Virtual Sensors and Data Fusion in a Multi-Level Context Computing Architecture

Bastien Pietropaoli, Michele Dominici, Frédéric Weis

Virtual Sensors and Data Fusion in a Multi-Level Context Computing Architecture

Bastien Pietropaoli
INRIA, Rennes-Bretagne Atlantique,
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Email: Bastien.Pietropaoli@inria.fr

Michele Dominici
INRIA, Rennes-Bretagne Atlantique,
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Email: Michele.Dominici@inria.fr

Frédéric Weis
IRISA, Université de Rennes 1,
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Email: Frederic.Weis@irisa.fr

***Abstract*—Computing context is a major subject of interest in smart homes. In this paper, we present how we adapted a general purpose multi-level architecture for the computation of contextual data to a prototype of smart home. After a quick explanation of why we use different methods at different levels of abstraction, we focus more on the low-level data fusion. To do this, we present the basics of belief functions theory and how we apply this theory to sensors to obtain stable abstractions. By doing this, we highlight the major problem appearing when processing directly sensor measures. We respond to this problem by introducing an abstraction of sensors we call virtual sensors. Some examples of virtual sensors are given.**

I. Introduction

Computing context is a major subject of interest in smart spaces such as smart homes. Contextual data are necessary for services to adapt themselves to the context and to be as efficient as possible [1].

Contextual data may be obtained via augmented appliances capable of communicating their state and a bunch of sensors. It becomes more and more real with the development of the Internet of Things [2]. Unfortunately, the gathered data are not always directly usable to understand what is going on and to build services on them. Thus, data fusion is required.

Data fusion is a complex task as many data imperfections can appear [3]:

- *Randomness*, due to physical systems (in our case, sensors);
- *Inconsistency*, due to overload of data or conflicting sources;
- *Incompleteness*, due to loss of data which may easily happen with wireless sensors;
- *Ambiguity* (or fuzziness), due to the model or to the natural language imprecision;
- *Uncertainty*, due to not fully reliable sources;
- *Bias*, due to systematic errors;
- *Redundancy*, due to multiple sources giving the same information.

Those imperfections are handled by many theories [4]. Many of them are commonly used to infer contextual information from sensors [5]–[12].

All the existing methods are interesting and offer advantages but most of them do not provide exactly the same type of contextual data and none of them offer a general solution for context reasoning [13]. Moreover, most approaches rely directly on sensor measures making them hard to redeploy in many different environments. As a matter of fact, any system based on sensors requires an *ad hoc* tuning phase as the behavior of sensors can slightly change from one environment to another. Solutions based on learning directly from raw sensor measures are also very sensitive to the quality of the training set.

In terms of conception of systems, reasoning directly on sensors to build high level abstractions or services is also difficult and requires knowledge and experience.

The main aim of this paper, focused on a case of smart home, is thus to illustrate why a multi-level context computing architecture can be interesting. We present how it is possible to merge different theories by feeding context models with *stable abstractions* of adapted levels. Those stable abstractions enable to repel the unavoidable *ad hoc* part of the system as close as possible to the sensors.

The paper is thus organized as follows: first, we present the different existing approaches for context computing we are using and we present how it is possible to merge them in a unique multi-level context computing architecture. Then, we present our use of the belief functions theory (BFT) at the lower-level of our architecture, its advantages and the major drawback. It is followed by the introduction of virtual sensors using common techniques to stabilize the sensor layers. Examples of virtual sensors are then presented. Finally, the paper is discussed and concluded.

II. Multi-level context computing architecture

In 2005, Coutaz *et al.* [1] suggested that context-aware applications should rely on multiple levels of abstractions. As depicted in Fig. 1, it is divided in four layers :

- *Exploitation layer*: the highest layer, it exploits contextual data to provide adapted services
- *Context and situation identification layer*: this is what analyzes ongoing situations and potentially predicts future situations

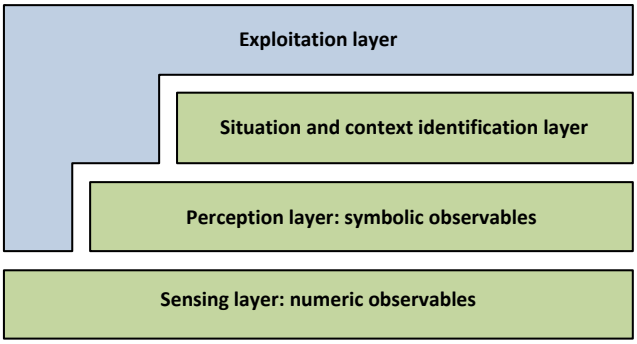

Fig. 1. Multi-level context computing architecture of Coutaz *et al.* [1].

- *Perception layer*: it offers a first layer of abstraction for small pieces of context independent of deployed sensors
- *Sensing layer*: it mainly consists of the data gathered by sensors

In this architecture, every layer is based on the results of its underlying layers. A similar decomposition of context computing is described by Bettini et al. in survey on context modeling and reasoning techniques [13]. It seems totally pertinent to us as in everyday life, when we are doing something such as cooking and someone asks what we are doing, we first answer that we are *cooking* independently of what we are really doing such as *mixing*, *breaking eggs*, etc..

As a consequence, we are convinced that when we build models, we first think of higher level of abstractions and then decompose in sub-abstractions and so on. Our idea here is thus to find methods that enable the building of such levels of abstractions.

With this idea in mind, we looked for theories and methods used for context computing and how we could merge them to fit the general purpose architecture given by Coutaz *et al.*.

A. Activity recognition

As we explained, the first level of abstraction coming to mind when describing what we are doing is high level abstractions such as *cooking*. Those activities are then the highest level abstraction we want our system to be able to identify. Plan recognition algorithms are used to analyze sequences of actions and thus predict future actions of users. It could be, in our case, adapted to identify ongoing activities and predict future ones. There exist different plan recognition algorithms [14]. However, one interested us particularly, PHATT introduced by Goldman, Geib and Miller [15].

In order to understand how PHATT is working, it is important to understand the hierarchical task network (HTN) planning problem which is "inverted" by the algorithm to perform plan recognition. It consists in automatically generating a *plan* starting from a set of *tasks* to execute and some constraints [16], [17].

The problem relies on the specification of a plan library made of two components: the *tasks* to execute, which can be *primitive* if they don't ask for any further planning or *open*, otherwise, and the *methods*, which are prescriptions of how decomposing a task in (partially-) ordered sub-tasks. Note that a same task can be decomposed using different methods, thus resulting in different sub-task sequences. HTN planning proceeds by decomposing non-primitive tasks recursively into smaller and smaller subtasks, until primitive tasks are reached that can be performed directly.

The principle behind the PHATT algorithm is to perform plan recognition relying on three phases: defining the plan library, modeling the plan execution and recognizing the current execution, starting from the observations [15], [18], [19].

The plan library is modeled like in the HTN planning problem presented above. The plan execution is modeled as a stochastic, generative model that selects actions to perform from a set of enabled primitive tasks called *pending set*, which is dynamically defined depending on the previous actions performed by the agent, the agent's goals and the plan library [19].

Assuming this model of plan execution, PHATT takes as input a sequence of observations, which correspond to agent's actions, and generates the set of all possible explanations for the observed sequence of primitive tasks, in terms of executed plans and, thus, goals. It then uses Bayesian inference to calculate the probabilities of the generated explanations and goals. The PHATT algorithm is thus particularly efficient to produce predicted contextual data.

In our case, we would like to predict future situations depending of the previously observed situations. To give an example, if we want to predict that the situation *dinner* will occur soon, it may be sufficient to have observed situations such as *cooking* and/or *setting the table*.

Basing a plan recognition algorithm on smaller observations of activities such as *breaking eggs*, *mixing*, etc. may be going too deep into details without any real reason. As a matter of fact, it can be hard, if not impossible, to think of all the small events that can compose activities of daily living. Thus, we don't want to apply a plan recognition algorithm directly from sensor measures but on high level abstractions such as situations.

B. Situation inference

For situation inference, a recent framework called Context Spaces has been proposed [9], [20], [21]. The Context Spaces modeling relies on a structure of context management organized in three levels: sensors, context and situations. Starting from data captured by sensors, the theory offers methods and algorithms to interpret and process these data and arrives at a representation of the context, including facts, assumptions and predictions [22], [23]. Reasoning mechanisms are then applied on top of this context representation in order to produce an answer (and a degree of confidence in the answer) to the question "is a given situation currently occurring?". Those situations could then be used as an input for the plan recognition algorithm.

The Context Spaces are based on a geometrical metaphor to model situations (see Fig. 2). In addition to the fact it is simple to think and visualize situations, in the Context Spaces, it is possible to have a context state (i.e. a set of observations) that corresponds to none of the modeled situation. It is particularly

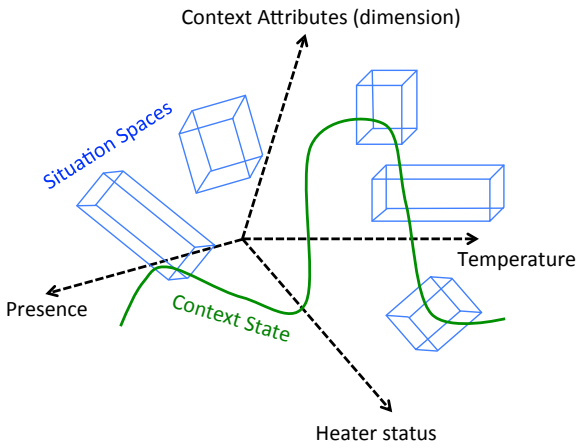

Fig. 2. Modeling of situations in the Context Spaces framework. The *context attributes* corresponds to small pieces of context such as *presence*, *temperature*, etc.. The solids called *situations spaces* represent the modeled situations where each context attribute is a dimension. The *context state* corresponds to the state of the system at a given time.

convenient as we do not want to always observe something. As a matter of fact, we do not claim to be able to model every possible daily activity in our smart home project.

Like for the plan recognition algorithm, when we think of what compose a situation, we often think in terms of general concept. If we stay with the cooking example, to describe this situation we may think of abstractions such as *presence in the kitchen*, *a cooking appliance is used*, and so on. Once again, knowing more details, for instance raw measures from sensors, may often be useless. However, the data required by situation inference mechanism are of lower levels of abstraction than situations required for plan recognition. Thus, we go deeper into details when going deeper in abstraction levels.

There exist other methods to infer situations. For example, in some work, the belief functions theory (BFT) is used [5]–[8] with binary sensors. As binary sensors can be considered of higher levels that sensor returning raw physical measures, those works are a good evidence that modeling situations is simpler with already high level abstractions.

C. Low-level data fusion

The information needed for situation inference are small pieces of contextual data such as, for instance, a presence in a room or a type of appliances being used. Those information may be obtained directly from binary sensors [5]–[8] but require sometimes low-level data fusion [12], [24].

A common theory to compute contextual data from sensors is the belief functions theory (BFT) (also called Dempster-Shafer theory or theory of evidence) [25], [26]. Based on the accumulation of evidence, it enables the use of multiple sources to infer contextual data, for instance the posture of someone [12] or the presence in a room [24]. For example, a presence in a room may be obtained from multiple sensors such as motion sensors, sound sensors, thin force sensors on chairs, etc..

This theory also offers powerful tools to handle conflicting sources and detect errors and failures [27]–[30]. Thus, it is particularly adapted to work with raw sensor measures.

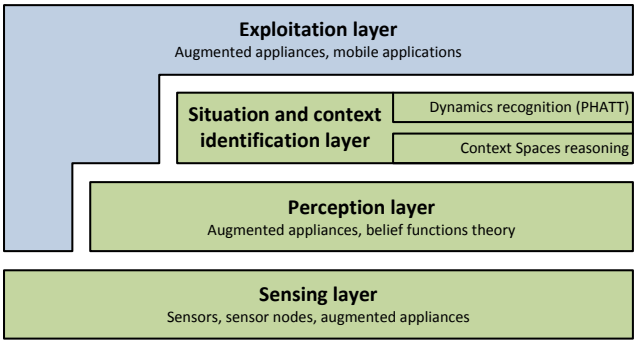

Fig. 3. Our multi-level context computing architecture (adapted from [1]).

D. Resulting architecture

All the presented methods offer different level of abstractions. Those abstractions can be thought easily by humans as they correspond to daily activity descriptions. Those abstractions are stable as they do not depend of the deployed sensors except for the lowest layer. The integration of the presented methods result in the architecture depicted in Fig. 3.

On the highest level, we place the plan recognition algorithm PHATT with input ongoing situations. Those situations are inferred, using the Context Spaces, from medium-level abstraction, called in our case *context attributes*, such as binary sensors on furnitures, augmented appliances capable of communicating their state and abstract data obtained from data fusion such as presence in a room, posture of someone, etc. The latter are computed using the belief functions theory applied to heterogeneous sensors.

As all the presented methods are capable of handling uncertainty, it is totally possible to propagate and conserve it from one layer to another [31]. At each level, it is possible to make a decision on the state that is considered as true and send to the upper layer a couple (state, confidence). It is also possible to transfer a set of possible states for the upper layer to be able to consider all possibilities. Each state is then attended with an equivalent of probability. The plan recognition algorithm normally uses probabilities associated to each measure. It can use confidence levels returned by the Context Spaces theory instead. And the Context Spaces theory can use degrees of beliefs as confidence level for the gathered context attributes. Thus, uncertainty is propagated through the different levels of the architecture.

III. Applying the Belief Functions Theory

For low-level data fusion, we decided to use the belief functions theory [25], [26]. In this section, we present the basics of the theory and how it is possible to apply this theory to sensors. We also discuss why this theory is adapted to the low-level data fusion and its limitations.

A. Frame of Discernment

In the BFT, the first thing that should be defined is a set of possible "worlds" called the *frame of discernment*. It is often noted:

$$\Omega = \{\omega_1, \omega_2, ..., \omega_n\} \quad (1)$$

These worlds have to be exclusive (i.e. $\omega_i \cap \omega_j = \emptyset$ if $i \neq j$) and if possible exhaustive meaning that the true state of the world has always been defined. To give an example, we can define a set of possible postures for someone as $\Omega = \{Seated, Standing, LyingDown\}$.

B. Mass function

Once the frame of discernment is created, a *mass function* (also called *basic belief assignment* or *body of evidence*) representing the degree of belief associated to each subset of Ω is defined such that:

$$m : 2^{\Omega} \longmapsto [0, 1]$$
$$\sum_{A \subseteq \Omega} m(A) = 1 \tag{2}$$

Each subset $A \subseteq \Omega$ with $m(A) > 0$ is called a *focal element*. When a focal element contains several worlds, the degree of belief given by the mass function cannot be distributed in any way between those worlds. The degree of belief is thus completely associated to the fact that the true state of the world is contained in the focal element but cannot be more specific. Thus, the mass function can represent uncertainty using degrees of belief but also imprecision using non-atomic subsets of Ω.

If Ω is a focal element, then $m(\Omega)$ is considered as the degree of *total ignorance*. If a mass function only has Ω as focal element, then it is called a *vacuous mass function* and represents a case where no evidence has been gathered.

C. Building mass functions

To build mass functions, it is possible to use methods exploiting statistics [32]. However, we wanted a method requiring only few experiments.

Instead of building mass functions from previous observations, we build for each sensor, applied to a context attribute, a set a mass functions. Fig. 4 gives an example of set of mass functions for a simple motion sensor in a case of *presence* detection ($\Omega = \{Yes, No\}$). The sensor used is a Phidget Motion Sensor. When connected to a USB interface, it returns a measure between 0 and 1000. A measure of 500 corresponds to no motion detected and any other measure is equivalent to its symmetrical around 500. This kind of set is built on intuition and can be fine-tuned after few experiments.

Once a set of mass functions is built, a projection on this set is done in order to obtain the corresponding mass function each time a raw data from that sensor is received. For instance, with the given Fig. 4, if the motion sensor returns a value of 450, then the resulting mass function would have two focal elements: $m(\{Yes\}) = 0.7$ and $m(\{Yes \cup No\}) = 0.3$. It is also possible to temporize beliefs to propagate them through time and potentially take into account previous beliefs [24].

A constraint to respect when building these sets of mass functions is the *least commitment principle*. In our case it can be translated by the fact that the belief induced by a sensor measure should not be too specific when it cannot be. In the given example, the motion is only a proof that somebody may be there but the gathered measure is not a good proof that nobody is there. That is why the set $\{No\}$ never appears in the set of mass functions in Fig. 4.

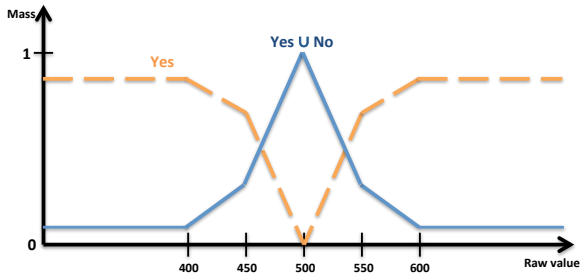

Fig. 4. Example of a set of mass functions associated to a motion sensor in the case of presence detection. To each measure given by the sensor is associated a mass function.

While this method is efficient to quickly build belief models, it is directly connected to the output of each sensor. Biased and noisy measures can cause major modifications on the resulting beliefs. This problem is not specific to our way of building belief functions because if we had chosen a method based on statistics, the statistics may be dependent of the bias and noise caused by a specific environment as well.

Finally, even if methods enable the detection and management of faults and failures [27]–[30], we present in the section IV how it is possible to get rid of the dependency between models and bias/noise of sensors.

D. Accumulating evidence

The theory of belief functions is about accumulating evidence on what is going on. To do this, the most common rule of combination is the Dempster's rule given by (for all $A \subseteq \Omega, A \neq \emptyset$):

$$m_1 \oplus m_2(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C) \tag{3}$$

$$\text{with } K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \tag{4}$$

$$\text{and } m_1 \oplus m_2(\emptyset) = 0 \tag{5}$$

This rule of combination is completely general and does not require identified pieces of evidence to work. Thus, when some data are lost, in wireless communications for instance, it is still possible to fuse the gathered evidence. The resiliency of the system is maximum when using this theory directly on sensor measures. Moreover, this combination rule enables the use of heterogeneous sensors for the computation of context attributes. One may not rely on only one type of sensors.

Another advantage of using such rule of combination is that it enables the use of different configurations of sensors to compute the same context attribute. The configuration can depend on where the system is deployed. For instance, if we want to detect the *presence* in diverse rooms, we may use different sensors for the kitchen, the bedroom, the bathroom and so on. While motion sensors may be good evidence of the presence of someone in the kitchen or the bathroom, thin force sensors on chairs, sofa and bed may be better of evidence of presence in the living room and the bedroom. The context

Subsets	$\{Yes\}$	$\{No\}$	$\{Yes \cup No\}$	$\emptyset$
m	0.7	0	0.3	0
$m \oplus m$	0.91	0	0.09	0
$m \oplus m \oplus m$	0.973	0	0.027	0

TABLE I. COMBINATION OF A MASS FUNCTION WITH ITSELF.

attributes then become stable abstractions not directly thought in terms of sensors.

One major drawback of this rule is that it does not manage properly conflicting sources. Thus, the rule is not defined for sources in total conflict (*i.e.* K = 1) and can lead to counter-intuitive results with highly conflicting sources [33]. To prevent side effects of conflicting sources, it is possible to use methods to discount the sources of conflict [12] and there also exist other combination rules to manage conflict [27], [34], [35]. All the existing rules keep the generality of the Dempster's rule and enable the use of non-identified pieces of evidence.

One another risk with the Dempster's rule is that the pieces of evidence should be independent [36]. Thus, if one is using multiple times the same sensor type, for reliability reasons for instance, the gathered evidence should not be fused altogether with other type of sources using the BFT as they are reflecting the same physical event. As a matter of fact, if for example we use three motion sensors and one sound sensor, the three motion sensors should defend the same evidence in majority against the sound sensor, which in fact should not be seen as a majority but only as half the sensors. This is due to the fact that the Dempster's rule of combination is not idempotent (*i.e.* $m \oplus m \neq m$ if m is not the vacuous mass function). Thus, if the three motion sensors observe the same phenomenon and result in the same mass function, the fusion of those three mass functions converge where it should not (cf Table I) and results in an overly committed mass function.

To conclude this section on belief functions theory, we can say that this theory is very flexible and brings resiliency to our system. Unfortunately, it has some drawbacks. Firstly, the models, which require in the best case few experiments for each useful sensor for each context attribute, are dependent of bias and noise. As a consequence, it increases the engineering work required to reuse the models when the system is deployed in various environments. Secondly, the independence of sources required by the Dempster's rule of combination can be a major limitation if we want redundancy to overcome the loss of data we observe with wireless communications.

IV. VIRTUAL SENSORS

As seen in previous sections, the sensor measures may be imperfect for multiple reasons. The most annoying reasons when deploying a system are biases and noisy measures. It requires fine tuning each type the system is deployed in a new environment. In order to prevent from doing this work again and again at levels where models are hard to build, we decided to add a new sublayer to the sensing layer (cf Fig. 3): virtual sensors.

Instead of modifying high level models, we create sensor abstractions such as *motion sensor*, *sound sensor*, *temperature sensor*, etc. It is particularly convenient when working with typed data such as temperature or sound level. It is possible to use different brands of sensors for sensors of the same type. Thus, those sensors, even if they are measuring the same physical event, can return very different data due to their range, sensibility, voltage, etc. By creating abstraction of sensors, it is possible to build models directly from typed data simplifying even more the building of models as those data have are understandable by humans.

Those virtual sensors are built very simply from common heuristics and can be used for:

- *Bias and noise compensation*: as some sensors can be sensitive to the environment, with few experiments in a new environment, it is easy to observe the deviation of any sensor from its normal behavior and thus correct it properly.
- *Data aggregation*: sometimes, redundancy is convenient for reliability but also for reasons of sensor range (for example, a sensor motion has often a limited angle of vision making them unable to observe an entire room alone). However, as we have seen in section III-D, multiple sources observing the same physical event should not be mixed altogether with other sources. By doing simple heuristics such as averaging or getting the maximum/minimum, it is possible to create virtual sensors, seen by the upper layer as a single sensor, composed of multiple sources of the same type.
- *Meta-data generation*: when gathering data from sensors, it is possible to generate meta-data such as variation of the measures, average on last X measures, etc. Those generated data are also seen by the upper layer as sensor data.

It is also possible in these virtual sensors to implement fault and failure detection mechanisms using the BFT. It enables the detection of fault in the case of sensors of the same type. At higher level, those mechanisms will detect inconsistency between sensors of different types which is not of the same utility.

Thus, those virtual virtual sensors, without disabling any features in our architecture, bring more stability for our models. Moreover, by keeping the virtual sensors very simple, they are easy to adapt and tune in a new environment and the overhead in terms of computation is reduced to the minimum and does not really impact the global system performance. Finally, the fine tuning part is always reduced to this level of our architecture and nothing else has to be changed when we move the system from one environment to another.

The introduction of virtual sensors results in the modification of our multi-level architecture depicted in Fig. 5.

V. EXAMPLES OF VIRTUAL SENSORS

We deployed a prototype of smart home in two rooms (kitchen and living room) using phidgets sensors [37]. The sensors were connected to wireless nodes, communicating using a 6LowPAN/RPL network [38], such as Zolertia and Wismote nodes (see Fig. 6). We emulated augmented appliances with Android applications (see Fig. 7). The data fusion algorithms

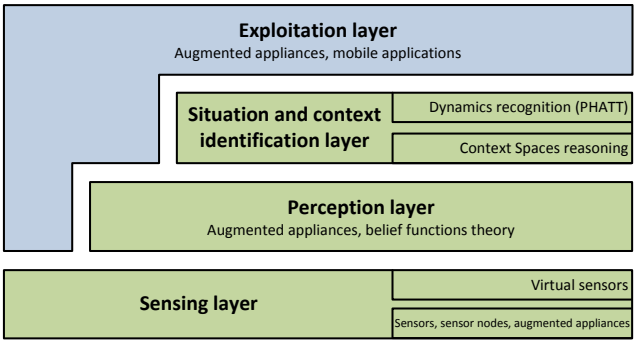

Fig. 5. Our multi-level context computing architecture (adapted from [1]) corrected with virtual sensors.

were running on PandaBoards[1] (ARMv7 1Ghz with 1GB of RAM) whose computational power was actually way higher than our real needs. That is why, we aim to use Raspberry Pi[2] (ARMv6 700Mhz with 512MB of RAM) instead.

The main context attribute we wanted to compute was the presence in the rooms. In order to do this, we placed diverse sensors: motion sensors, thin force and vibration sensors on chairs, contact sensors on windows, entrance sensors and CO2 sensors. Those sensors where placed in both rooms. Unfortunately, in order to observe the motion into those rooms, multiple motion sensors had to be deployed. Moreover, the "no motion detected" measure (corresponding to our zero on Fig. 4) was not exactly the same for each motion sensor. It could be due to different battery level as well as the electronic precision.

Thus, we created virtual sensors to correct those biases and merge the motion into a unique motion sensor. With motion sensors placed as depicted in Fig. 8, the corresponding virtual sensor had the following behavior:

- The two measures are received: the highest motion measure observed is returned. If no motion has been detected, then a "no motion" measure is returned.
- Only on measure is received: if motion has been detected, the measure is returned. If no motion has been observed, no measure is returned. As a matter of fact, the lost data could have correspond to a detected motion and there is no sufficient clue that no motion has been detected.
- No measure is received: no measure is returned.

With this virtual sensor adapted to each room, we were able to reuse the belief model created for the abstraction *motion sensor* applied to *presence* in all the rooms. It is particularly convenient as it is a lot more difficult to tune belief models than to tune bias correction.

Another virtual sensor that has been deployed was one to count the number of people seated on the chairs by simply applying a threshold in the weight detected by the thin force sensors and counting the number of persons detected. As it is possible to lose data in wireless communications, we were aware that the returned number of persons seated is in fact the minimum number of persons seated. As a matter of fact, if three persons are seated but the measures of only two sensors are received, the virtual sensor is able to return "there are at least two persons seated".

Finally, we deployed a virtual sensor to be able to exploit a CO2 sensor. This sensor is not reactive as it takes time for the CO2 level measured to vary significantly enough to deduce anything on the presence. As a matter of fact, the CO2 level increases slowly when someone enters the room and it continues to increase for a while after the last person leaves the

[1]http://pandaboard.org/
[2]http://www.raspberrypi.org/

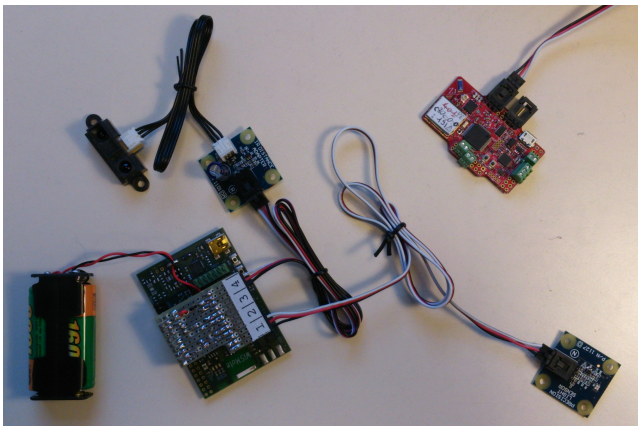

Fig. 6. Wireless nodes (Zolertia on the top right corner and Wismote on the bottom left corner) with connected phidgets sensors (here, a light sensor and an IR distance sensor).

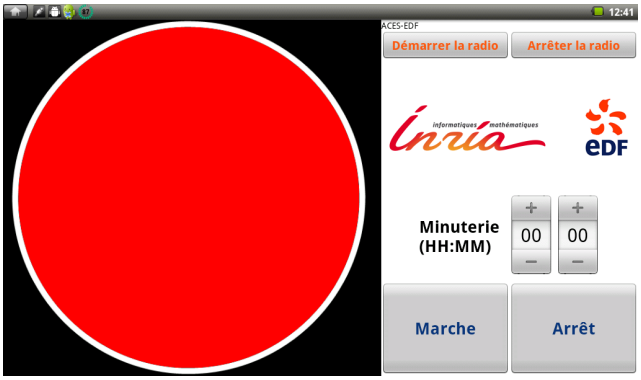

Fig. 7. Emulated augmented appliances: here, a hot plate and a radio.

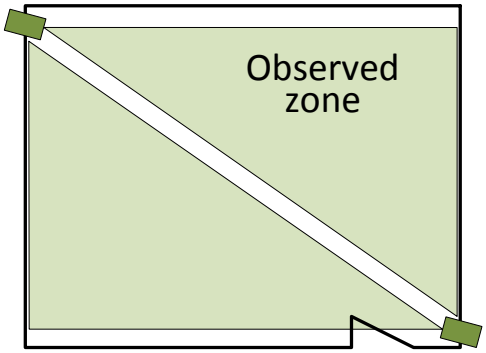

Fig. 8. Placement of motion sensors in a room. At least two sensors have to be installed to be able to observe the majority of the room.

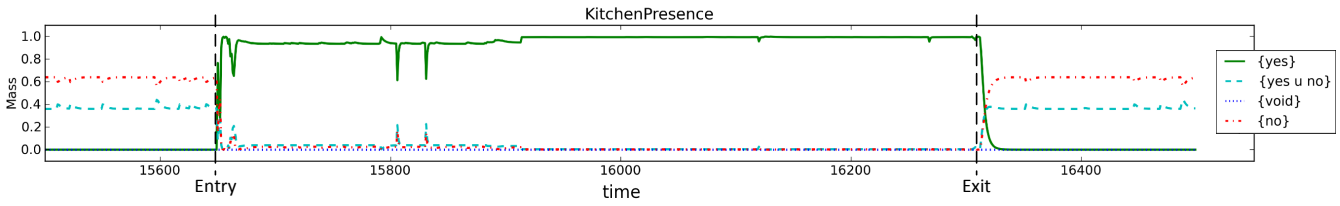

Fig. 9. Resulting mass for the detection of presence in the kitchen. It is obtained via the fusion of masses obtained with the measures of CO2 sensor, motion sensors, pressure and vibration sensors on chairs.

room. We thus used a sensor on the entrance to detect passage of people and infer on what should be the real measure of our virtual CO2 sensor.

Figure 9 depicts the resulting mass function for the detection of presence in the kitchen. The increasing and decreasing times of the mass at the beginning and in the end are due to temporization applied to the building of mass functions [24]. The spikes are due to loss of data and partly compensated by temporization. These results were obtained using the implementation of belief functions THE GAME[3].

While these examples may seem trivial, they were particularly convenient and accelerated a lot the deployment of our prototype. Moreover, the belief models created for this prototype will be usable in any future deployment with only virtual sensors to tune.

VI. Discussion

In this paper, we presented how a general purpose architecture for context computing could be adapted for smart homes using existing methods of data fusion and situation inference. Even if we are convinced that the methods we decided to use are adapted to the level of abstractions we require, we are aware that many other theories and methods may exist.

That is why we reason in terms of abstraction when building our models. Any layer could be replaced or patched using a theory without modifying the other layers. Moreover, the theories we use could be used at different levels without any problem. For example, if the plan recognition algorithm PHATT works initially with a very detailed tree of possible actions, it can certainly predict small pieces of context as well as future situations. If we constraint this algorithm to the upper part of our architecture, it is because we think that the prediction of situations is a lot more useful than the prediction of small actions in the case of smart home applications. Moreover, as we defended it, it is a matter of modeling ease.

For the BFT, it is actually possible to use it at many levels of abstraction. As a matter of fact, we use it to compute context attributes but it can also be used at very low level to fuse homogeneous sources and get a unique measure or it can be used at higher level for situation inference [5]–[8].

The architecture suggested by Coutaz *et al.* [1] is general and flexible. Thus, the methods we use to adapt it to our smart home applications is not the unique way of doing it. One is free to use any method or theory at any level depending on his or her criteria like ease of modeling, computation time or performance in context identification.

[3]Available at: https://github.com/bpietropaoli/THEGAME/

VII. Conclusion

In this paper, we presented an example of multi-level data fusion architecture used in a prototype of smart home with a specific focus on low-level data fusion techniques. The use of multi-level data fusion has many advantages:

- Simple modeling: by using different levels of abstractions, the models are closer to the way of thinking of humans. Moreover, those models require less engineering as they are stable from one deployment to another. It thus increases a lot the adaptability of the system and reduce the time of deployment.
- Cut complexity: each method used has a high complexity when treating a whole flow of raw measures. By using as input higher level abstractions, we reduce for the upper layers the number of processed data and thus the complexity.
- Each level of abstraction has its interest: in our architecture, each level of abstractions has its own time and physical scale. For instance, a situation is not detected as quickly as the presence in a room. Thus, each level of abstract has its own interest as a contextual data required by services. For example, a light automation service may not use the higher levels of abstraction as low-level abstraction such as presence may be sufficient and more reactive. Conversely, a heater automation service may be interested only in high level abstractions for more stability in time and because of inertia.

The paper focused with more details on low-level data fusion using the belief functions theory. We presented the basics of the belief functions theory and how it is possible to apply this theory with heterogeneous sensors to compute contextual data. This short presentation highlighted major drawbacks when trying to port models from one deployment to another. To respond to the problem of portability, we added a sublayer composed of virtual sensors enabling the abstraction of sensors. Even if the abstraction of sensor types may come in the future with standardization of sensors, for now, it has to be done manually in order to ease the engineering when deploying systems in multiple environments. By doing such abstraction, we reduce the *ad hoc* part required for each deployment to the lowest level possible.

ACKNOWLEDGMENT

The authors would like to thank EDF that has funded this research.

REFERENCES

[1] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan, "Context is key," *Commun. ACM*, vol. 48, pp. 49–53, March 2005.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[3] D. Dubois and H. Prade, "La problématique scientifique du traitement de l'information," IRIT, Université Paul Sabatier, Toulouse, Rapport de recherche 02-08R, mars 2002, bb.

[4] P. Smets, "Theories of uncertainty," in *Handbook of Fuzzy Computation*, I. Press, Ed. IOS Press, 1998, ch. Section B.1.2.

[5] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney, and S. Devlin, "Evidential fusion of sensor data for activity recognition in smart homes," *Pervasive and Mobile Computing*, vol. 5, no. 3, pp. 236 – 252, 2009, pervasive Health and Wellness Management.

[6] J. Liao, Y. Bi, and C. Nugent, "Activity recognition for smart homes using dempster-shafer theory of evidence based on a revised lattice structure," in *Proceedings of the 2010 Sixth International Conference on Intelligent Environments*, ser. IE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 46–51.

[7] S. McKeever, J. Ye, L. Coyle, and S. Dobson, "Using dempster-shafer theory of evidence for situation inference," in *Smart Sensing and Context*, ser. Lecture Notes in Computer Science, P. Barnaghi, K. Moessner, M. Presser, and S. Meissner, Eds. Springer Berlin / Heidelberg, 2009, vol. 5741, pp. 149–162.

[8] S. Mckeever, J. Ye, L. Coyle, C. Bleakley, and S. Dobson, "Activity recognition using temporal evidence theory," *J. Ambient Intell. Smart Environ.*, vol. 2, pp. 253–269, August 2010.

[9] A. Padovitz, "Context management and reasoning about situations in pervasive computing," Ph.D. dissertation, Monash University, Australia, 2006.

[10] C. W. Geib and R. P. Goldman, "Partial observability and probabilistic plan/goal recognition," in *Proceeding of the IJCAI workshop on Modeling Others from Observations (MOO)*, 2005.

[11] P. Chahuara, M. Vacher, and F. Portet, "Localisation d'habitant dans un environnement perceptif non visuel par propagation d'activation multisource," in *MAJECSTIC*, Bordeaux, France, 13-15 oct. 2010, p. 8pp.

[12] V. Ricquebourg, M. Delafosse, L. Delahoche, B. Marhic, A. Jolly-Desodt, and D. Menga, "Fault Detection by Combining Redundant Sensors: a Conflict Approach Within the TBM Framework," in *COGIS 2007, COGnitive systems with Interactive Sensors*. Stanford University, 2007.

[13] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 161–180, Apr. 2010.

[14] S. Carberry, "Techniques for plan recognition," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 31–48, Mar. 2001.

[15] R. P. Goldman, C. W. Geib, and C. A. Miller, "A New Model of Plan Recognition," *Artificial Intelligence*, vol. 64, pp. 53–79, 1999.

[16] M. La Placa, H. Pigot, and F. Kabanza, "Assistive planning for people with cognitive impairments," in *Proc. of Workshop on Intelligent Systems for Assisted Cognition hosted by Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[17] M. Ghallab, D. Nau, and P. Traverso, "Hierarchical task network planning," pp. 229 – 261, 2004.

[18] C. W. Geib and R. P. Goldman, "Plan recognition in intrusion detection systems," *DARPA Information Survivability Conference and Exposition,*, vol. 1, p. 0046, 2001.

[19] ——, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artificial Intelligence*, vol. 173, no. 11, pp. 1101 – 1132, 2009.

[20] A. Padovitz, A. Zaslavsky, and S. W. Loke, "A unifying model for representing and reasoning about context under uncertainty," in *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, July 2006.

[21] A. Padovitz, S. Loke, A. Zaslavsky, and B. Burg, "Verification of uncertain context based on a theory of context spaces," *International Journal of Pervasive Computing and Communications*, vol. 3, no. 1, pp. 30–56, 2007.

[22] A. Boytsov, A. Zaslavsky, and K. Synnes, "Extending context spaces theory by predicting run-time context," in *NEW2AN '09 and ruSMART '09: Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 8–21.

[23] A. Boytsov and A. Zaslavsky, "Extending context spaces theory by proactive adaptation," in *Smart Spaces and Next Generation Wired/Wireless Networking*, ser. Lecture Notes in Computer Science, S. Balandin, R. Dunaytsev, and Y. Koucheryavy, Eds. Springer Berlin / Heidelberg, 2010, vol. 6294, pp. 1–12.

[24] B. Pietropaoli, M. Dominici, and F. Weis, "Belief inference with timed evidence," in *Belief Functions: Theory and Applications*, ser. Advances in Intelligent and Soft Computing. Springer Berlin / Heidelberg, 2012, vol. 164, pp. 409–416.

[25] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Annals of Mathematical Statistics*, vol. 38, pp. 325–339, 1967.

[26] G. Shafer, *A Mathematical Theory of Evidence*. Princeton: Princeton University Press, 1976.

[27] E. Lefevre, O. Colot, and P. Vannoorenberghe, "Belief function combination and conflict management," *Information Fusion*, vol. 3, no. 2, pp. 149–162, 2002.

[28] P. Smets, "Analyzing the combination of conflicting belief functions," *Inf. Fusion*, vol. 8, no. 4, pp. 387–412, Oct. 2007.

[29] F. Delmotte, "Detection of defective sources in the setting of possibility theory," *Fuzzy Sets and Systems*, vol. 158, no. 5, pp. 555–571, 2007.

[30] B. Marhic, L. Delahoche, C. Solau, A. M. Jolly-Desodt, and V. Ricquebourg, "An evidential approach for detection of abnormal behaviour in the presence of unreliable sensors," *Inf. Fusion*, vol. 13, no. 2, pp. 146–160, Apr. 2012.

[31] M. Dominici, B. Pietropaoli, and F. Weis, "Experiences in managing uncertainty and ignorance in a lightly instrumented smart home," *Int. J. Pervasive Computing and Communications*, vol. 8, no. 3, pp. 225–249, 2012.

[32] A. Aregui and T. Denoeux, "Constructing consonant belief functions from sample data using confidence sets of pignistic probabilities," *International Journal of Approximate Reasoning*, vol. 49, pp. 575–594, November 2008.

[33] L. Zadeh, *On the Validity of Dempster's Rule of Combination of Evidence*, ser. Memorandum UCB/ERL-M. Electronics Research Laboratory, University of California, 1979.

[34] A. Martin, A. L. Jousselme, and C. Osswald, "Conflict measure for the discounting operation on belief functions," in *Information Fusion, 2008 11th International Conference on*, 2008, pp. 1–8.

[35] C. Osswald and A. Martin, "Understanding the large family of dempster-shafer theory's fusion operators - a decision-based measure," in *Information Fusion, 2006 9th International Conference on*, 2006, pp. 1–7.

[36] T. Denceux, "The cautious rule of combination for belief functions and some extensions," in *Information Fusion, 2006 9th International Conference on*, 2006, pp. 1–8.

[37] S. Greenberg and C. Fitchett, "Phidgets: easy development of physical interfaces through physical widgets," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*, ser. UIST '01. New York, NY, USA: ACM, 2001, pp. 209–218.

[38] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, Ltd, 2009.