way of querying (for filtering and abstracting the generated environment model we applied our query language SELECTSCRIPT). Let us go back to the delivery scenario, with which we introduced this paper. We, therefore, modified the scenario according to our local surroundings, whereby the robot does not have to deliver cargo but simply to enter `room_309`. So we start with only a model of the robot, as it is depicted in Fig. 5a, knowing its relative position to the 4th floor. By querying our system for all `locations` whose base is `floor_4`, it is possible to reconstruct a simple model of entire floor, with the correctly placed robot. This model can be used to identify the position of the target room and its relative position to the robot. Additionally, this

[13]Our plugins: `code.google.com/p/eos-openrave-plugins`

model is used for an initial trajectory planning (see Fig. 5b), which allows to it remove all `locations` that were not touched by the calculated path afterwards (see Fig. 5d). The remaining `locations` are used as bases to query for all related sensors, robots, objects, abstracted data and historic raw measurements (like a point cloud, retrieved from an earlier Kinect scan). All of these entities are then placed within the correct spatial context into the model, compare with Fig. 5c.



(a) Volksbot (b) 4th floor

(c) Complex environment (d) Trajectory filtered locations

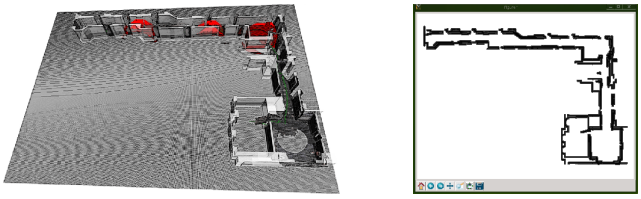
Fig.[Ani.14] 5: Query sequence: (a) starting from the robot model and its relative position; (b) adding location information about the 4th floor, which is used for trajectory planning; (d) the resulting trajectory is used as a filter to segregate information about not required locations; (c) integration of entities, such as external sensors (red) and Kinect scans.

The resulting model can be used afterwards in multiple ways. On the one side it allows to identify and retrieve all relevant information about the current operational area, about included objects, robots, sensors, and their current configurations. But on the other side, it can be further abstracted to generate application specific views. The entire environment model is far too complex and exaggerated for the purpose of navigation. Thus, a simple occupancy grid map might be the better choice. As depicted in the screen shots in Fig. 6, something like maps can be simply generated by applying different filters.

V. COMPARISON WITH RELATED WORK

Sharing data between entities is not a new idea, in contrast to its organization and structuring for smart environments. The outcome is that data gained from different sources can be reused, combined and transformed into more meaningful representations. Our notion of a meaningful representation of data is based on environment models (cf. Sec. IV-A). These models are used for two purposes, they are required for including external measurements, which requires knowledge

[14]Animation: `www.youtube.com/watch?v=Xt403wPCYD8`



(a) Environment model with filter (b) Occupancy grid map

Fig.[Ani.14] 6: Abstraction of maps from complex models by applying filter-functions.

about the type of sensor, relative positions, etc., but also to integrate newly made observations. In concrete terms, an autonomous robot should be able to look around the corner, by using external cameras to detect humans and other obstacles, but it should also be able to share this experience.

As the following overview reveals, there is some related research that implicitly presents solutions our initial *questions 2* and *1*. But it also reveals that the growing complexity of environment models and the ability for distribution are also an outcome of previous developments and future requirements.

A. Simple One-Purpose Models

The differences in tasks and environments in robotics applications lead to a diversity of world models. An early overview on the state-of-the-art given in [25] reveals that most of these models were designed for specific (robotic) applications only. Examples are "Constructive Solid Geometry" [26], a geometric world model based on primitive geometrical objects, occupancy grid maps [27], octrees [27], etc. All of these examples are directly linked to the available sensor information and its respective data formats. The authors of [28] present a more specialized 3D occupancy grid map, which can be customized in terms of update rates and accuracy, to serve different needs. Other abstractions with more than just spatial semantics are not considered.

B. Complex Multi-Purpose Models

Nowadays there is a shift to more general and complex models, resulting from the fact that robots and their environments themselves have become more complex. New developments should serve various purposes, like decision-making, trajectory planning, obstacle avoidance, etc.

A theoretical concept of such a multi-purpose environment model was discussed in [29], its architecture adopts the psychological principals for artificial perception and consciousness. The concept differentiates between two levels of consciousness and two models. The first level of consciousness simply describes basic and reflexive (hard-wired) behavior that reacts directly to sensor inputs. Whereas, the second level uses two types of abstractions, a self and a world model, generated from system-knowledge and sensory inputs. Higher level planning and predicting is cut off from all sensory inputs and uses only the models. Unfortunately, there is no further explanation of how such models look like or what aspects of the environment they describe.

In contrast to this, Hsiao and Roy presented an environment model in [30] and [23] that describes intersecting workspaces of robots and humans as a 3D physics simulation. In fact, this approach was a source of inspiration for our concept of co-simulating the environment. Their approach is formally based on ODE[15], a high performance library for simulating rigid body dynamics. It allows to describe an environment and the objects within in terms of shapes, masses, velocity, forces, and colors. These properties were used for enhanced predictions and it was demonstrated that the cooperation between humans and robots requires such a complex environmental representation. It enables a robot to change its own viewpoint and to interpret human commands, such as "Give me the blue screwdriver on my left!", correctly. But this environment model is built upon a fixed and centralized simulation that does not allow the dynamic integration of entities, and it does not consider specialized entities (external robots and sensors).

The idea that an environment model can be constructed from multiple sources is presented in [31]. It describes a vehicle's road environment conceptually as a (very specialized) object-oriented model. It uses a-priori information and information obtained from on-board sensors or from through car2car communication. That means that the awareness of another car in front can be obtained either from the local sensor system or from the communicated position of the front car itself. Types of entities are restricted to vehicles, pedestrians, and traffic signs, represented as 2D points on a lane. This type of modeling is ideal for situation assessment, because all required information is already translated into a simplified structure with some semantics. However, the environment of an autonomous robot is far more complex.

A concept of a complex environment model for autonomous systems, which reflects our notion, was presented in [32], [33]. It separates between sensor data, a world model, and knowledge. Sensor data is analyzed with the help of already existing knowledge, and the resulting information is passed to the world model. Knowledge is defined in terms of specific methods and algorithms for analyzing sensor data. The world model consists of objects (labeled with attributes), representing entities of interest. These objects are interconnected in a scene via relations. It remains unclear how data, knowledge, and the world model are stored or how scenes, including all details about the environment, are represented. But it reveals that there is a need for a symbolic abstractions/level to describe situations, similar to the examples within the next subsection.

C. Logic-Based Models

Models based on logic can are another type of environment modeling. Whereby, such systems already work on abstracted sensor data in terms of predicates and rules, which 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. In most cases it is based on the situation calculus [34]. An example is "alGOl in LOGic" better known as GOLOG [35] with its specialized dialects.

KnowRob [36] is another example of a knowledge processing system, based on Prolog and OWL (Web Ontology Language). It offers tools for the automated acquisition of concepts through observation and experience, which can be used for learning and reasoning. Perception modules therefore create 3D environment maps, track human motions, segment objects, and record robot activities, which means that all data is already abstracted by a fixed set of algorithms into a fixed set of concepts. Thus, sensor data is lost after this procedure and cannot be used for later analysis. Abstracted information is put into a knowledge base and can be easily queried afterwards. Next to its connection to RoboEarth (see next subsection) it also shows that relations can be directly extracted from a spatial model. For example frequently changing relations, such as "on", "in" or "below", are completely determined by the positions of these objects. As the authors argue themselves, storing an object's positions and all possible relations within a knowledge base would cause too much overhead, calculating the relations on demand is more elegant. We simply extend this consideration, if we claim that also all other information/relations can be extracted from a complex environment model, as discussed in Sec. IV.

D. Distribution

A distributed scene graph with uncertainty support was presented in [37]. It is intended to be used as a shared world model for robotic applications. This approach is specialized onto geometrical representations and spatial relations with support for semantic annotations. It does not allow to share further sensory data between entities, which could be used afterwards to identify or generate additional elements, that could be integrated into the scene graph.

RoboEarth [38] can be considered as a top-down approach, closely related to the questions of what data is required to support KnowRob. Whereby, we started bottom-up by querying what types of data are available and how they can be organized. RoboEarth is used to store and access the required (abstract) concepts, such as robots, objects (i. e., CAD models, point clouds, images), environments (i. e., maps and information on coordinate systems), and action recipes. This data is separated in two distinct database systems, one based on Apache Hadoop[16], for storing data hierarchically and a graph database for storing complex semantic relations between data. Thus, its main purpose lies in reasoning and in sharing knowledge between robots, but it does not allow to share sensor data and functionality.

The PEIS-Ecology (Physically Embedded Intelligent System) supports the idea of distributed robots (cf. [39]). Actors, sensors, and everyday objects are able to share a predefined set of functionalities dynamically, to serve tasks, they cannot fulfill on their own. All systems and functionalities are therefore described with metadata, which allows to determine their adequate combination or execution of functions.

[15]Open Dynamics Engine: `www.ode.org`

[16]Project website: `hadoop.apache.org`

A combination of heterogeneous and distributed robots was also presented in [40] for outdoor scenarios. It combined sensor information from different robots and used it for localization, mapping, and path planning. A similar approach was made by DAvinCi [41]. In contrast to the previous attempt, it was built on a cloud-based service[16]. A heterogeneous set of robots uploads its data that is afterwards combined for map building and segmentation. The main problem in both approaches is, that they imply that all heterogeneous robots are able to share a global 2D map, which requires that all robots are equipped with sensor systems at the same height, used for localization. Otherwise, maps[17], generated at different heights for the same location, differ too much. It is thus more appropriate to be able to share sensor data from the same location and to generate maps afterwards (cf. [42]).

VI. CONCLUSION

A holistic access to data in smart environments requires some kind of organization and the ability for interpretation. These facts are mostly neglected in related research efforts. Therefore, we started out by examining, what are relevant types of data in smart environments and how this data can be organized. Our solution is formally based on Cassandra, allowing every entity to store its data for its own purpose, but also to share it with interested entities. This allows it to extract any information afterwards, whereby we propose the usage of environment models and views.

REFERENCES

[1] D. Cook and S. Das, "How smart are our environments? An updated look at the state of the art," *Pervasive and Mobile Computing*, vol. 3, pp. 53–73, 2007.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, 2010.

[3] J. Shi, J. Wan, H. Yan *et al.*, "A Survey of Cyber-Physical Systems," in *Proc. of the Intl. Conf. on Wireless Communications and Signal Processing*, 2011.

[4] P. Bellavista, A. Corradi, M. Fanelli *et al.*, "A Survey of Context Data Distribution for Mobile Ubiquitous Systems," *ACM Computing Surveys*, vol. 45, pp. 1–49, 2013.

[5] A. Dey, D. Salber, and G. Abowd, "A Context-based Infrastructure for Smart Environments," GeorgiaTech, Tech. Rep., 1999.

[6] S. Remy and M. Blake, "Distributed Service-Oriented Robotics," *IEEE Internet Computing*, vol. 15, pp. 70–74, 2011.

[7] I. Tuomi, "Data is more than knowledge: Implications of the reversed knowledge hierarchy for knowledge management and organizational memory," in *Proc. of the 32nd Annual Hawaii Intl. Conf. on System Sciences (HICSS-32)*, 1999.

[8] *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators (IEEE 1451.0)*, IEEE Std., 2007.

[9] S. Kabadayi, A. Pridgen, and C. Julien, "Virtual Sensors: Abstracting Data from Physical Sensors," in *Proc. of the Intl. Symposium on on World of Wireless, Mobile and Multimedia Networks*, 2006.

[10] M. Quigley, K. Conley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[11] A. Lakshman and P. Malik, "Cassandra - A decentralized structured storage system," *Operating systems review*, vol. 44, pp. 35–40, 2010.

[12] S. Vijaykumar and S. Saravanakumar, "Future Robotics Database Management System along with Cloud TPS," *Intl. Journal on Cloud Computing: Services&Architecture (IJCCSA)*, pp. 431–438, 2011.

[13] J. van der Veen, B. van der Waaij, and R. Meijer, "Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," in *Proc. of the 5th Intl. Conf. on Cloud Computing (CLOUD)*, 2012, pp. 431–438.

[17] See an example at: `www.youtube.com/watch?v=y6LqLNB4VDk`

[14] A. Dietrich, S. Mohammad, S. Zug, and J. Kaiser, "ROS Meets Cassandra: Data Management in Smart Environments with NoSQL," in *Proc. of the 11th Intl. Baltic Conference (Baltic DB&IS)*, 2014.

[15] W. Vogels, "Eventually consistent," *ACM Communications*, 2009.

[16] 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 *Proc. of Emerging Technologies in Factory Automation (ETFA '10)*, Bilbao, Spain, 2010.

[17] W. Meeussen, J. Hsu, and R. Diankov. (2012, 4) URDF - Unified Robot Description Format. [Online]. Available: www.ros.org/wiki/urdf

[18] R. Diankov, R. Ueda, and K. Okada, "COLLADA: An Open Standard for Robot File Formats," 2011. [Online]. Available: www.jsk.t.u-tokyo.ac.jp/rsj2011/_downloads/2Q1-5.pdf

[19] D. Meadows, W. Behrens, D. Meadows *et al.*, *Dynamics of Growth in a Finite World*. Camebridge, MA: Wright-Allen Press, 1974.

[20] P. Johnson-Laird, *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard Univ. Press, 1986.

[21] A. Dietrich, S. Zug, and J. Kaiser, "Towards Artificial Perception," in *SAFECOMP 2012 Workshops*. Springer-Verlag Berlin, 2012.

[22] ——, "Geometric Environment Modeling System," in *Conf. on Manufacturing Modelling, Management and Control (IFAC)*, 2013.

[23] D. Roy, K. Hsiao, and N. Mavridis, "Mental Imagery for a Conversational Robot," *IEEE Transactions on Systems, Manufactoring and Cybernetics*, vol. 34, pp. 1374–1383, 2004.

[24] A. Dietrich, J. Kaiser, S. Zug *et al.*, "Application Driven Environment Representation," in *The 7th Intl. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, 2013.

[25] E. Angelopoulou, T. Hong, and A. Wu, "World model representations for mobile robots," in *Proc. of the Intelligent Vehicles Symp. (IV)*, 1992.

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

[27] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous robots*, vol. 15, pp. 111–127, 2003.

[28] R. Harle and A. Hopper, "Dynamic world models from ray-tracing," in *Proc. of the 2. Annual Conf. on Pervasive Computing and Communications (PerCom)*, 2004.

[29] H. Caulfield and J. Johnsonb, "Artificial Perception and Consciousness," in *Proc. of the 6th Intl. Conf. on Education and Training in Optics and Photonics*, 2000.

[30] K. Hsiao, N. Mavridis, and D. Roy, "Coupling perception and simulation: Steps towards conversational robotics," in *Proc. to the Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[31] 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)*, 2010.

[32] A. Belkin, A. Kuwertz, Y. Fischer *et al.*, "World Modeling for Autonomous Systems," *Innovative Information Systems Modelling Techniques*, vol. 1, pp. 135–158, 2012.

[33] A. Kuwertz, "Towards Adaptive Open-World Modeling," Vision and Fusion Laboratory, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Tech. Rep., 2012.

[34] J. McCarthy, "Situations, Actions, and Causal Laws," Stanford University Artificial Intelligence Project, Tech. Rep., 1963.

[35] H. Levesque, R. Reiter, Y. Lesperance *et al.*, "GOLOG: A logic programming language for dynamic domains," *The Journal of Logic Programming*, vol. 19, pp. 59–83, 1994.

[36] M. Tenorth and M. Beetz, "KnowRob – Knowledge Processing for Autonomous Personal Robots," in *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[37] S. Blumenthal, H. Bruyninckx, W. Nowak *et al.*, "A Scene Graph Based Shared 3D World Model for Robotic Applications," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013.

[38] M. Waibel, M. Beetz, J. Civera *et al.*, "RoboEarth," *IEEE Robotics & Automation Magazine*, vol. 18, pp. 69–82, 2011.

[39] A. Saffiotti, M. Broxvall, B. Seo *et al.*, "The PEIS-ecology project: a progress report," in *Proc. of the ICRA Workshop on Network Robot Systems*, 2007.

[40] L. Parker, K. Fregene, Y. Guo *et al.*, "Distributed heterogeneous sensing for outdoor multi-robot localization, mapping, and path planning," *Multi-Robot Systems: From Swarms to Intelligent Automata*, 2002.

[41] R. Arumugam, V. Enti, L. Bingbing *et al.*, "DAvinCi: A cloud computing framework for service robots," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[42] S. Zug, F. Penzlin, A. Dietrich *et al.*, "Are Laser Scanners Replaceable by Kinect Sensors in Robotic Applications?" in *IEEE Intl. Symp. on Robotic and Sensors Environments (ROSE)*, 2012.