A New Concept for a Cooperative Fusion Platform*

Wendelin Feiten[1], Susana Alcalde Bagüés[1], Michael Fiegert[1], Feihu Zhang[2], Dhiraj Gulati[2] and Tim Tiedemann [3]

***Abstract*— The increasing traffic and the increasing number of sensors both in cars and in the infrastructure pose new challenges but also create new opportunities for traffic control. If the sensor data in various states of interpretation and aggregation could be shared and reused, it would be possible to minimize accidents and improve the traffic situation. In this paper we describe an approach to automatically configure sensor data fusion systems across the boundaries of independent subsystems, where information on all levels can be exchanged. The basis for this is a formal description of all required meta-information that enables the reasoning for automatic configuration.**

I. INTRODUCTION

An ever increasing number of sensors are being installed in the car and in the infrastructure. If we want to use this information for traffic monitoring and control we need to be able to aggregate the information available on the fly. As different vehicles enter a given area, a completely new set of participants in an overall system emerges that has never before been composed in exactly this manner. In order for the individual vehicles and the overall system to perform optimally, the sensor data need to be exchanged and integrated. We need a new solution for dynamic and adaptive sensor data fusion able to switch fully automated between different sensor configurations.

In this paper we present the background, the ideas and the solution approaches for our smart, adaptive data aggregation system, for short SADA.

A. Purpose of SADA

SADA could be seen as a generic fusion system. We develop a new way of doing fusion. Our objective is to adapt the fusion process at run time to the sensors and the applications available. In SADA the per se independent subsystems team up to form one larger system. As an example, look at a car that needs to be parked in a road side parking spot (see Fig 1). Maybe the car doesnt have enough sensors itself, so it uses the ultrasonic sensor of the car in front of the parking spot, and a camera that is mounted in the infrastructure and optimally perceives the parking spot. The information on the distance from the curb and from other cars is calculated from the data of those sensors and transmitted back to the driver (or to the software module that actually parks the car automatically). In this paper, we explain how we want to make it possible that subsystems can configure themselves automatically.

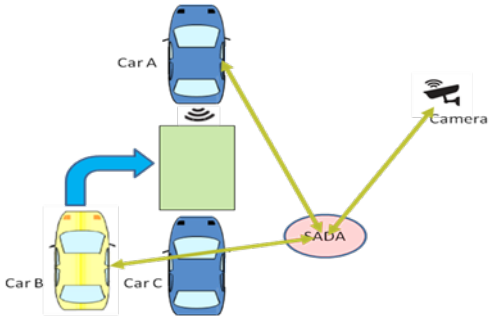

Fig. 1. Application Example Parking

B. Related work

SADA is not the first attempt to make cars exchange information with other cars and with the infrastructure. Basically, a car radio does it, receiving traffic information and thus informing the driver. This information concerns reports like traffic congestions, weather conditions or construction sites. The same type of information is also sent digitally over the radio data system. Navigation systems use information from the infrastructure, i.e. the GPS signals and traffic information via radio. Navigation applications in smart phones use GPS, too, and in addition receive information over the wireless phone and also send information back to the infrastructure. Even car to car information exchange has been investigated in research projects like SimTD, Ko-FAS, Drive2X, Testfeld Telematik [1].

The topic generic data fusion was also part of the subproject ProFusion 2 of the EU project PReVENT. There, the focus was mainly on a generic data store for data fusion applications. Adaptive or even context sensitive data fusion methods were not considered. Subsequent EU projects dealing with traffic safety but effort in concrete data fusion algorithms but without picking up the subjects adaptivity and context sensitivity [2].

Also, higher aggregated information has been used within one car. Traditionally an application was very much vertically integrated, each one using its own sensors and its own processor. Meanwhile there are attempts to reuse the higher

*This work was partly funded by the Federal Republic of Germany, Ministry for Economic Affairs and Energy, within the program IKT EM III, grant no. 01ME14002A.

[1]Susana Alcalde Bagüés, Michael Fiegert and Wendelin Feiten are with Siemens AG, Corporate Technology, Munich, Germany, e-mail: {wendelin.feiten, susana.alcalde, michael.fiegert}@siemens.com

[2]Feihu Zhang and Dhiraj Gulati are with fortiss GmbH, Munich, Germany, e-mail: {zhang,gulati}@fortiss.org

[3]Tim Tiedemann is with DFKI GmbH, Robotics Innovation Center, Bremen, Germany, e-mail: tim.tiedemann@dfki.de

level information like tracks and occupancy maps across applications.

C. What is new in SADA

The projects mentioned above use a fixed catalog of information types that are transmitted between subsystems. SADA aims to reuse the information even further. It will add flexibility to the exchange of information inside a car as well as between cars and the infrastructure. It will allow the use of all kinds of information at all levels of aggregation. This is only possible as long as the information formally specify its content as meta-information. That means, for a sensor, the data produced in a given environment depends on the physical properties of the sensor and on the properties of that environment. Therefore information about the sensor and the environment is required to interpret the sensor data in a meaningful way. Today, this kind of information does not have a formal definition or at least is not usable automatically. But a fusion engineer needs to take this information into account to tune a sensor data fusion system and meet the quality requirements of a given application.

Today data fusion systems are designed with a particular purpose in mind. Each application is implemented completely independently of others. The classic development of sensor fusion is therefore expensive. SADA will explicitly model the formal properties of sensors and environments and use this explicit knowledge to automatically adjust the sensor fusion algorithms. This will result in a generic fusion system that can meet the requirements of any application. Furthermore, the information transmitted between cars and infrastructure today is only high level. SADA will allow including all levels of aggregated data, from raw sensor readings to entire scene models, inside one car as well as between cars and infrastructure. This flexibility can only be achieved when the meta-information and its formal models are available along with the sensor data.

Configuring a state of the art sensor data fusion system involves many manual steps: i) selecting the sensors to use, ii) selecting and configuring the preprocessing steps and filters; iii) selecting and configuring the inference method; iv) defining the control flow (what to do when). SADA aims to implement a system that can perform these steps automatically. The sensor fusion system should configure itself, depending on the task and on the available resources.

II. COOPERATIVE FUSION PLATFORM

In order to better explain the differences between a usual sensor data fusion process and the SADA approach, we briefly summarize the general structure of our cooperative fusion platform in section A. Its modifications and extensions are described step by step.

A. General fusion architecture

1) Classical view: In the classical view of a sensor data fusion system (Fig 2), a sensor captures properties of the environment and produces sensor data.

The information flow is seen as going from the environment over the sensor to the sensor data.

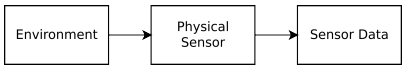

Fig. 2. Classical view of Sensor Data

2) Separation of Sensor and Environment: For the automatic generation of an algorithm to interpret the sensor data, the meaning of the data needs to be understood by the machine. This understanding need to be based on properties of the environment and on properties of the sensors, independent of one another, as explained in more detail in section B. The next stage to achieve a cooperative fusion platform leads to the generation of a Measurement process that follows the so called measurement model (see section B) to generate the expected sensor data for the application. With that in mind we represent a basic architecture of a sensor data fusion system as shown in Fig 3.

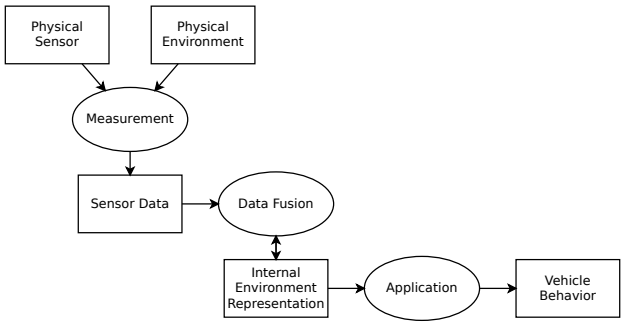

Fig. 3. The basic architecture of a sensor data fusion system

3) Generic Inference: In the basic architecture, usually the Data Fusion part is tailored to the properties of physical sensors and environment and to the requirements of the representation of the environment, given by the application. On the other hand our cooperative fusion platform has a number of generic inference algorithms available to be used to perform the data fusion. We name it Generic Inference.

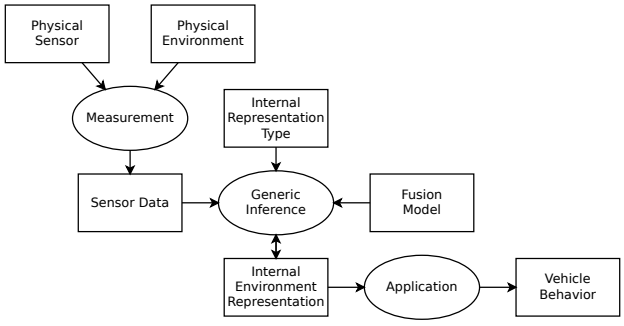

Fig. 4. Architecture of a cooperative fusion platform

This Generic Inference needs to be adapted to the type of the internal environment representation, which in turn

is specific for the application. What means that the way the sensor data is interpreted by the algorithm needs to be adjusted to the requirements of the application. These are parameters that also depend on the properties of the sensors and of the environment. In the architecture proposed for the cooperative fusion platform (see Fig. 4), the collection of these parameters to be applied by the fusion algorithm is called the Fusion Model.

4) Simulation: One central task of the engineers in developing a sensor fusion system is to determine the way that the sensor data are interpreted with respect to the resulting representation of the environment. This involves a lot of experimentation. If the configuration of a fusion system is to be done automatically, experimentation in the real world generally is not an option. One way out of this is the use of simulation. In the SADA approach, the properties of the sensors and the properties of the environment are modeled in sufficient detail such that a simulation will produce the same (or sufficiently similar) sensor data as result from real world measurements. This simulation uses a measurement model, described in more detail in B.4. The simulation is illustrated in the top left corner of Fig. 5. The models and the simulation algorithm can be (and need to be) validated by experiments.

5) Get the Fusion Model by Machine Learning: The Fusion Model is determined in a way that the resulting Internal Environment Representation is as close as possible to what the application needs. What the application needs is expressed in an ideal Target Internal Representation which is derived directly from an Environment Model, taking into consideration the requirements of the application (see Fig. 4).

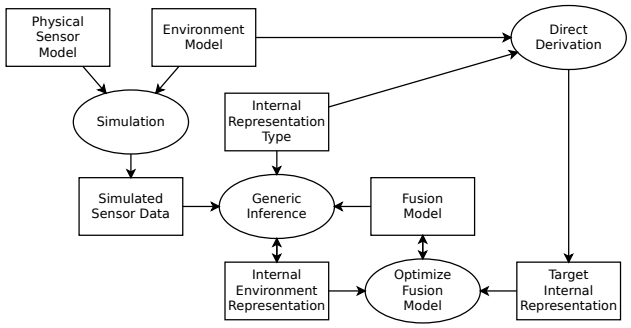

Fig. 5. Machine Learning of the Fusion Model

With this target and the simulation of sensor data, the parameters in the Fusion Model can be optimized in such a way that the Generic Inference produces an Internal Environment Representation as close as possible to the target. In other words, the Fusion Model is obtained from the task by Machine Learning.

6) Creating instances of sensor data fusion systems: In a cooperative fusion platform for each application, a specific instance of a sensor fusion system according to Fig. 5 needs to be created. A new fusion topology is generated for each application based on the availability of sensors. This is done using the information of the formal models (section B), higher level information and semantic descriptors of the instances corresponding to the modules of the SADA architecture.

To achieve the goal of a cooperative fusion platform the complete fusion process in SADA is divided and redesigned into logical modules. Modules are made data driven. This modularization helps to achieve the configuration of the system on the fly, as the modules can be inserted and removed on demand, the modules are described semantically to facilitate the automatic reconfiguration of the system. As an example of how it works, starting from the available resources (e.g. an ultrasonic sensor measurement), a reasoner component selects applicable instances of pre-processing and inference algorithms (e.g. a grid update algorithm) to produce the desired internal representation (e.g. an occupancy grid map). A planner composes these applicable instances to create the fusion system.

B. Higher level information and context information

If a sensor fusion system is to be configured automatically, no engineer will set up parameters or feed it with the suitable sensor models and environment models. But a program can take over what the expert engineer does if the knowledge is made available and understandable by the program. The information need to be written beforehand and must be independent of the application.

The problem is that such information comes from different domains. The manufacturers of sensors know about the properties of their sensors. The developers of fusion algorithms know about the scope and properties of their algorithms. Application developers define what kind of representation of the state of the world they prefer. All this information needs to be brought together to design a sensor fusion system. In the following sections different information domains that need formal description to enable fully automated configuration of a fusion system are described.

1) Sensors: Sensors are an integral part of any data fusion system. The interpretation of the sensor data depends of the sensor physical properties and also of the processing algorithms they incorporate. This information is a key part to implement a fusion system. For instance, for a camera we need to know the focal length, pixel counts, pixel sizes, radial distortions, and field of view. For the radar sensor: opening angle, range, carrier frequency, modulation, pulse length and pulse shape, etc. These are parameters are best known by the respective manufacturers.

2) Smart Sensors: It is not only the physical properties of the hardware that need to be described. This can be understood following the example of the radar sensor. In a radar sensor, the raw signals are processed already in the sensor to reduce the amount of wrong measurements and to get rid of noise. This processing depends on assumptions done about the environment. A radar sensor that has been designed to return tracks (which means a vehicle tracked

over time) from a measurement on a highway will return nothing but noise if operated in a closed room that contains much metal. The exact algorithm is in general considered a trade secret of the manufacturer. However for the fusion engineer it is very important to understand the behavior of the sensor.

One might think that these are two different aspects of a sensor: the hardware and the processing software. In this regard the sensor actually consists of two parts the raw data acquisition part and the raw data processing part. This type of sensor, that contains preprocessing software is therefore often called smart sensor.

The designer of the overall system needs to understand the measurements returned by the sensor.

3) Environment: Along with both the physical properties of the sensor as well as the data processing steps, the properties of the environment influence crucially the sensor data and the conclusions that might be drawn from the data. For example, if a laser scanner is placed in the desert and does not return any measurements, probably there are no obstacles to be detected. On the other hand, if a laser scanner is placed in a mirror maze and also does not return any distance there, this should be not interpreted like there are no obstacles. The properties of the environment need to be also described and formal modeled. It should be possible to describe the environment properties that determine how sensors perceive this environment. For radar sensors this depends both on the geometry (to a large extent) and on the reflectance.

4) Measurement model: The measurement model says, given an environment, which sensor data will be returned from the sensor. Since the measurement model depends on both the sensor properties and the environment properties, neither the sensor experts alone nor the environment modeling experts alone can create the appropriate measurement model.

5) Generic Fusion algorithm: In SADA the uncertainty that is associated with any sensor data is expressed via probability density functions and fused with probabilistic inference algorithms. To select the appropriate algorithm for an application, based on its requirements, we need to add the meta-data that will allow a system automatically to select the right algorithm for an application. In SADA we concentrate on the most general algorithms. One very basic and often used algorithm is the Kalman Filter (see Section III.b). Another very generic class of sensor data fusion algorithms is the probabilistic graphical model, especially the factor graphs. We use GTSAM for the solution of the factor graph. For more detail refer to [3].

6) Internal representations: The most popular internal representation types, known as environment representations, for the automotive applications are occupancy grid maps and track lists. The properties of the internal representations also need to have a semantic formal description. This is e.g. in a track list position and orientation in a two dimensional plane. In order to be machine readable it needs to be formally stated to which coordinate system this refers. Very often it is the vehicle kinematic coordinate system, which is located at the center of the rear axle, with the x-axis pointing into the driving direction, which needs to be said.

C. Description with semantic models

There are different aspects that influence the decision on how to formally represent the additional information mentioned in the previous section B. They can be roughly characterized by the concepts pairs: descriptive vs. functional; complete vs. incremental and norms vs. ontologies.

1) Descriptive vs. procedural: A descriptive knowledge model describes what a thing is, what properties of a thing constitute the fact that a thing belongs to a certain class of things, or that one class of things is a subclass of another class of things. This is a more static view of what the world is. In general the things in the world are organized in concept hierarchies. A functional knowledge model says how new information on the world can be derived from previously known information.

2) Complete vs. incremental: A complete specification of terms and concepts tries to describe all the aspects that might become relevant in great detail. This needs to be made very thoroughly. The incremental specification tries to describe the terms and concepts as the need for them appears and checks how consistent they are with the existing descriptions.

In practice this leads to hierarchical models that at the top level only contain the basic information of a concept. Each time something new needs to be described, i.e. some other instance needs to be derived; this may depend on more detailed information about the concept. This detail is then added to the concept. With increasing detail, this results in a concept hierarchy.

3) Norms vs. ontologies: Today the typical means to provide for interoperability is defining norms and standards and sticking to them. A norm is in general endowed with some authority (government authorities, industry consortia, etc.) Norms specify the terms and agreements on shapes, functions, protocols etc. as precisely and completely as possible, so that whenever somebody wants to use these there is no more source of misunderstanding and no need for further agreements. Machine readability is not yet wide spread with norms.

For ontologies, machine readability is a basic feature. There is a formal description language (OWL 2) with a wide support and many related tools and technologies, like inference mechanisms, service detection and service composition. The interoperability of different domains is supported by upper ontologies. The trend is towards automated cooperative knowledge processing.

D. SADA approach

In the SADA project, we strongly lean towards the formal description of information using ontologies. However, the available tools turned out not to be ideal for our project. The available editors are not well suited for the joint development of concepts and of operators (services, algorithms, etc) that relate concepts with each other. It was also not possible to

define a concept incrementally using a hierarchical representation. We also found that the available reasoners used today, together with known ontologies, generally make an open world assumption. The answer to a query is always yes if a solution is possible, where for SADA a reasoner is needed that returns a constructive proof relying on the closed world assumption. Therefore, we need to improve the user interface to support the interactive development of higher level descriptions, support the definition and use of hierarchical models and integrate this with reasoners/planners that are based on the closed world assumption.

1) Hierarchical concept definition: The concepts in SADA are developed step by step and can be used at various levels of specialization. As an example see Fig. 6, where a car needs to estimate the position and orientation in a city environment.

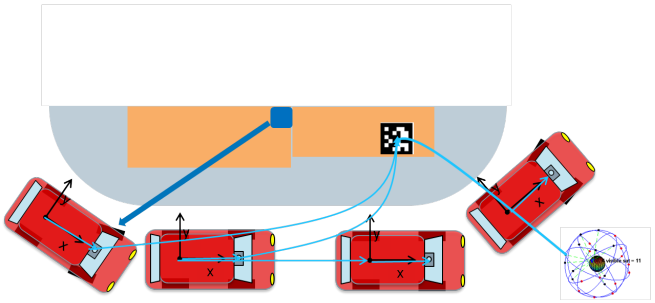

Fig. 6. Localization with landmark

In this example, a car needs to localize itself with respect to world coordinates. The available resources are: 1) a camera in the car, and the position and orientation (for short: pose) of the camera w.r.t. the car are known; 2) a landmark, and the pose of the landmark w.r.t. the camera can be determined by some algorithm and 3) a communication so that the car learns about the landmark and its pose w.r.t. world coordinates from the infrastructure.

The high level description of this information is given with the concept Pose3D. Fig. 7 shows an excerpt. This is similar to the work of Stocker et al. (see [4]). In order to be useful the reference coordinate system of the pose and the coordinate system whose relative pose is described need to be stated explicitly. So the first part says that a Pose3D has both a Reference of the type CS3D and a Referent of the same type, and that a CS3D has an identifier ID.

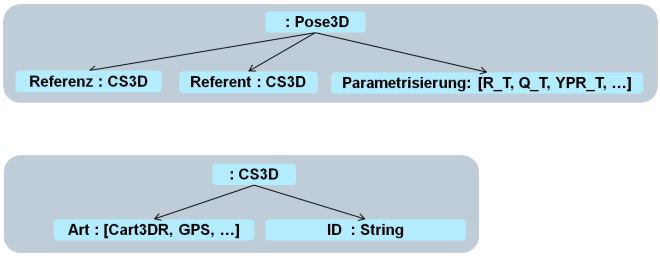

Fig. 7. Concept hierarchy

2) Planning: These descriptions of the resources are used for planning to put the resources together to produce the required information. As planner a for example a forward production system can be used. As an easy example we can formulate the operation Invert. It produces the pose of the former Reference coordinate system w.r.t. to the former Referent coordinate system (see Fig. 8). A similar operator can be defined for the composition of poses on the same level of detail. With these operators, the planner can produce the pose of the car w.r.t. the world from the available resources.

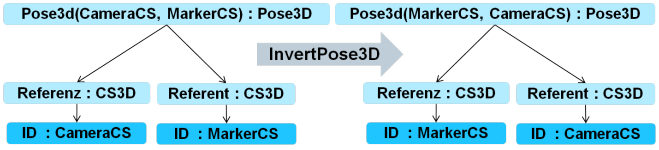

Fig. 8. Operator Invert

III. SMART DATA AGGREGATION ALGORITHMS

A. Generic Fusion Algorithms

The imperfection of data is the main challenging problem of the fusion systems for which various approaches have been proposed in [5]. General fusion algorithms use probability theory to handle the uncertainties during the fusion process. The Bayes filter is thus developed to estimate the state of a dynamic system from observations. The well known Kalman Filter (KF) is proposed as a special case of the Bayes filter with the assumptions in linear Gaussian cases [5]. When dealing with nonlinear systems, the Extended Kalman Filter (EKF) [6] and the Unscented Kalman Filter (UKF) [7] are often utilized. However, both filters can only handle simple nonlinear filtering to a limited extent. To solve this problem, the particle filter is proposed to the densities in the Bayes filter based on random sample approximations [8].

B. Generic version of the Kalman Filter

The Kalman Filter can be considered as a Generic Inference algorithm. The information flow for one time step is shown in Fig. 9.

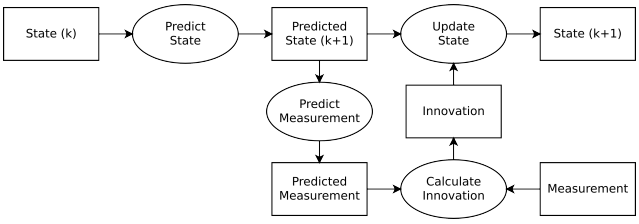

Fig. 9. The generic Kalman Filter

The basic formulae are practically identical for a wide range of applications. There are already products to support the design of sensor data fusion systems based on this observation (BASELABS). To this purpose, the core Kalman Filter algorithm is implemented in a way that different application specific parts can be easily exchanged (i.e. the state spaces

for the tracks and the sensors). The compatibility of the signals between these modules is then checked automatically, based on the type declarations of the signals. Below the connection to the higher level descriptions as suggested above are explained.

C. State space and state prediction

The Kalman Filter tries to estimate the state of a system. In the traffic scenario, this could be the poses of the vehicles (including ego vehicle) and their velocities and accelerations, poses of environment features and so on. In general, this state is expressed with a vector or real variables. In terms of the higher level model using a hierarchical concept, the state is an instance of a concept. The function that predicts the next state is an operator that produces a new instance (with a new time stamp) from a given instance. For the Kalman Filter, it doesnt really matter how this instance is derived in detail, as long as it complies with the desired type. This type also contains as one part the covariance of the predicted state.

D. Measurement prediction and measurement

Similar to the state prediction operator, there are measurement prediction operators. Depending on the state (which is described including the meta information and higher level information by a concept), and on the properties of the environment and of the sensors, both of them again described via concepts, the expected measurement is derived. The measurement is again an instance of a concept. In the Kalman Filter case, the measurement prediction instance is characterized by a vector of real numbers together with a covariance matrix.

A measurement is based on sensor readings, and in the Kalman Filter the sensor readings need to be processed to result in an instance of the same concept as the measurement prediction is. The difference between the prediction and the measurement is what is new to the system, also called the Innovation.

E. Kalman gain and measurement update

Based on the innovation, another operator creates a new instance of the concept of the state. There is a formula for that for the Linear Kalman Filter and also for some other cases, and there are generic implementations of this operator (an easy sequence of matrix vector operations). The suitable instance of this operator can then be selected from a library or it might be generated by a code generator.

From the theory it is known that the probability distribution function (pdf) of the state estimate after the measurement update is more concentrated than the pdf of the state prediction (i.e. before the measurement update). If the perception goal is to have a state estimate with a concentrated pdf, this meta information tells the planner that the Kalman Filter produces this desired instance (of whatever concept is encoded in the state in a special case).

F. Probabilistic Graphical Model

An alternative solution, compared to Bayes filtering techniques, is the graph based formulations for batch processing. The benefit is the flexibility to nonlinear models in contrast to Bayes filtering. It addresses the nonlinear issues by using optimization algorithms such as Gauss-Newton iterations or the Levenberg-Marquardt algorithm [9]. Many approaches have been developed like the SAM Algorithm [10], its incremental variant iSAM [10] and the g2o framework [11]. The use of the probabilistic graphical model is described in more detail in [3].

IV. CONCLUSIONS

The parallel development of the formalism for capturing the hierarchical models and the inference is under way, and the resulting editor/viewer supports rapid development of the ontologies.

Also, various approaches have been tested for the higher level inference (SWIProlog with an extension to track the successful predicates, and a forward production rule system), with the result that the high level inference is technically feasible.

In parallel we investigate how these developments can be related to the existing set of tools and concepts for the semantic web (OWL 2, DL, service description, service discovery etc.)

REFERENCES

[1] S. A. Bagüés, T. Tiedemann, C. Backe, D. Gulati, S. Lorenz, and P. Conradi, "Towards dynamic and flexible sensor fusion for automotive applications," Accpted International Forum on Advanced Microsystems for Automotive Applications (AMAA 2016), 2016.

[2] J. Dokic, B. Mller, and G. Meyer, "European roadmap smart systems for automated driving," 2015. [Online]. Available: http://www.smart-systems-integration.org/

[3] M. Fiegert and M. Hutter, "Rule based generation of factor graphs for smart adaptive data aggregation," Submitted IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2016.

[4] M. Stocker, M. Rönkkö, and M. Kolehmainen, *Making Sense of Sensor Data Using Ontology: A Discussion for Residential Building Monitoring*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 341–350.

[5] B. D. Anderson and J. B. Moore, "Optimal filtering. 1979," 1979.

[6] N. Sünderhauf, "Robust optimization for simultaneous localization and mapping," 2012.

[7] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. Ieee, 2000, pp. 153–158.

[8] A. Doucet, N. J. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump markov linear systems," *Signal Processing, IEEE Transactions on*, vol. 49, no. 3, pp. 613–624, 2001.

[9] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[10] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1365–1378, 2008.

[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.