Self-Understanding & Self-Extension: A Systems and Representational Approach

Jeremy L. Wyatt, Alper Aydemir, Michael Brenner, Marc Hanheide, Nick Hawes, Patric Jensfelt, Matej Kristan, Geert-Jan M. Kruijff, Pierre Lison, Andrzej Pronobis, Kristoffer Sjöö, Danijel Skočaj, Alen Vrečko, Hendrik Zender, Michael Zillich

***Abstract*—There are many different approaches to building a system that can engage in autonomous mental development. In this paper we present an approach based on what we term *self-understanding*, by which we mean the use of explicit representation of and reasoning about what a system does and doesn't know, and how that understanding changes under action. We present a coherent architecture and a set of representations used in two robot systems that exhibit a limited degree of autonomous mental development, what we term *self-extension*. The contributions include: representations of gaps and uncertainty for specific kinds of knowledge, and a motivational and planning system for setting and achieving learning goals.**

***Index Terms*—robotics, robot learning, architectures, representations**

I. Introduction

WHAT is needed for an agent to learn in a truly autonomous fashion? One way is to give that agent knowledge of what it knows and doesn't know, and to make it reason with these representations to set its own *epistemic goals*. An epistemic goal is a goal to be in a certain knowledge state. In this paper we describe this representation and systems approach to autonomous mental development. We present an architecture, together with a set of representations that explicitly capture what the robot and other agents do and don't know at any one time, i.e. representations of their epistemic state. We also describe representations of how this epistemic state will change under action. Such representations with algorithms for reasoning about them we refer to as conferring a degree of *self-understanding*, and allow the construction of systems that are able to plan how to extend the knowledge they have of the environment, i.e. knowledge *self-extension*. We also describe a goal management system that allows the robot to choose quickly between different epistemic goals. We argue that such an approach will be necessary in the long term as robot systems become able to generate many goals for filling gaps in and reducing uncertainty in knowledge.

Jeremy L. Wyatt, Marc Hanheide and Nick Hawes are with the University of Birmingham, email: {jlw,nah,m.hanheide}@cs.bham.ac.uk

Michael Brenner is with Albert-Ludwigs-Universität, email: brenner@informatik.uni-freiburg.de

Pierre Lison, Geert-Jan M. Kruijff and Hendrik Zender are with DFKI, Saarbrücken, Germany, email: {plison,gj,zender}@dfki.de

Patric Jensfelt, Andrzej Pronobis, Kristoffer Sjöö and Alper Aydemir are with KTH Stockholm, email: {patric,krsj,pronobis,aydemir}@csc.kth.se

Matej Kristan, Alen Vrečko and Danijel Skočaj are with University of Ljubljana, email: {matej.kristan,alen.vrecko,danijel.skocaj}@fri.uni-lj.si

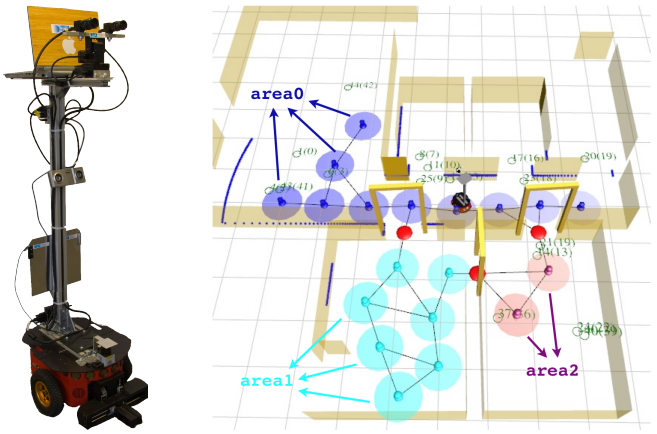
Michael Zillich is with Vienna University of Technology, email: zillich@acin.tuwien.ac.at

Manuscript received February 28, 2010

It is important to understand a little of the different types of incompleteness in knowledge. We use *incompleteness* as an umbrella term to cover many different types of *knowledge gaps* and *uncertainty about knowledge*. We can think about a typology of incompleteness in knowledge based on three dimensions of variability. These are *the nature of the incompleteness*, the *type of knowledge that is incomplete*, and whether the incompleteness is represented in a *quantitative or qualitative* manner.

With regard to the nature of the incompleteness, in the simplest case we may have a variable or variables that have a defined set of possible values or hypotheses from which the true value is known to be drawn. We refer to this as *simple uncertainty*. We can also have *uncertainty about the number of variables* needed in a model, i.e. about the model complexity. Finally we can also have cases where the agent knows that a variable is of an unexperienced class, i.e. there is *novelty*. This can include cases where the variables are continuous but where the observation models for a class are quite confident and do not generalise well to some new observation. The type of knowledge that is incomplete may vary enormously. Four simple types that cover a variety of cases include contingent knowledge about the current world *state*, *structural* knowledge about the universal relationships between variables, knowledge consisting of *predictions* of action outcomes or events, and knowledge about their *causes*. Finally there is a question about whether the representation is qualitative or quantitative. In qualitative representations of gaps or uncertainty we have a set of possible values for the variable, or a statement that the variable value is unknown, or knowledge that there may be many variables that are unmodelled. In quantitative representations we will have some kind of scalar values attached to hypotheses (e.g. is this a cup or mug) or statements (such as whether there is novelty or not), and in our case these will typically be probabilities. Note that by a *quantitative gap* or *quantitative uncertainty* we do not mean that the underlying space for the variable is continuous or discrete, but instead that the way the incompleteness is represented involves an expression of preference for one hypothesis or statement versus another.

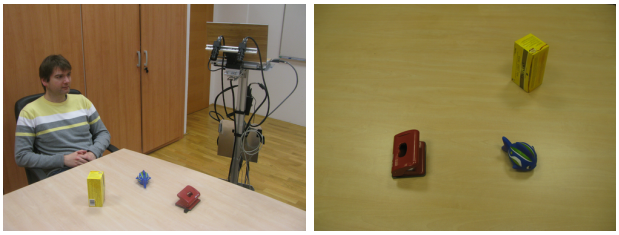
In this paper we deal with filling qualitative gaps, qualitative uncertainty in state, quantitative uncertainty about structural knowledge, and novel states. We provide empirical proof that our approach works and illustrate different aspects of it through two robot systems we have implemented. We call these Dora, and George (Figs. 1 and 2). We provide links to



(a) The Dora platform: a P3 mobile robot base with a custom-built superstructure and different sensors.

(b) Visualisation of Dora's map of a partially explored environment. Coloured disks denote *place* nodes (colour indicates segmentation into different rooms, `area0,1,2`). Small green circles represent opportunities for spatial exploration (*placeholders*). Red nodes indicate places where doorways are located.

Fig. 1. Dora the Explorer: a robot system for goal-driven spatial exploration.



(a) Scenario setup. (b) Observed scene.

Fig. 2. George scenario: continuous interactive learning of visual properties.

videos of these systems running in the real world[1], and analyse their behaviour on larger amounts of data using simulations. We now describe Dora and George in more detail to give concrete examples that will run through the paper.

Dora is able to explore an office environment, choosing between filling different types of incompleteness in her maps of the environment. Dora illustrates the architecture, the representations of gaps in spatial knowledge, the use of epistemic states in goal setting and action planning, and in the use of our motivation system. In Dora's case the current incomplete knowledge she can model and fill can be seen by reference to Fig. 1(b). Here we can see a visualisation of a map that Dora has built after a partial tour of an office environment. The map consists of a graph where the nodes (which we call *places*) are partitioned into areas by landmarks, such as doors. Dora has representations of two kinds of incompleteness. She represents unexplored regions of space, by maintaining a set of hypothesised places, which we call *placeholders*. These are depicted in Fig. 1(b) as small unfilled circles with numeric labels. This is uncertainty about how many variables are needed to model the space. Second, Dora has the ability to categorise areas into categories such as office, kitchen, coffee room and corridor. In Fig. 1(b) it can be seen that none of the areas currently have known categories. This is simple state uncertainty, as Dora knows a certain number of types of area, and cannot represent or reason about novel area types. During the autonomous part of the mapping process Dora will choose the order in which to reduce these different kinds of incompleteness. To map unexplored areas by adding nodes to the topological map she will conduct laser based mapping while visiting the hypothesised placeholders. To categorise a room she will search for objects that indicate its category, e.g. kitchens typically contain objects such as milk cartons and cups, and offices objects such as bookshelves and journals.

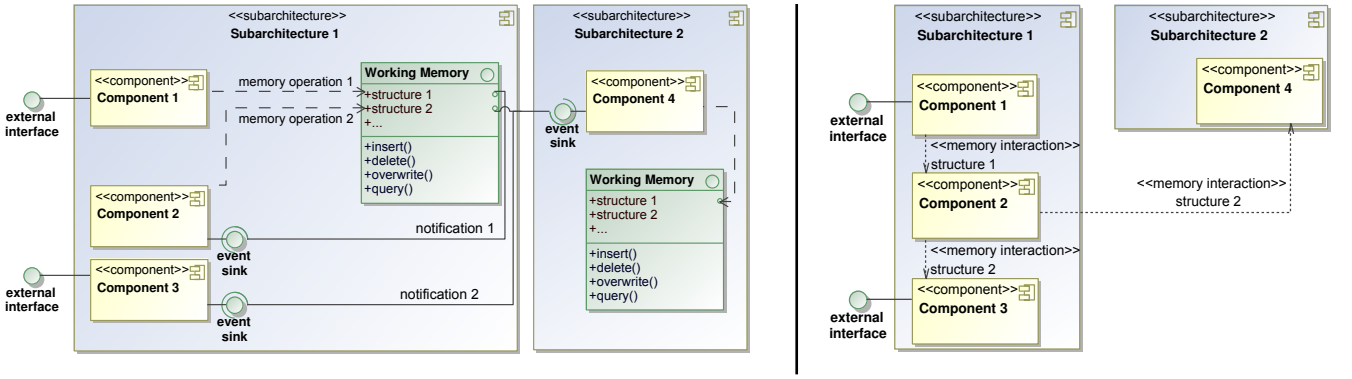
George is a system that converses with a human to reduce incompleteness it has about the properties of objects on a table top. George illustrates the way we represent uncertainty and incompleteness in models of the structural relationships between visual information and linguistic descriptions of objects. What visual properties, for example, make an object round or square, vs. red or yellow? George has representations of the uncertainty it has as to which sub-spaces of a set of visual features are associated with particular adjectives. This is a type of structural uncertainty. George can learn from a tutor, but crucially he can also decide which questions to ask in order to fill a particular gap he has identified. A typical scene during George's learning is depicted in Fig. 2. A typical dialogue snippet might be:

G: Which colour is the elongated object?
H: The elongated object is yellow.
G: OK.
G: Is the square object blue?
H: No it is not blue. It is red.
G: OK.

During this dialogue the robot and the human reason about each others beliefs, what they know and don't know, and how to establish common understanding. This is type of state uncertainty, since the robot can only model the human as having one of a known set of beliefs. In the dialogue each agent makes references to objects that they understand will be distinguishing to the other agent, such as referring to the elongated object. More importantly George asks questions which are prompted by detection of gaps such as state novelty. He asks: *"Which colour is ..?"* when he realises that the colour is one he hasn't experienced before. When he is simply uncertain about which of a number of colour classes is present he asks instead whether the object has the most likely colour class: *Is the object blue?*. Both George and Dora also have mechanisms for doing non-monotonic inference or learning. George can unlearn erroneous representations of colour and shape, and in Dora's case she can withdraw support for inferences about room category, or the partition of her map.

The rest of the paper is structured as follows. In Section II we describe the architectural model. Section III describes the model that connects information from multiple modalities, and how we have engineered those representations to explicitly capture uncertainty and incompleteness in the amodal model. In Section IV covers representations of space, cross-modal relations, epistemic goals, and epistemic action effects, all as used in Dora and/or George. In Section V we describe our approach to goal management, and finally in Sections VI and VII we describe the Dora and George systems and present an experimental analysis their behaviour.

[1]Available at `http://cogx.eu/`



(a) In CAS, there are components, which run in parallel, asynchronously updating shared structures on a common working memory. They can also take input from sensors or give output to actuators. Subarchitectures are coupled to make an overall system.

(b) A simplified illustration of the interaction patterns mediated through the working memories focusing on information flow from sources to sinks.

Fig. 3. Building systems with CAS. The form on the right is used as a short hand in the later system diagram for Dora.

II. An Architecture for Multi-Modal Processing

In this section we describe the basic architecture we employ, which we call CAS (CoSy Architecture Schema) [1]. We refer to it as a schema because it actually defines a space of specific architectures, we refer to the schema when talk about this space, and to an architecture when we mean a specific architecture employed in a particular robot system. The schema is essentially a distributed working memory model, where representations are linked within and across the working memories, and are updated asynchronously and in parallel. The key idea is that it replaces a single world model (still prevalent in robotics) with multiple, linked world models, enabling it to work in the face of inconsistent evidence, uncertainty and change. The decomposition into working memories groups processes that commonly share information, and is typically by sensory modality. So that in Dora and George we build separate subsystems (called *subarchitectures*) for vision, communication and spatial understanding. As we shall see in Sections III and IV each subarchitecture contains representations which explicitly capture uncertainty and incompleteness. The system overall can reason about this uncertainty or incompleteness and plan how to act so as to fill that knoweldge gap, perhaps by employing information in another modality. We now describe the key aspects of CAS relevant to this paper.

A. Subarchitecture Design

1) Components: Our schema starts on the level of a collection of processing components (Fig. 3(a)). Every component is concurrently active, allowing them to process in parallel. We do not specify any constraints on the contents of components: they could have behave like a node in a connectionist network, an activity in a behaviour-based system, or an entire function in a functionally-composed system. Components can take input either directly from sensors, or from working memory. They can also directly control actuators in the manner of closed loop controllers, or initiate fixed action patterns. Components can have processing triggered by the appearance of certain information on the shared working memory, and can modify structures on that memory. Components may also have their own private (i.e. component-internal) memory. Components are typically designed around two archetypes: managed and unmanaged. *Unmanaged components* are low-latency processes that run all the time, regardless of overall system state. *Managed components* by contrast are typically computationally expensive processes, which only run when there is a demand for their services. These components are only run when a particular configuration of information is present on working memory.

2) Shared Working Memories: Rather than exchange information directly, processing components are connected to a *shared working memory* (Fig. 3(a)). The contents of the working memory are solely composed of the outputs of processing components. Each working memory is connected to all other working memories in the system. This allows components to exchange information across subarchitectures. In our implementation of CAS the communication method between the working memory and the components determines the efficiency of the model. But for now let us consider simply that the schema itself allows reading and writing to working memories, and transfer of information between them.

This use of shared working memories is particularly well suited to the *collaborative refinement of shared structures*. In this approach to information processing, a number of components use the data available to them to incrementally update an entry on working memory. In this manner the results of processing done by one component can restrict the processing options available to the others in an informed way. As all components are active in parallel, the collective total processing effort (i.e. the amount of work done by all the components in solving a problem) may be reduced by sharing information in this way. This feature turns out to be a very powerful aspect of the schema.

B. System Design Practices with CAS

While a system could be composed of a single subarchitecture, we intend that there should typically be several subarchitectures in operation. In the integrated systems we describe in this paper we have about four subarchitectures. The architecture makes no assumptions about whether system decomposition should be predominantly according to behaviour

or information processing function. What is important is that the decomposition groups components that commonly exchange information via shared structures. One of the main benefits of having distributed working memories is that the working memory can act as a filter for its local components, only letting them become aware of events elsewhere in the system when necessary.

In the systems described here (Dora and George) we have separate subarchitectures for vision, linguistic communication, and spatial understanding. To have coherent global action, however, the system benefits from linking these separate models. In other words there are data structures, or symbols on one blackboard, that are related to those on another. For example the visual system might have a symbol corresponding to a blue cup sitting on a table, and the communication system might have a symbol corresponding to a mention of a blue thing by a human. To make sense of the human's words the robot has to decide whether these two symbols are connected, or whether the human is referring to some other object (perhaps another blue object in the scene). An important question is what mechanisms can be used to link these efficiently? We call this symbol linking problem the *binding problem* and we describe it in the next section. In particular we will talk about how we can solve the binding problem in a way that allows us to represent uncertainty in which symbols should be bound to which, and also gaps in the robot's overall picture of the world right now. Binding essentially involves creating new symbols that refer back to each of the modality specific symbols. We refer to the representations that are created by binding as multi-modal representations. At the highest level of abstraction, however, binding produces an essentially amodal model of the robot's world.

III. Modelling multi-modal beliefs

So far we have described an architecture capable of supporting processing on groups of modality specific representations. High-level cognitive capabilities must generally operate on high level (i.e. abstract) representations that collect information from multiple modalities. This requirement raises the double issue of (1) how these high-level representations can be reliably generated from low-level sensory data, and (2) how information arising from different subsystems can be efficiently fused into unified multi-modal structures.

We present here a new approach to multi-modal information binding [2], [3], based on a Bayesian framework. The approach is implemented in a specific subarchitecture in our systems called the *binder* [4]. The binder is directly connected to all subsystems in the architecture. It serves as a central hub for the information gathered about entities currently perceived in the environment. The data structures included in the binder are inherently probabilistic. Each property or information bit pertaining to an entity is given a probability value, reflecting the confidence level of the subsystem. This enables the system to deal with varying levels of noise and uncertainty, which are unavoidable for most sensory-motor processes.

Based on the data structures made available in this repository, the binding algorithm seeks to "merge" or unify the perceptual inputs arising from the various subsystems, by checking whether their respective features correlate with each other. The probability of these correlations are encoded in a Bayesian network. This Bayesian network might for instance express a high compatibility between the haptic feature "shape: cylindrical" and the visual feature "object: mug" (since most mugs are cylindrical), but a very low compatibility between the features "shape: cylindrical" and "object: ball".

The resulting multi-modal information structure is called a *belief* in our terminology. The task of the binder is to decide which proxies from different modalities belong to the same real-world entity, and should therefore be merged into a belief. The outcome of this process is a joint probability distribution over possible beliefs. These beliefs integrate in a compact representation of all the information included in the perceptual inputs. They can therefore be directly used by the deliberative processes for planning, reasoning and learning.

A. Representations

The three central data structures manipulated by the binder are **proxies**, **unions** and **beliefs** (also see Fig. 4(a)).

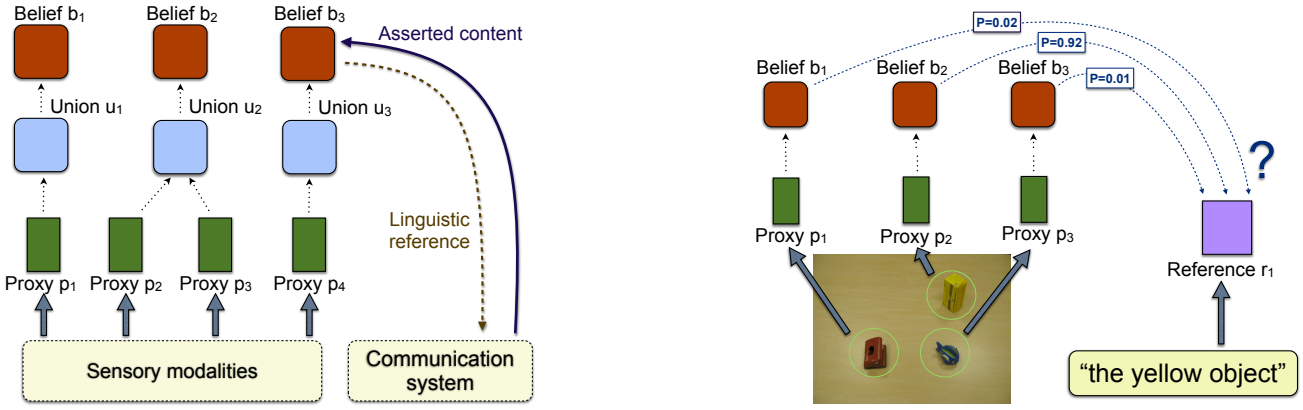
1) Proxies: A mid-level, uni-modal representation of a given entity in the environment. Proxies are inserted onto the binder by the various subsystems included in the architecture.

A proxy is essentially defined as a multivariate probabilistic distribution over a set of features. The distributions included in the proxy can be either discrete (as for categorical knowledge) or continuous (as for real-valued measures).

2) Unions: A mid-level, multi-modal representation of an entity, constructed by merging one or more proxies. Just like proxies, unions are also represented as a multivariate probabilistic distribution over possible features. Unions are essentially a transitional layer between proxies and beliefs.

3) Beliefs: A high-level, amodal representation of an entity in the environment. Beliefs are generally build on top of unions, but they are expressed in an amodal format and encode additional information related to the specific situation and perspective in which the belief was formed, such as its *spatio-temporal frame*, its *epistemic status* and its *saliency value*:

- The spatio-temporal frame is defined according to a spatial model (set of possible "places" in the environment), a temporal model (points on a continuous temporal interval), and possibly a perspective on these two models from the viewpoint of a particular agent.
- The epistemic status of a belief (or subpart of a belief) can be either private, attributed or shared. Private beliefs are beliefs which are internal to the agent, while attributed beliefs are beliefs an agent ascribes to another agent (e.g. A believes that B believes X). Shared beliefs are beliefs which are part of the common ground for all agents.
- Finally, the salience is a multivariate density function $\Re^n \rightarrow [0, 1]$, where each variable defines a particular, real-valued saliency measure. It provides an estimate of the "importance" or quality of standing out of a particular entity relative to neighboring ones [5]. The salience is used to drive the attentional behaviour of the agent by specifying which entitites are currently in focus.



(a) Construction of multi-modal beliefs.

(b) Reference resolution for the expression "the yellow object".

Fig. 4. Multi-modal information binding: belief construction (left) and application in a reference resolution task (right).

Beliefs are indexed via an unique identifier, which allows us to keep track of the whole development history of a particular belief. Beliefs can also be connected with each other using relational structures of arbitrary complexity.

To account for this rich representation, beliefs are formalised according to a *belief model*, which is a mathematical structure defining a space of possible belief instances.

B. Binding algorithm

To be able to create beliefs out of proxies, the binder must decide for each pair of proxies arising from distinct subsystems, whether they should be bound into a single union, or fork in two separate unions. The decision algorithm for this task is based on a well-known technique from probabilistic data fusion, called the *Independent Likelihood Pool* (ILP) [6]. Using the ILP, we are able to compute the likelihood of every possible binding of proxies, and use this estimate as a basis for constructing the beliefs. The multivariate probability distribution contained in the belief is a linear function of the feature distributions included in the proxies and the correlations between these.

A Bayesian network encodes all possible feature correlations as conditional dependencies. The encoded features may be discrete or continuous. The (normalised) product of these correlations over the complete feature set provides an useful estimate of the "internal consistency" of the constructed belief – a belief with incompatible features will have a near-zero probability, while a belief with highly correlated features will be associated with a high probability.

C. Referencing and updating beliefs

The beliefs are high-level symbolic representations available for the whole cognitive architecture. As such, they provide an unified model of the environment which can be efficiently used when interacting with the human user. An important aspect of this is *reference resolution*: how to connect linguistic expressions such as "this box" or "the ball on the floor" to the corresponding beliefs about entities in the environment.

Reference resolution is performed using the same core mechanisms as for binding – a Bayesian network specifies the correlations between the linguistic constraints of the referring expressions and the belief features (in particular, the entity saliency and associated categorical knowledge). The resolution process yields a probability distribution over alternative referents (see Fig. 4(b) for an example), which is then retrieved by the communication subsystem for further interpretation.

In addition to simple reference, the interaction with a human user can also provide *new* content to the beliefs, as in cross-modal learning scenarios. Via (linguistic) communication, the human user can thus directly extend or modify the robot's current beliefs, in a top-down manner, without altering the incoming proxies. If this new information conflicts with existing perceptual knowledge, the agent can decide to trigger a clarification request to resolve the conflict.

An utterance such as "This is yellow" illustrates these two complementary mechanisms. First, the linguistic expression "this" must be resolved to a particular entity in the environment. Since "this" is a (proximal) deictic, the resolution is performed on basis of the saliency measures. In the absence of any other constraint, the most salient entity is simply selected and retrieved. Second, the utterance not only refers to an existing entity in the environment, but it also provides new information about it – namely that is is yellow. This asserted content must therefore inserted into the robot's beliefs. This is realised by selecting the belief pertaining to the referred-to entity and incorporating the new, attributed information into its content representation.

D. Implementation

The outlined approach has been fully implemented as a separate subsystem in the cognitive architecture. It includes a central working memory where proxies can be inserted, modified or deleted. The belief set is automatically updated to reflect the incoming information. A GUI allows the user to monitor at runtime the binder behaviour.

The Bayesian network encoding the feature correlations can be either manually specified, or learned using various machine learning techniques (see section VII).

IV. Representations to support self-extension

This section explains how different domains of knowledge are represented in our system. We present three types of representations: representations of space, primarily related to the Dora scenario; cross-modal representations used in the George scenario; and representations of epistemic state and action effects used by planning. Each representation focuses on structuring and abstracting what is known, but also on representing uncertainty and knowledge gaps explicitly.

A. Representations of space

Spatial knowledge constitutes a fundamental component of the knowledge base of a mobile agent, such as Dora, and many functionalities directly depend on the structure of the spatial knowledge representation. These include spatial localisation, navigation, wayfinding and autonomous exploration, but also understanding and exploiting semantics associated with space, human-like conceptualisation and categorisation of and reasoning about spatial units and their relations, human-robot communication, action planning, object finding and visual servoing, and finally storing and recalling episodic memories.

In our system, spatial knowledge is represented in multiple layers, at different levels of abstraction, from low-level sensory input to high level conceptual symbols. Moreover, continuous space is discretised into a finite number of spatial units. The abstraction and discretisation process is one of the most important steps in representing spatial knowledge as it allows the representation to be made compact, tractable and robust to changes that occur in the world. Discretisation drastically reduces the number of states that have to be considered, e.g. during the planning process, and serves as a basis for higher level conceptualisation.

The representation is designed for representing complex, cross-modal, spatial knowledge that is inherently uncertain and dynamic. Our primary assumption is that spatial knowledge should be represented only as accurately as it is required to provide all the necessary functionality of the system. This keeps the complexity of the representation under control, makes the knowledge more robust to dynamic changes and substantially reduces the effort required to synchronise the representation with the environment. Additionally, uncertainties are associated with the represented symbols and gaps in spatial knowledge are explicitly modelled.

Fig. 5 gives a general overview of the structure of the representation. It is sub-divided into layers of specific representations. We distinguish between four layers which focus on different aspects of the world, abstraction levels of the spatial knowledge and different spatial scales. Moreover, each layer defines its own spatial entities and the way the agent's position in the world is represented. At the lowest abstraction level we have the sensory layer which maintains an accurate representation of the robot's immediate environment extracted directly from the robot's sensory input. Higher, we have the place and categorical layers. The place layer provides fundamental discretisation of the continuous space explored by the robot into a set of distinct places. The categorical layer focuses on low-level, long-term categorical models of

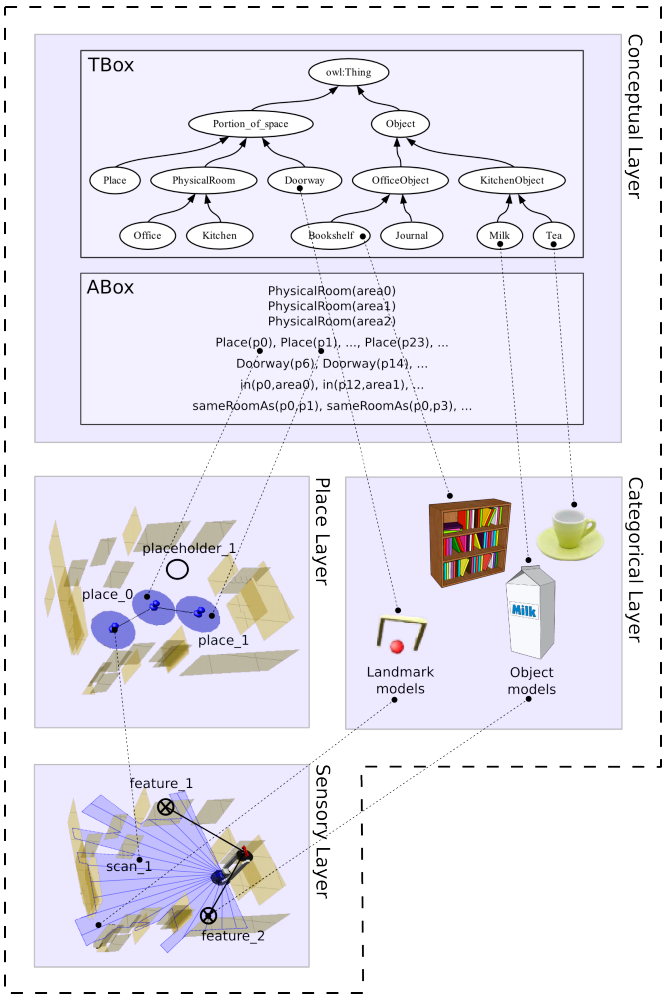


Fig. 5. The layered structure of the spatial representation. The position of each layer within the representation corresponds to the level of abstraction of the spatial knowledge. The ABox in the conceptual layer corresponds to the example in Fig. 1(b) on page 2.

the robot's sensory information. Finally, at the top, we have the conceptual layer, which associates human concepts (e.g., object or room category) with the categorical models in the categorical layer and groups places into human-compatible spatial segments such as rooms.

The following paragraphs provide additional details about each of the layers and their instantiations within our system. The system provides only an initial instantiation of the representation that validates correctness and usefulness of the knowledge structure within an integrated cognitive system. At the same time, as it will be mentioned below, some of the underlying algorithms do not adhere fully to the principles behind the representation. For a detailed theoretical discussion on those principles and optimal implementations, we refer the reader to [7].

1) Sensory Layer: In the sensory layer, a detailed model of the robot's immediate environment is represented based on direct sensory input as well as data fusion over space around the robot. The sensory layer stores low-level features and landmarks extracted from the sensory input together with their exact position. Measures of uncertainty are also included in this representation. Landmarks beyond a certain distance

are forgotten and replaced by new information. Thus, the representation in the sensory layer is akin to a sliding window with robocentric and up-to-date direct perceptual information.

The representation in the sensory layer helps to maintain stable and accurate information about the robot's relative movements. Moreover, it allows for maintaining and tracking the position of various features while they are nearby. Finally, the sensory layer provides the low level robotic movement systems with data for deriving basic control laws, e.g., for obstacle avoidance or visual servoing.

In the current implementation, the sensory layer is maintained by two subsystems: a metric SLAM algorithm [8] that builds a global metric map of the environment and an Active Visual Search (AVS) component which represents hypotheses about objects found in the environment using a local grid map. The SLAM algorithm explicitly represents the uncertainty associated with the pose of the robot and the location of all landmarks using a multivariate Gaussian distribution encoded using a state vector and a covariance matrix [9], [8]. At the same time, the AVS component maintains hypotheses about existence of an object of a specific category at a specific location using a probabilistic grid representation [10]. The probabilistic grid representation is formed from multiple cues about object location one of which is the presence of obstacles in the SLAM map. This is the prior on which basis the AVS algorithm determines the next best viewpoint based on a randomized art-gallery algorithm [11].

The existence of the global metric map violates some of the assumptions behind the proposed representation; however, it is only used internally. In future instantiations, the allocentric SLAM algorithm will be replaced by a robocentric method [12], [13], [14]. Here, in order to verify the correctness of such concept, we restrict access to the metric map from other components of the system, exposing only local and relative (with respect to the robot) metric information – with the exception of the navigation system that still uses the allocentric SLAM algorithm.

2) Place Layer: The place layer is responsible for the fundamental, bottom-up discretisation of continuous space. In the place layer, the world is represented as a collection of basic spatial entities called *places* as well as their spatial relations. The aim of this representation is not to represent the world as accurately as possible, but at the level of accuracy sufficient for performing required actions and robust localisation despite uncertainty and dynamic variations.

Besides places, the place layer also defines paths between them. The semantic significance of a path between two places is the possibility of moving directly between one and the other. In addition, the place layer explicitly represents *gaps in knowledge about explored space*. Space that has not yet been explored by the robot has no places in it. Therefore, tentative places are generated, which the robot would probably uncover if it moved in a certain direction. These hypothetical places allow for reasoning about unknown space, and for planning and executing exploratory activities. They are annotated as *placeholders* to keep them apart from ordinary, actual places, but are otherwise identically represented and interconnected. For an illustrative example of several places and placeholders identified during spatial exploration, see Fig. 1(b) on page 2.

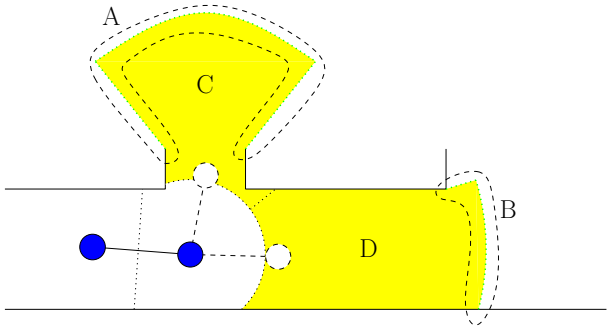


Fig. 6. Placeholder creation. Dashed circles are hypotheses, each representing one placeholder. A and B are frontier length estimates, C and D are coverage estimates for the respective placeholders.

Two quantitative measures are associated with each placeholder providing an estimate of information gain related to each exploration task. These are used by the motivation system, described later in Section V on page 11. The measures used are the *coverage estimate* (CE) and the *frontier length estimate* (FLE), cf. Fig. 6. The former is obtained by measuring the free space visible from the current node and not near to any existing node, and assigning it to the closest hypothesis. This heuristically estimates the number of new places that would result from exploring that direction. The FLE is analogously extracted from the length of the border to unknown space. By prioritising these two measures differently, the motivation mechanism can produce different exploratory behaviours.

3) Categorical Layer: The categorical layer contains long-term, low-level representations of categorical models of the robot's sensory information. The knowledge represented in this layer is not specific to any particular location in the environment. Instead, it represents a general long-term knowledge about the world at the sensory level. For instance, this is the layer where models of landmarks or objects are defined in terms of low-level features. The position of this layer in the spatial representation reflects the assumption that the ability to categorise and group sensory observations is a fundamental one and can be performed in a feed-forward manner without any need for higher-level feedback from cognitive processes.

The categorical models stored in this layer give rise to concepts utilised by higher-level layers. In many cases complex models are required that can only be inferred from training data samples. In case of models that correspond to human concepts, they can be learnt in a supervised fashion, using a top-down supervision signal.

In our system, the categorical layer was realised through visual categorical models of objects employed by the AVS component and a simple door detection algorithm used as a landmark model. The AVS component uses the object recognition method proposed in [15] and the models associated with object classes reside in the categorical layer. However, using only this algorithm does not provide the pose of objects nor the uncertainty associated with it and is not robust to cases where two objects appear similar from a certain viewpoint. Therefore, a natural extension to this procedure which estimates the pose and class of objects is also implemented [10]. Additionally, in our experiments, we employed appearance and geometry-

based models of place categories [16]. Although currently not being used in the Dora scenario, those models constitute a part of the categorical layer.

4) Conceptual Layer: The conceptual layer provides a symbolic ontological representation of space that makes use of human-compatible concepts. The taxonomy of the spatial concepts and properties of spatial entities, as well as the instances of these concepts are linked to the lower levels of the spatial model. This associates semantic interpretations with the low-level models and can be used to specify which properties are meaningful, e.g., from the point of view of human-robot interaction. The main purpose of the conceptual layer is to represent a segmentation of the environment into rooms. Moreover it provides human-compatible concepts for these rooms based on the objects they contain, and it can supply default assumptions about which kinds of objects are likely to be found in which kinds of rooms.

The representation underlying the conceptual map is an OWL-DL ontology[2], consisting of a taxonomy of concepts (*TBox*) and the knowledge about individuals in the domain (*ABox*), cf. Fig. 5 on page 6, cf. [17]. Here is an example of a *concept definition* in the current implementation which defines a kitchen as a room that contains at least two typical objects:

$$\mathsf{Kitchen} \equiv \mathsf{Room} \sqcap \geq 2\mathsf{contains}.\mathsf{KitchenObject}$$

Besides the usual inferences performed by the OWL-DL reasoner, namely *subsumption checking* for concepts in the TBox (i.e., establishing subclass/superclass relations between concepts) and *instance checking* for ABox members (i.e., inferring which concepts an individual instantiates), an additional *rule engine* is used to maintain a symbolic model of space under *incomplete* and *changing* information.

The discrete places from the place layer and their adjacency are the main pieces of knowledge that constitute the input for that reasoning. One, it maintains a representation that groups places into rooms. Furthermore, using observations (visually detected objects, appearance- and geometry-based room categories) it can infer human-compatible concepts for a room, and raise expectations about which other kinds of objects are prototypically likely to be present. The ongoing construction of the conceptual map is potentially nonmonotonic. The overall room organisation may be revised on the basis of new observations. The further association between room concepts and salient, proto-typical object types is established through the "locations" table of the OpenMind Indoor Common Sense[3] database by Honda Research Institute USA Inc.

In the current implementation, the conceptual layer can be used to determine *knowledge gaps in the categorisation of rooms.* It is considered a gap in knowledge if for a given room (i.e., an instance of PhysicalRoom) its basic level category is unknown. This is assumed to be the case if no more specific concept than PhysicalRoom (i.e., Office or Kitchen, cf. Fig. 5 on page 6) can be inferred for the individual. This knowledge gap persists until the robot has gathered enough evidence (i.e., contained objects) for inferring a subconcept.

[2]http://www.w3.org/TR/owl-guide/
[3]http://openmind.hri-us.com/

B. Representations of epistemic state and action effects

Decisions about what actions to perform next are not pre-programmed in our robot, but are made by the *planning* subarchitecture. In this section, we describe how knowledge and knowledge-producing actions are modelled such that the planner can reason about how knowledge gaps can be filled. Planning systems traditionally use representations based on propositional logic. Most notably, the classic STRIPS formalism and its modern descendent PDDL are based on such a propositional representation. The representation we use for the planning system on our robot, however, is based on the SAS$^+$ formalism [18]. Here, instead of propositions, we use multi-valued state variables (MVSVs) v, each with an associated domain $vdom(v)$ describing the set of possible values $x \in vdom(v)$ that v may assume. A state is a function s associating variables with values from their domain. In recent years, SAS$^+$ has been shown to enable powerful reasoning techniques in planning algorithms and systems based on SAS$^+$ now dominate the International Planning Competition. For the modelling needs of our robot applications, we have developed the SAS$^+$-based modelling language MAPL [19].

For robotic planning, MAPL provides, in addition to the computational advantages, several representational ones. Firstly, it stays close to the feature/value model used by other subarchitectures of our robot. In particular, the mapping between binder states and planning states is greatly simplified: Roughly, each feature f of a union u in a belief model is mapped onto a state variable $f(u)$. For example, if the belief model describes that a room has been categorised as a kitchen by attributing the feature *areaclass : kitchen* to a union u, this would correspond to an assignment *areaclass(u) = kitchen* in a planning state.

The main reason for using an SAS$^+$-based representation is that we can employ it to explicitly model knowledge and gaps in knowledge, so that the planner can efficiently reason about them. To this end, we must relax the assumption that in a state s all variables v have a value x. Instead, we accept states that are only partially defined, i.e., where some variables are undefined or "unknown". Conversely, we also need to represent future states in which gaps will have been filled. By nature, we can not know in advance which value a variable v will assume then, but we can nevertheless exploit the knowledge that, e.g., after executing a sensing action the value of v will be "known". To this end, we use so-called *Kval* variables Kval_v with $vdom(\mathit{Kval}_v) = \top, \bot$. With *Kval* variables we can also model the epistemic effects of sensing actions. For example, the action of running a room categorisation algorithm in a room is modelled in MAPL as follows:

```
(:sensor categorise_room
 :agent (?a - agent)
 :parameters (?r - room ?loc - place)
 :precondition
    (= (pos ?a) ?loc)
    (contains ?r ?loc)
 :effect (Kval ?a (areaclass ?r))
)
```

In words, this action model describes that an agent can sense the area class of a room, i.e. its being a kitchen,

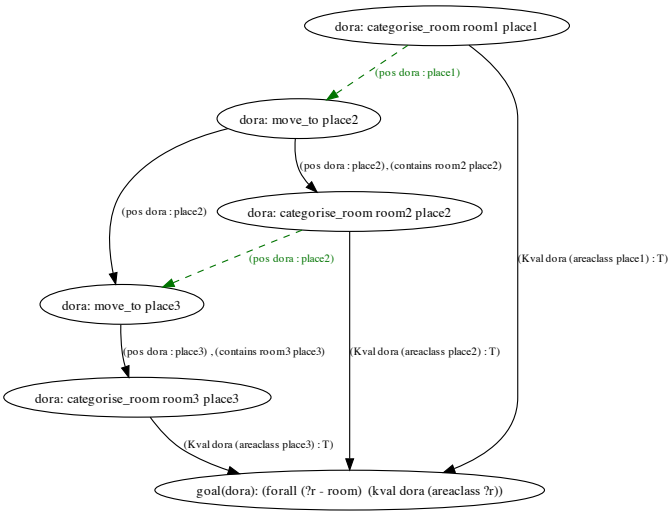


Fig. 7. A plan using sensory actions to satisfy epistemic goals.

office or hallway, once the agent is at a place that belongs to the room in question. At planning time, the outcome of observing *areaclass(r)* is yet unknown, therefore the effect of `categorise_room(r,loc)` is formally described as $Kval_{areaclass(r)} = \top$.

Of course, *Kval* variables can appear in goal formulae as well, so that we can conveniently express *epistemic goals*, i.e. goals concerned with closing knowledge gap. Goal formulae can contain expressions in first-order logic, in particular conditionals and quantifiers, so that we can give the robot goals like "categorise all rooms and explore all currently known places" , which would correspond to $\forall loc.Kval_{areaclass(loc)} = \top \wedge \forall place.explored(place)$.

Interestingly, due to the use of a quantified formula the goal will be re-evaluated repeatedly during the continual planning process, i.e. the planner will autonomously adapt its plan to explore and categorise newly discovered places and rooms. A (slightly simplified) example of a plan using sensing actions that satisfy epistemic goals is given in Fig. 7.

In the George scenario and in our next instantiation of Dora, information will not only be obtained by sensing, but also through interaction with humans. To plan for such multiagent interactions the robot must also reason about the knowledge of the other agents. We can express nested beliefs using MVSVs as well, e.g., "the robot R believes that human H believes that object o is a pen" is modelled as $B^{R,H}_{type(o)} = pen$. Knowledge gaps may arise in several variants when nested beliefs are used, depending on which agent is ignorant of the other's belief. Again, with MVSVs we can represent the differences succinctly using agent-specific "unknown" symbols. Consider, e.g., the difference between the statements "R knows that H does not know the location of the cornflakes" ($Kval^{R,H}_{pos(cornflakes)} = \perp^H$) and "R does not know if H knows the location of the cornflakes" (($Kval^{R,H}_{pos(cornflakes)} = \perp^H$). Just as sensing actions are modelled using standard *Kval* variables, we can use nested *Kval* variables to describe *speech acts*. In particular, we can describe *wh-questions* and answers to them ("where", "what colour", etc.) by modelling the appropriate nested belief effects. (Note: the planner was not used for dialogue planning in the George system as presented in this paper, but will be in its next instantiation).

a) Related Work: Reasoning about knowledge is classically studied in the field of epistemic logic [20]. The work presented here integrates concepts of epistemic logic into a planning representation so that a planning agent can reason about how it can change its own or another agent's state of knowledge in the future [19]. Similar approaches have been used previously to model planning in the presence of sensing [21], [22]. The MAPL representation presented here can additionally model nested and mutual beliefs, which is crucial for describing teamwork and dialogue [23], [24].

C. Representations for cross-modal learning

Cross-modal learning plays an important role in a self-extending system. It enables the system to, based on interaction with the environment and people, extend its current knowledge by learning about the relationships between symbols and features that arise from the interpretation of different modalities. It involves processing of information from multiple modalities, which have to be adequately represented. One modality may exploit information from another to update its current representations, or several modalities together may be used to form representations of a certain concept. In this subsection we focus on the former type of interaction between modalities and present the representations that are used for continuous learning of basic visual concepts in a dialogue with a human. While Section III describes the formation of belief models, which supervise the learning in the visual domain, this subsection focuses on representations that are being updated in this continuous learning process. All these principles are integrated and demonstrated in the George scenario described in Section VII.

1) Representations for visual concepts: The visual concepts are represented as generative models, probability density functions (pdf) over the feature space, and are constructed in online fashion from new observations. In particular, we apply the online Kernel Density Estimator (oKDE) [25] to construct these models. The oKDE estimates the probability density functions by a mixture of Gaussians, is able to adapt using only a single data-point at a time, automatically adjusts its complexity and does not assume specific requirements on the target distribution. A particularly important feature of the oKDE is that is allows adaptation from the positive as well as negative examples [26]. The continuous learning proceeds by extracting visual data in a form of a highdimensional features (e.g., multiple 1D features relating to shape, texture, color and intensity of the observed object) and oKDE is used to estimate the pdf in this high-dimensional feature space. However, concepts such as *color red* reside only within lower dimensional subspace spanned only by features that relate to color (and not texture or shape). Therefore, during online learning, this subspace has to be identified to provide best performance. This is achieved by determining for a set of mutually exclusive concepts (e.g., colors green, blue, orange, etc.) the subspace which minimizes the overlap of the corresponding distributions. The overlap between the distributions is measured using

the Hellinger distance as described in [27]. Therefore, during online operation, a multivariate generative model is continually maintained for each of the visual concepts and for mutually exclusive sets of concepts the feature subspace is continually being determined. The set of mutually exclusive concepts can then be used to construct a Bayesian classifier in the recognition phase, when the robot is generating a description of a particular object in terms of its color, shape, etc. However, since the system is operating in an online manner, the closed-world assumption can not be assumed; at every step the system should take into account also the probability of the "unknown model" as described in the following.

2) Accounting for unknown model: While maintaining good models of the visual concepts and being able to adapt those models is crucial for the robots online operation, the ability to detect gaps in the knowledge presented by these models is equally important. Generally speaking the robot collects the visual information about its environment as follows. First it determines a region in an image which contains the interesting information, then it "segments" that region and extracts the feature values z from which it later builds models of objects, concepts, etc. The visual information may be ambiguous by itself, and segmentation may not always be successful. We will assume that some measure of how well the segmentation was carried out exists and we will denote it by $s \in [0, 1]$. High values of s (around one) mean high confidence that a good observation z was obtained, while low values relate to low confidence.

Let $m \in \{m_k, m_u\}$ denote two possible events: (i) the observation came from an existing internal model m_k, and (ii) the observation came from an unknown model m_u. We define the knowledge model as a probability of observation z, given the confidence score s:

$$p(z|s) = p(z|m_k, s)p(m_k|s) + p(z|m_u, s)p(m_u|s). \quad (1)$$

The function $p(z|m_k, s)$ is the probability of explaining z given that z comes from one of the learnt models, $p(m_k|s)$ is the a priori probability of any learnt model given the observer's score s. The function $p(z|m_u, s)$ is the probability of z corresponding to the unknown model, and $p(m_u|s)$ is the probability of the model "unknown" given the score s.

Assume that the robot has learnt K separate alternative internal models $M = \{M_i\}_{i=1:K}$ from previous observations. The probability $p(z|m_k, s)$ can then be further decomposed in terms of these K models,

$$p(z|m_k, s) = \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s). \quad (2)$$

If we define the "unknown" model by M_0 and set $p(z|m_u, s) = p(z|M_0, m_u, s)p(M_0|m_u, s)$, then (1) becomes

$$p(z|s) = p(m_k|s) \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s) + p(m_u|s)p(z|M_0, m_u, s)p(M_0|m_u, s). \quad (3)$$

Note that the "unknown model", M_0, accounts for a poor classification, by which we mean that none of the learnt models supports the observation z strongly enough. We assume that the probability of this event is uniformly distributed over the feature space, which means that we can define the likelihood of model M_0, given observation z by a uniform distribution, i.e., $p(z|M_0, m_u, s) = \mathcal{U}(z)$. Note also, that the only possible unknown model comes from the class M_0, therefore $p(M_0|m_u, s) = 1$.

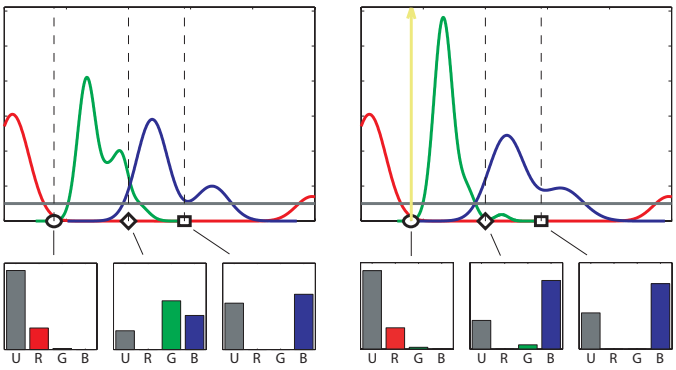


Fig. 8. Example of detecting the knowledge gaps and updating the 1D KDE representations. Top row: probability distributions for three colours (red, green, blue lines) and unknown model (gray line) in 1D feature space. Bottom row: a posteriori probabilities for the unknown model (U) and three colours (R, G, B) for three feature values denoted by the circle, the diamond and the square. Left column: before updates, right column: after updates.

The observation z can be classified into the class M_i which maximizes the a posteriori probability (AP). The a posteriori probability of a class M_i is calculated as

$$p(M_i|z, s) = \frac{p(z|M_i, m, s)p(M_i|m, s)p(m|s)}{p(z|s)}, \quad (4)$$

where $m = m_k$ for $i \in [1, K]$ and $m = m_u$ for $i = 0$.

In our implementations, the distribution of each i-th alternative of the known model $p(z|M_i, m_k, s)$ is continuously updated by the oKDE [25], while the a priori probability $p(M_i|m_k, s)$ for each model is calculated from the frequency at which each of the alternative classes M_i, $i > 0$, has been observed. The a priori probability of an unknown model (and implicitly of a known model), $p(m_u|s)$ is assumed non-stationary in that it changes with time. The following function decreases the "unknown" class probability with increasing number of observations N:

$$p(m_u|s) = e^{-0.5(\frac{N}{\sigma_N})^2}, \quad (5)$$

where σ_N is a user specified parameter that specifies how the robot's internal confidence about learned models changes with time.

With above definitions, the knowledge model is completely defined and allows discovery of knowledge gaps. They can be discovered through inspection of the AP distribution. In particular, we can distinguish two general cases:

- The observation z can be best explained by the unkown model, which indicates the gap in the knowledge; the observation should most probably be modeled with a model, which has not yet been learned.
- The a priori probability of the model that best explains the observation is low, which indicates that the classification

is very uncertain and that the current model can not provide a reliable result.

3) Illustrative example: For a better visualization of the knowledge update and gap discovery we will restrict our example to a one-dimensional case. Fig. 8 illustrates detection and filling of knowledge gaps for three cases (feature values) denoted by the circle, the diamond, and the square. The plots in the left column depict the models and the recognition at a particular step in the learning process, while the right column depicts the situation after the system has updated these models considering the detected knowledge gaps and the answers from the tutor.

Consider a scene similar to that presented in Fig. 2. Let us assume that the circle in Fig. 8 represents the yellow object and that the yellow colour has not been presented to the robot before. Therefore, the corresponding model for colour yellow has not yet been learned and the feature value obtained from the segmented yellow object fails in a not yet modeled area. This value is thus best explained by the "unknown model", which has the highest a posteriori probability. The robot detects this gap in his knowledge and asks the tutor "*Which colour is this object?*", and after the tutor provides the requested information, the robot initializes a model for yellow colour. However, since only one sample does not suffice to build a reliable representation, the yellow colour will only be able to be recognized after some additional yellow objects are observed.

The feature value denoted by a diamond in Fig. 8 is best explained by a green model, however this recognition is not very reliable, therefore the robot asks the tutor: "*Is this object green?*" to verify its belief. After the tutor replies "*No. It is blue.*", the robot first unlearns the representation of green and updates the representation of blue. The corrected representations, depicted in the pdfs in the right column in Fig. 8, then enable the correct recognition as indicated by the second bar plot in the right column of the Fig. 8.

The last case denoted by the square shows another example of non-reliable recognition, which triggers the additional clarification question to the tutor: "*Is this object blue?*" After the robot gets a positive answer, it updates the representation of blue, which increases the probability of the recognition.

V. Goal management: choosing between different epistemic goals

In the previous sections the focus was very much on the *representation* of knowledge gaps and on the *understanding* of knowledge limitations. In this section we discuss a generic framework to *generate* and *manage* epistemic goals that correpsond to knowledge gaps in these representation. We propose a *goal generation and management framework (GGM)* that enables the robot to decide *which* gaps in its representation to eliminate *when* and generate appropriate behaviour to self-extend its knowlegde.

We have built on the work of [28], to produce the design illustrated in Figure 9. This design is a general framework, or schema, for an architecture for goal generation and management that tackles the mentioned issues. It specifies a collection of interacting elements which must be included in any instantiation of the framework, although the precise details of the instantiation will inevitably vary between instances. The elements of the framework are described in more detail below.

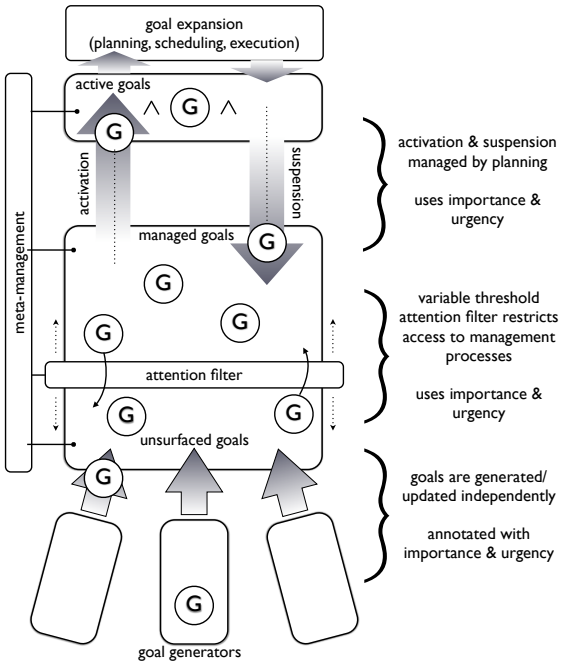


Fig. 9. The goal generation and management framework.

At the bottom of the framework, a system's drives are encoded as multiple *goal generators*. These are concurrently active processes which monitor the system's state (both the external world and internal representations) and produce *goals* to satisfy the system's drives. Generators can also remove previously generated goals if they are judged to no longer be appropriate. In this manner we can say that the system's drives are encoded in the goal generators (either explicitly or implicitly). We work from the assumption that as a goal passes up through the framework from a generator and influences a system's behaviour, it is inspected by processes of greater and greater computational complexity. Therefore the lower strata of the framework exist to protect these processes (and thus overall system resources) from having to consider more goals than is necessary (where this could be a contingent judgement). The main mechanism in the framework for protecting the management processes is the *attention filter*. This is a coarse barrier which uses simple, fast processing to let some some goals through to the management mechanisms whilst blocking others. Goals which make it through this filter are described as *surfaced*, thus the goals which fail to pass the filter are referred to as *unsurfaced*. A collection of management processes determine which of the surfaced goals should be combined to serve as the goals being actively pursued by the system. If a goal is selected in this way we describe it as *activated*. If a goal is removed from the set of goals being pursued by the system we refer to it as *suspended*.

In order to fulfil their roles, the filtering and management processes require information on which to base their decisions.

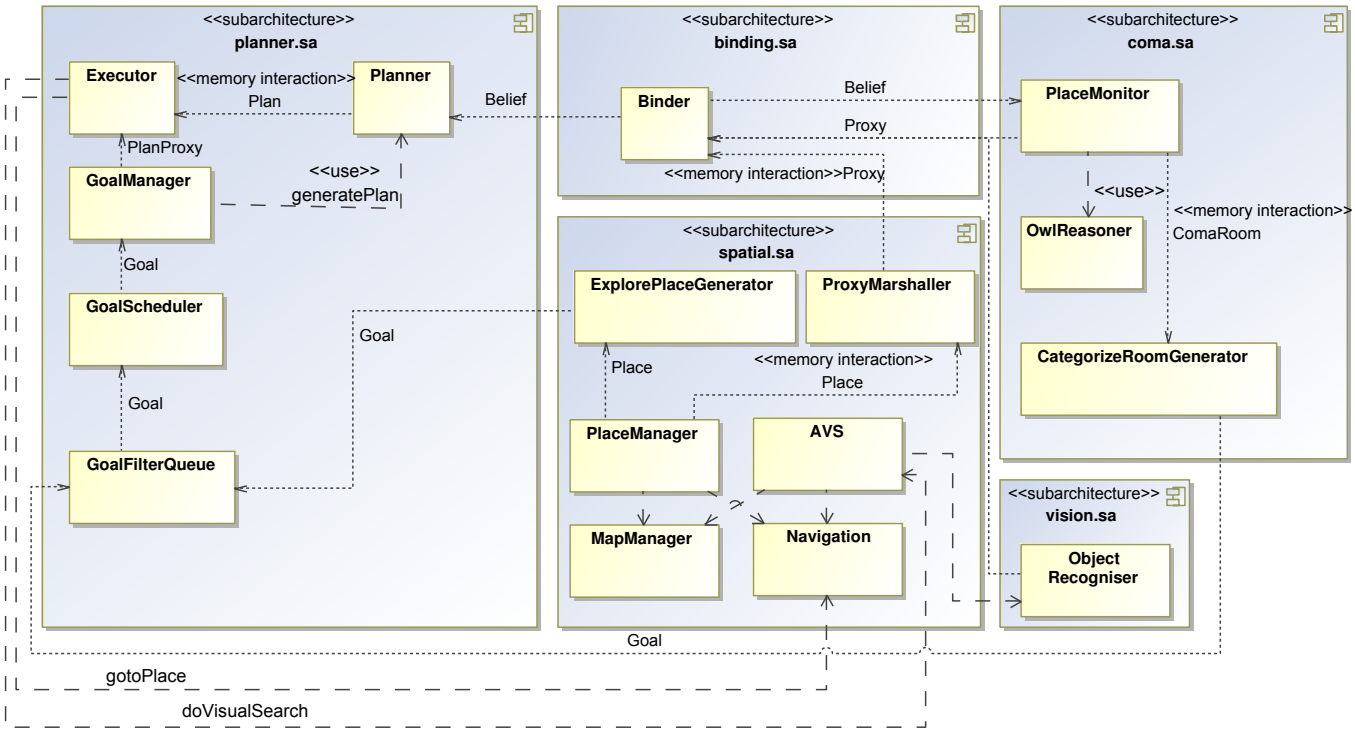


Fig. 10. Dora system architecture. For clarity, all memory interactions are not depicted as information flow mediated through working memories in the subarchitectures but as directed dotted connections of sources and sinks. The dashed lines represent synchronous request-reply calls to components by mutually modifying memory structures.

Following the original work [28], the framework requires that goal generators annotate each goal with a description of the goal's *importance* and *urgency*, and keep these descriptions up to date as long as the goal exists. Importance should reflect the significance of the goal to the agent (as motivated by the related drive). Urgency should reflect the necessity of achieving the goal sooner rather than later. As we shall see later, producing importance and urgency descriptions for use in such a framework is a problem in itself. In addition to these descriptions, the framework allows the management processes to use whatever approaches are required to select and maintain a set of active goals. Perhaps the minimum requirements on these processes is the ability to check whether a goal, or collection of goals, can be achieved (thus positing planning as a goal activation, as well as achievement, mechanism).

The GGM is currently implemented as one of the core concepts in our exploring robot Dora (cf. Sec. VI). In this system we derive the importance of a goal from an estimated *information gain* computed for the epistemic goals. In brief, the information gain for achieving the goal of exploring a yet unexplored place is derived from the measures shown in Fig. 6. The information gain of categorizing a room is similarly designed, assuming that a categorising bigger rooms yields more information. The GGM continuously monitors these measures of information gain and relates it to the *costs* to actually achieve this goal acquired by asking the planner. We are currently not employing a notion of *urgency* in our implementation as the robot's drives are not prioritised so far.

The GGM in cooperation with planning implements *action selection and execution* in our systems, allowing the robots to expose effective and efficient self-extending behaviour.

VI. Dora the Explorer

The current implementation of Dora is focused on spatial representations and two types of knowledge gaps that give rise to epistemic goals to fill these gaps: explore a place and categorise a room. Dora's system architecture is composed of five of the subarchitectures discussed earlier in this paper running all on one Laptop computer on the autonomous robot, cf. Fig. 1(a). The composition is sketched in Fig. 10. The diagram is adopted from UML 2.0 specification and illustrates the information flow between components, and also across subarchitectures. Most of the flow is realised by interactions with the working memories in an event-driven manner as proposed by CAS in Sec. II. For clarity and readability the interactions are not shown as connections to the respective memories but linking sources and sinks of information directly following the schema pictured in Fig. 3(b). The diagram does however not include all components. We focus here on those that are required to understand the architecture facilitating self-understanding and -extension, disregarding those that can be seen only as support services and sensor abstraction.

The application domain of Dora is also reflected in this architecture when we compare it to the architecture of George: spatial.sa is only relevant for a mobile robot and the reasoning about spatial concepts implemented in coma.sa is likewise specific to the scenario (cf. Sec. IV-A4). Furthermore, Dora is studied as our first system that employs goal management and planning to choose and pursue epistemic goals.

vision.sa only plays a minor role in this system. The ObjectRecogniser component (based on the FERNS detector [29]) detects objects visually when triggered by AVS. The

ProxyMarshalling is a mediator component selecting spatial information to be presented as Proxy to the binder. The PlaceMonitor in coma.sa fulfills a similar purpose. As can be seen in the figure, binding.sa in Dora as in George is working on a unified representation of beliefs and proxies only, allowing a simple transformation of information in the planning domain required by the Planner. As can been seen as well, the goal management scheme of motivation in Dora is implemented as part of planner.sa. Epistemic goals are created from observing knowledge gaps in the specific representations in other SA's working memories as discussed in Sec. IV-A2 and IV-A4. These goals are filtered, scheduled, and planned for following the principles introduced in Sec. V. The Executor eventually executes and monitors actions by triggering either the navigation to explore a place or the AVS component (cf. Sec. IV-A1) which autonomously interacts with other components to visually search for objects that allow the OwlReasoner to derive conceptual knowledge about rooms (cf. Sec. IV-A4).

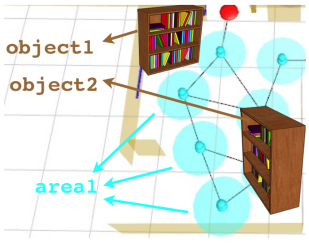
Fig. 10 on the previous page also illustrates our decomposition strategy. The only representations that are currently exchanged across borders of subarchitecture are information related to binding (cf. Sec. III) and the epistemic goals corresponding to the knowledge gaps.

A. Dora running

A prototypical run with the current Dora implementation unfolds as follows:

1) Dora starts from scratch without any knowledge about the specific environment she is operating in.
2) Optionally, Dora can be given a short tour by a human instructor to create an initial representations already containing some knowledge gaps. Fig. 1(b) on page 2 shows such a partially known environment.
3) Dora autonomously explores her environment having drives to self-extend with respect to two types of knowledge gaps: unexplored places as they have been defined in the place layer and yet uncategorised rooms as defined in conceptual layer of the spatial representations. In the example in Fig. 1(b) on page 2, the rooms area0, area1, area2 give rise to room categorisation drives, whereas the different placeholders lead to exploration goals. Note that a number of placeholders (notably the ones labelled "8(7)", "7(16)", and "20(19)") are in space that will later be segmented into new rooms, which then, in turn, will also need to be categorised.
4) A room categorization goal is considered satisfied when a more specific concept can be inferred for a PhysicalRoom instance in the ABox of the conceptual map layer, cf. Fig. 11. An exploration goal is satisfied if the robot either turns the placeholder into a real place or discards it, because it turned out as a false hypothesis.

The two types of gaps are created and monitored by the components PlaceManager@spatial.sa and PlaceMonitor@coma.sa, respectively. Fig. 10 illustrates how these components submit hypotheses about ComaRoom (a detected but not yet categorised room) and Place (a detected but not yet explored place) to their working memories. From these gaps



(a) Dora has found 2 bookshelves.

Bookshelf(object1)
Bookshelf(object2)
in(object1,area1)
in(object2,area1)

→ Office(area1)

(b) New facts stored in the ABox.

Fig. 11. Continuing the example in Fig. 1(b) on page 2: based on the presence of 2 OfficeObject instances the DL reasoner infers that area1 instantiates Office.

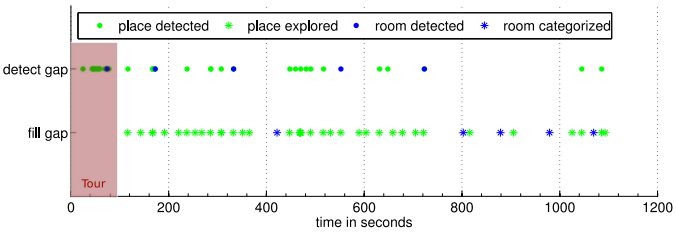


Fig. 12. Exemplary course of action for filling knowledge gaps in a real run.

epistemic goals are generated by the goal generators ExplorePlaceGenerator and CategoriseRoomGenerator, respectively. Thus, a number of goals is always present in the system corresponding to these gaps. Dora's behaviour is driven by the selection of a subset of these goals by goal generation and management (GGM, cf. Sec. V) and the execution of actions according to generated plans to achieve these goals.

Fig. 12 visualises an exemplary run of Dora illustrating the course of action she takes with respect to self-extending for the two types of knowledge gaps. The x-axis shows the time in seconds since the beginning of the run. The figure thus indicates the time when a new gaps are detected (upper line) and the time, when a particular epistemic goal has been accomplished to fill a gap in the knowledge. This particular run comprised an initial tour taking Dora from the corridor (the long room in the centre of Fig. 1(b) on page 2) to the room in the upper-right corner in that figure. It can be seen in Fig. 12 that she is passively detecting gaps in her knowledge in the phase labelled "Tour", but not yet autonomously extending it. Only after the tour Dora interleaves categorisation of rooms to fill gaps with the exploration of new places. A video[4] of the real robot operating in an office environment can support comprehension of this illustration and provide the reader with a better understanding of Dora's generated behaviour.

Taking a closer look on the actual processes in the system, we can see how the interaction between component works. Fig. 13 on the following page pictures the activity that the robot goes through from the detection of a knowledge gap to its filling. The example illustrated in the figure is corresponding to "explore place" only, but the structure is similar for "categorise room". It starts with spatial.sa hypothesising a new place and thus generating a Place in the working memory that is marked as being hypothetical. This generation triggers binding and ExplorePlaceGenerator to create a Belief about this place and an epistemic Goal to explore this place. After

[4] http://cogx.eu/results/dora/

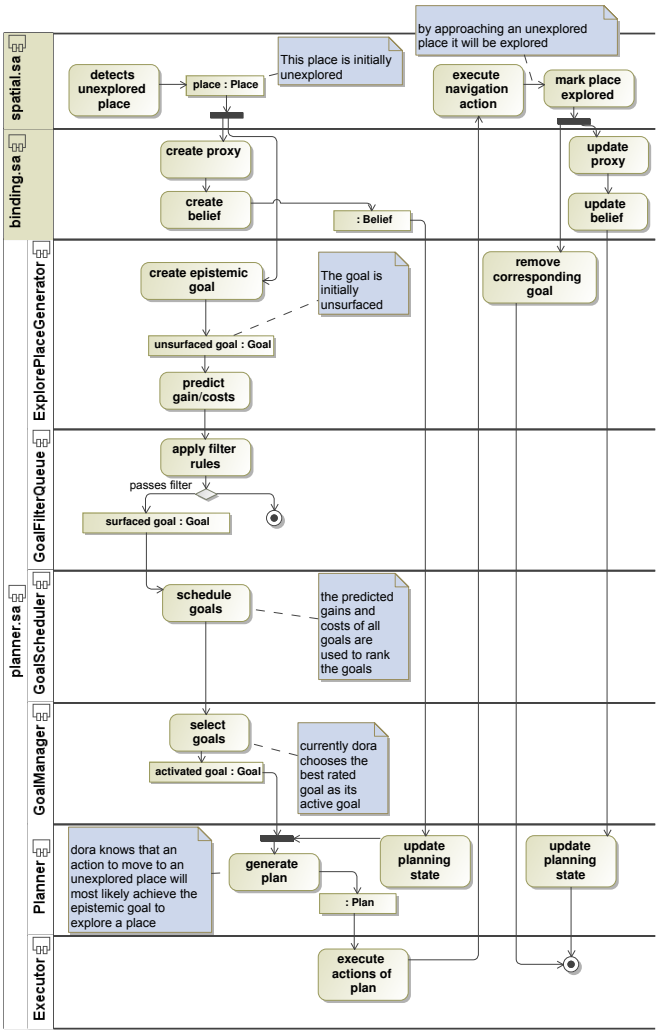


Fig. 13. Activity diagram illustrating the "path" of a knowledge gap from its generation to its filling.

the motivation-related components have filtered and scheduled the generated goal, the planner is triggered to generate a plan to actually achieve it. The Executor then executes the actions of the plan. One action will be to navigate towards the placeholder which will – in this example – make it explored. This update is again propagated through the working memory, resulting in the goal to be removed and the belief being updated asynchronously.

In this paper we present two experiments, each studying slightly different aspects of the Dora scenario. The first experiment focuses on the spatial representation with an emphasis on the nonmonotonic reasoning about space in the conceptual layer. The second one focuses on the goal management and the benefits of having such a management framework in the system. For sake of reproducibility and focus on specific aspects these experiments are carried out in simulation. Our simulation framework transparently substitutes the sensors and actuators, still allowing to run the core system unmodified and in a situated way with sensing-actuation loops closed. The simulator operates using the floor plan of one of the real environments in which Dora operates (cf. Fig. 14).

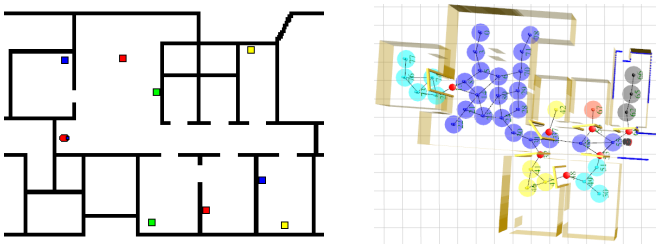


Fig. 14. Stage simulation model used in the experiments (l) and screenshots of the visualisation tool acquired during one of the three experiments (r).

B. Experiment 1: Spatial Representation

One consequence of the uncertainty and partiality of the observations Dora is dealing with is that the map building process is nonmonotonic. Structural and conceptual abstractions may need to be reconsidered in the light of new evidence acquired during the active exploration. In this experiment we assess the accuracy and appropriateness of our nonmonotonically built spatial representation as the robot keeps exploring.

Setup: The map consisted of eight rooms: a corridor, a terminal room, a lab, two offices, two restrooms, and a printer room, cf. Fig. 14. This constitutes the ground truth for our tests of the accuracy of the room maintenance. The robot was ordered to perform an autonomous exploration, which means that only placeholder exploration goals were considered by the motivation system. To evaluate the coverage that this exploration yields, we determined a gold standard of 60 Place nodes to be generated in order to fully, densely and optimally cover the simulated environment. We achieved this by manually steering the robot to yield an optimal coverage, staying close to walls and move in narrow, parallel lanes.
We performed three runs with the robot in different starting positions, each time with an empty map. Each run was cut-off after 30 minutes. The robot was then manually controlled to take the shortest route back to its starting position.

For the evaluation, the system logged the state of its ABox each time a new room was created, or an existing one was deleted. This subsumes cases in which rooms are split or merged. At each such step, the generated map was compared to the ground truth for the room representation and to the gold standard for Place node coverage. The first room instance to cover part of a ground-truth room is counted as *true positive (TP)*. If that room instance extends into a second room, it is counted as TP only once, and once as a *false positive (FP)*. Each additional room instance inside a ground-truth room is also counted as FP. *False negatives (FN)* are ground-truth rooms for which no room instance exists. Using these measures, precision P, recall R and the balanced f-score F for the room maintenance are as follows: $P = |\,TP\,|/(|\,TP\,|+|\,FP\,|)$, $R = |\,TP\,|/(|\,TP\,|+|\,FN\,|)$, $R = 2\times((P\times R)/(P+R))$. We compute a normalised value for coverage using $coverage = |\,nodes\,|/60$.

Results: Fig. 15 on the next page shows the development of the relevant measures during the three experimental runs. As can be seen, the accuracy (balanced f-score) of the representation is monotonically increasing towards a high end result (0.8, 0.79 and 0.93, resp.). The increases and decreases in precision during the individual runs are due to the introduction and

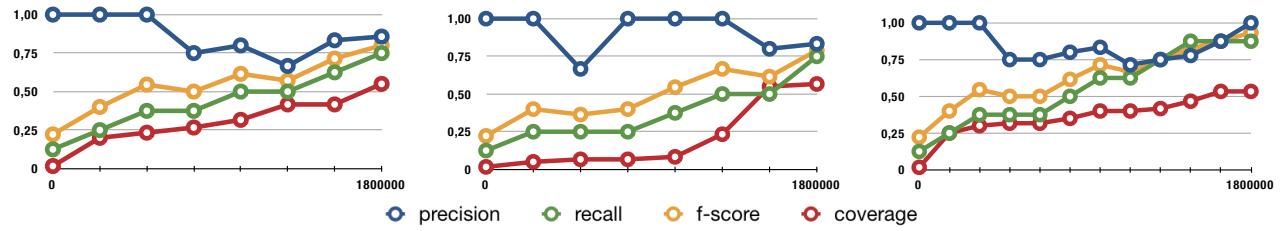


Fig. 15. Plots for precision, recall, balanced f-score and coverage of each of the three experimental runs. The Y-axis shows the normalised values for precision, recall, balanced f-score, and coverage (0–1). The X-axis is time, in milliseconds.

retraction of false room instances. Recall can be interpreted as coverage in terms of room instances. After 30 minutes the exploration algorithm yielded a relatively high recall value (0.75, 0.75 and 0.875, resp.), i.e., most of the rooms had been visited. A recurring problem here was that the two smallest rooms were often only entered by a few decimetres. This was enough to consider the corresponding Placeholder to be explored, but not enough to create an additional Place node beyond the doorway – which would have been the prerequisite for room instance creation. The node coverage that the algorithm achieved after 30 minutes (33, 34, 32 out of 60, respectively) can be attributed partly to the 30-minutes cut-off of the experiment, and partly to the exploration strategy which goes for high information gain Placeholder first. These tend to be in the middle of a room rather than close to its walls.

C. Experiment 2: Goal Management

As discussed in Sec. IV-B0a and V, we have two alternatives to express Dora's drives. First, we can explicitly use quantifiers to create on conjunctive goal for the overall system to explore all places and categorise all rooms and rely on the replanning ability of the continual planning (cf. Sec. IV-B0a). We term this system setup *Conjunct Goal Set (CGS)*. The proposed alternative is to make use of the goal generation and management approach and let it select and schedule the individual goals. Our hypothesis is that (i) the effort for planning is reduced as we chunk the problem into smaller pieces, making it tractable if it comes to more complex problems, and (ii) goal management is a powerful and simple means to encode domain knowledge into the behaviour generation in Dora. We refer to this second setup as *Managed Goal Set (MGS)*.

Setup: For this study we restricted the space to be explored five rooms and the corridor (being the right part of Fig. 1(b) on page 2 without the large room). The ultimate goal in this setup for the robot is to categorise all rooms using the two typical objects placed in each of the five rooms. The objects describe one of three categories according to the OpenMind Indoor Common Sense Database (room, office, and kitchen), allowing the conceptual mapping SA to categorise these rooms.

A single run starts with a short tour through the corridor. Then Dora is switched to autonomous mode and starts acting in response to her goals. Fig. 12 on page 13 is generated from one of these runs including the tour and the categorisation of five rooms. In total we ran the system 15 times: 8 in MGS configuration and 7 in CGS. A run for the CGS configuration was defined as complete when the conjunctive goal was achieved

TABLE I
PLANNING TIME MEASURES (ALL IN SECONDS).

	CGS	MGS
avg. time per planning call	0.621 s	0.292 s
avg. time spent on planning	48.843 s	8.858 s

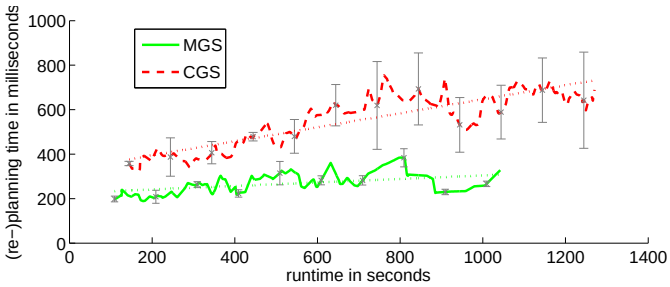


Fig. 16. Averaged planning time during a system run.

(i.e., no places left unexplored and no rooms uncategorised). The MGS configuration was said to be complete when no more surfaced goals remained.

Results: As part of our experiments that are fully detailed in [30] we were interested in the effect the GGM approach has on the complexity of problems to be solved by planning. So me measured the time Dora spend planning in the runs for the two different setups. These measures are summarised in Table I. The differences between the averaged timings taken for the two configurations are statistically significant with $p < 0.0001$ in Mann-Whitney testing for all measures shown in the table.

As the first row of the table indicates, there is a significant difference between the average time taken by a single call to the planner. A call occurs either when the goal management activates a new goal or when replanning is triggered by a state change. Planning calls in CGS take more than twice the time compared to MGS. This is due to the higher complexity of the planning problems in the CGS configuration (it is planning for the conjunction of all epistemic goals rather than a single goal). If we look at the average time spent on planning in total per run (second row in Table I) the difference is more prominent. This is due to the fact that in the CGS configuration the planner is triggered more often: 79.0 times on average, compared to 31.1 times for the MGS configuration. This is because the longer plan lengths required in CGS are more likely to be affected by state changes and thus require more frequent replanning.

Figure 16 shows how the complexity of planning problems evolves as the system is running. It depicts the length of single planner calls against the runtime of the system. For comparability, this plot has been created from a partial set

of all runs (five of each configuration) containing only those in which Dora successfully categorised all five rooms. The planning time is averaged at discrete time steps across all the runs of each setup. The error bars indicate the standard error in averaging. From this figure it is apparent that, in agreement with the data in Table I, less planning effort is required in MGS compared to CGS. It can also be seen that the progression over runtime is different in the two cases. While the trend, indicated by a linear fitting shown as a dotted line in Fig. 16, is a shallowly included line for MGS, a steeper increase in average planning time can be seen for CGS. This steeper increase can be associated with the increasing size of the planning problems the CGS configuration faces as Dora's knowledge increases: planning for all possible goals over a larger and larger state becomes increasingly difficult. This underpins our hypothesis that with a suitable mechanism for goal selection we can tackled the challenge of increasingly complex environments and correspondingly high numbers of knowledge gaps in our representations.

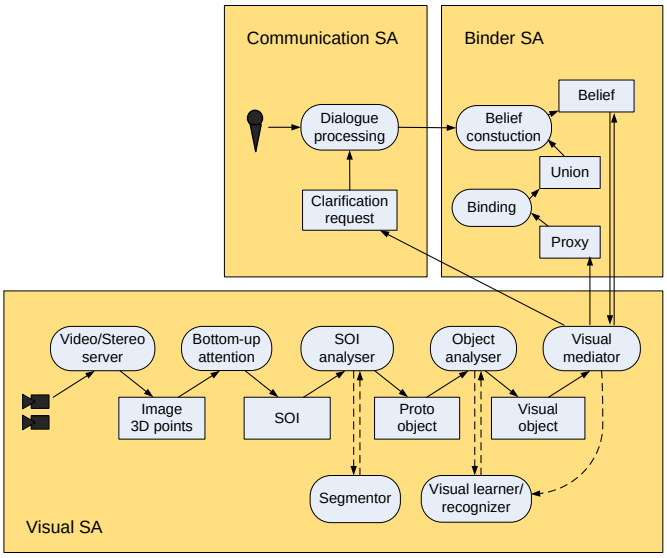


Fig. 17. Architecture of the George system.

VII. George: curiosity driven cross modal learning

The George scenario has been designed to demonstrate, monitor, and show progress on the development of the integrated system for *learning the association between visual features of an object and its linguistically expressed properties*. The main goal is, therefore, to integrate the developed vision routines, learning and recognition competencies, dialogue capabilities, as well as different kinds of representations and belief models in an overall system.

A. Scenario setup and example script

The robot operates in a table-top scenario, which involves a robot and a human tutor (see Fig. 2(a)). The robot is asked to recognize and describe the objects in the scene (in terms of their properties like colour and shape). There are a single or several objects (i.e., up to five) in the scene (but still, with limited occlusion). The human positions new objects on the table and removes the objects from the table while being involved in a dialogue with the robot. At the beginning the robot does not have any representation of object properties, therefore he fails to recognize the objects and has to learn. To begin with, the tutor guides the learning and teaches the robot about the objects. After a while, the robot takes the initiative and tries to detect the ignorance and to learn autonomously, or asks the tutor for assistance when necessary. The tutor can supervise the learning and correct the robot when necessary; the robot is able to correct erroneously learned representations. The robot establishes the transparency and verbalizes its knowledge and knowledge gaps, as well as intended actions. In a dialogue with the tutor, the robot keeps extending and improving the knowledge. The tutor can also ask questions about the scene, and the robot is able to answer (and keeps answering better and better). At the end, the representations are rich enough to accomplish the task - to correctly describe the initial scene.

Two main types of learning are present in the George scenario, which differ on where the motivation for learning update comes from. In tutor driven learning the learning process is initiated by the human teacher, while in the tutor assisted learning, the learning step is triggered by the robot.

Tutor driven learning is suitable during the initial stages, when the robot has to be given information, which is used to reliably initiate (and extend) visual concepts. Consider a scene with a single object present:

H: Do you know what this is?
G: No.
H: This is a red object.
G: Let me see. OK.

Since at the beginning George doesn't have any representation of visual concepts, he can't answer the question. After he gets the information, he can first initiate and later sequentially update the corresponding information.

After a number of such learning steps, the acquired models become more reliable and can be used to reference the objects. Therefore, there can be several objects in the scene, as in Fig. 2, and George can talk about them:

H: What colour is the elongated object?
G: It is yellow.

When the models are reliable enough, George can take initiative and try to learn without being told to. In this curiosity driven learning George can pose the question to the tutor, when he is able to detect the object in the scene, but he is not certain about his recognition. As described in Section IV-C in such *tutor assisted* learning there are two general cases of detecting uncertainty and knowledge gaps. If the robot can not associate the detected object with any of the previously learned models, it considers this as a gap in his knowledge and asks the tutor to provide information:

R: Which colour is this object?
H: It is yellow.
R. OK.

The robot is now able to initialize the model for yellow and,

after the robot observes a few additional yellow objects, which make the model of yellow reliable enough, he will be able to recognize the yellow colour.

In the second case, the robot is able to associate the object with a particular model, however the recognition is not very reliable. Therefore, the robot asks the tutor for clarification:

R: Is this red?
H: No. This is yellow.
R. OK.

After the robot receives the answer from the tutor, he corrects (unlearns) the representation of the concept of red and updates the representation of yellow and makes these two representations more reliable.

In such mixed initiative dialogue George continuously improves the representations and learns the reliable models of basic visual concepts. After a while George can successfully recognize the acquired concepts and provide reliable answers:

H: Do you know what this is?
G: It is a blue object.
H: What shape is the red object?
G: It is elongated.

B. System architecture and processing pipeline

The George system is composed of three subarchitectures: *Binder SA*, *Communications SA* and *Visual SA*, as depicted in Fig. 17. The components of visual subsystem (SA) can be divided in three distinct layers: the quantitative layer, the qualitative layer and the mediative layer.

The quantitative layer processes the visual scene as a whole and implements one or more *bottom-up* visual attention mechanisms. A bottom-up attention mechanism tries to identify regions in the scene that might be interesting for further visual processing. George has currently one such mechanism, which uses the stereo 3D point cloud provided by *stereo reconstruction component* to extract the dominant planes and the things sticking out from those planes. Those sticking-out parts form spherical 3D spaces of interest (SOIs). The *SOI Analizer* component validates the SOIs and, if deemed interesting (SOI persistence, stability, size, etc.), upgrades them to *proto-objects* adding information that is needed for the qualitative processing (e. g. segmentation mask).

The qualitative layer processes each interesting scene part (object) individually, focusing on qualitative properties. After the extraction of the visual attributes (Visual Learner-recognizer), like color and shape, the *Object Analyzer* upgrades the proto-objects to *visual objects*. Visual objects encapsulate all the information available within Visual SA and are the final modal representations of the perceived entities in the scene. Also, the learning of visual attributes is performed on this layer.

The main purpose of *the mediative layer* is to exchange information about the perceived entities with other modalities. This is usually not done directly, but via specialised a-modal subarchitectures like the Binder SA (Section III). The *Visual Mediator component* adapts and forwards the modal information about objects to the binder (each visual object is represented by a dedicated proxy in the binder).

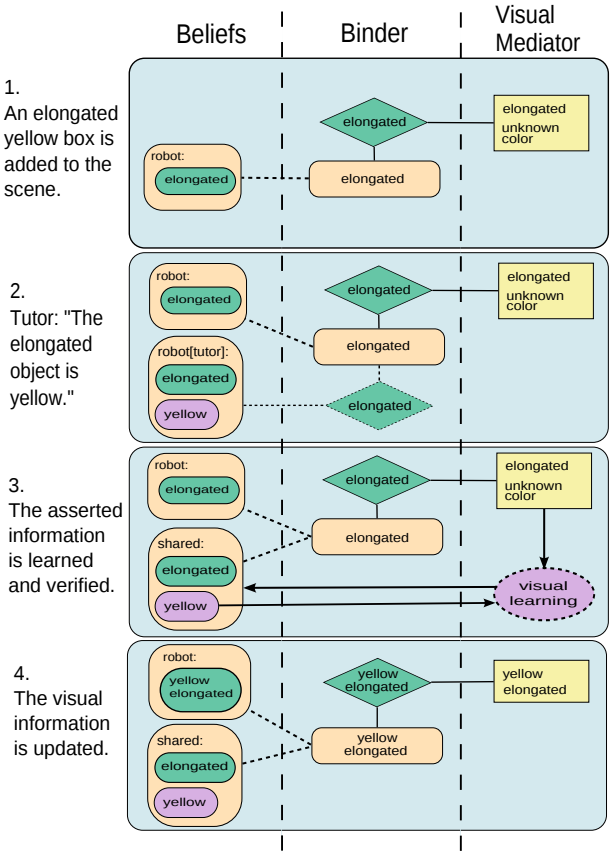


Fig. 18. Example of processing pipeline. The green color represents restrictive information, while the violet color denotes asssertive information. Only the beliefs and other data structures pertaining to the yellow tea box are shown.

The component also monitors beliefs for possible learning opportunities, which result in modal learning actions. Another important functionality of the mediator is to formulate and forward clarification motivations in the case of missing or ambiguous modal information. Currently, these motivations are directly intercepted by Communication SA, which synthesizes a question about the certain object property.

Now, let us describe the processing pipeline on one illustrative example. We will describe in more detail what happens after the human places several objects in the scene (see Fig. 2) and refers to the only elongated object in the scene (the yellow tea box) by asserting "*H: The elongated object is yellow.*".

In Visual SA the tea box is represented by a *SOI* on the quantitative layer, a *proto-object* on the qualitative layer and a *visual object* on the mediative layer. Let us assume that the *Visual Learner-recognizer* has recognized the object as of elongated shape, but has completely failed to recognize the color. In the binder this results in a one-proxy union with the binding features giving the highest probability to the elongated shape, while the color is considered unknown. This union is referenced by the single robot's private belief in the belief model (Fig. 18, step 1).

The tutor's utterance 'The elongated object is yellow.' is processed by the Communication SA, resulting in a new belief attributed to the tutor. This belief restricts the shape to elongated and asserts the color to be yellow. Before the belief is actually added to the belief model, the binder translates it

to a binding proxy (phantom proxy) with the shape restriction as a binding feature. In the most probable configuration, the phantom proxy is bound to the existing union, which already includes the visual proxy representing the tea box (Fig. 18, step 2). The union is promptly referenced by the attributed belief and the phantom proxy is deleted soon after.

In Visual SA, the mediator intercepts the event of adding the attributed belief. The color assertion and the absence of the color restriction in the robot's belief is deemed as a learning opportunity (the mediator knows that both beliefs reference the same binding union, hence the same object). The mediator translates the asserted color information to equivalent modal color label and compiles a learning task. The learner-recognizer uses the label and the lower level visual features of the tea box to update its yellow color model. After the learning task is complete, the mediator verifies the attributed belief, which changes its epistemic status to shared (Fig. 18, step 3). The learning action re-triggers the recognition. If the updated yellow color model is good enough, the color information in the binder and belief model is updated (Fig. 18, step 4).

A similar process also takes place in tutor assisted learning, when the robot initiates, based on an unreliable recognition, the learning process, e.g., by asking "*R: Is this red?*". In this case, the need for assistance reflects in a robot's private belief that contains the assertion about the red color and references the union representing the object. Based on this belief the Communication SA synthesizes the above question. When the robot receives the positive answer, he updates the representation of red, using a very similar mechanism as in the case of tutor driven learning.

C. Experimental results

The system was primarily developed to work in an interaction with a user. However, to comprehensively analyse the proposed learning strategies, such interactive work is time consuming and impractical. Therefore, we instead performed quantitative evaluation in simulation. The simulation environment uses stored images, which were previously captured and automatically segmented. We used a number of everyday objects, similar to those presented in Fig. 2. Each image, containing a detected and segmented object, was then manually labeled. In the learning process the tutor is replaced by an omniscient oracle, which has the ground truth data available. In this way the extensive tests could be automatically performed and a reliable evaluation of the proposed methods were obtained.

Six visual attributes were considered; four colours (red, green, blue, yellow) and two shapes (elongated, compact). The database that we used for learning contains 500 images. 400 images were used to incrementally learn the representations of six visual properties, while the rest 100 of them were used as test images. We repeated the experiment for 100 runs by randomly splitting the set of images into the training and test set and averaged the results across all runs.

During the experiment, we kept incrementally updating the representations with the training images using the Tutor driven (denoted as TD) and the Tutor assisted (denoted as TA) learning strategies. Note that in both cases the first 15 images were added in a tutor driven mode to form the initial models. At each step, we evaluated the current knowledge by recognising the visual properties of all test images. The learning performance was evaluated using two performance measures: *recognition score*, which rewards successful recognition (true positives and true negatives) and penalises incorrectly recognised visual properties (false positives and false negatives), and *tutoring costs*, which measure the level of the tutor's involvement, as defined in [31].

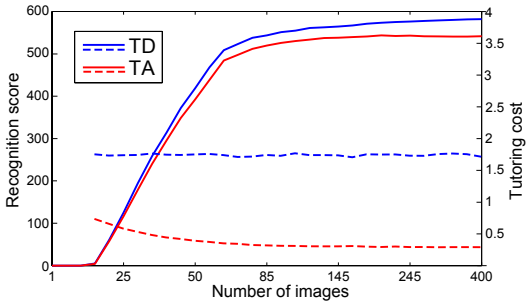


Fig. 19. Experimental results. TD: Tutor driven learning, TA: Tutor assisted learning, solid line: Recognition score, dashed line: Tutoring cost.

Fig. 19 shows the evolution of the learning performance over time for both learning strategies. The first thing to note is that the overall results improve through time. The growth of the recognition score is very rapid at the beginning when new models of newly introduced concepts are being added, and still remains positive even after all models are formed due to refinement of the corresponding representations.

Tutor-driven approach performs better, since the correct information is always given by the tutor. The inherent problem of any continuous learning framework, which involves autonomous updating of the knowledge, is propagation of errors. This is also reflected in the lower performance of the Tutor assisted approach. However, we also have to take into account the tutoring costs that occur during the learning. In Tutor-driven learning mode they are almost constant; the tutor always gives all the information about the current object, which is available. The costs of Tutor-assisted learning are significantly lower. The robot keeps asking the tutor only at the beginning of the learning process; after its knowledge gets improved the number of questions drops and most of the costs relate to the fact that the tutor has to listen to the robot and await for its questions. There is, as expected, a trade off between the quality of the results and cognitive load the tutor has to invest in the learning process. The best option would therefore be to first invoke the tutor driven approach and later on, when the models are reliable enough, switch to the tutor assisted mode.

VIII. Conclusion

In this paper we have presented a way of thinking about autonomous learning that is focussed on architectures and representations. The representational component of our theory is two-fold: on the one hand we employ representations of uncertainty and gaps in different modalities; on the other we represent how that lack of knowledge may change under action. The architectural theory is coupled: representations are

shared within working memories and linked across them, again in a way that explicitly represents the different ways they might be linked. In other words our systems reason explicitly about the multiple and uncertain ways that information from different modalities might be related. We also represent novelty in the structural rules that represent the universal relationships across modalities. Finally the architectural part of the theory also describes a way that possible learning goals can be quickly ranked, so that as systems are scaled that only a feasibly small subset are actually planned for.

We have shown that this approach works, by implementing two robot systems. Each illustrates different aspects of our approach. Dora illustrates the architecture, representations of gaps and uncertainty in spatial representations, the goal management system, and the use of planning with epistemic goals. George illustrates how we explicitly represent uncertainty in multi-modal representations of a specific situation, and uncertainty and novelty in the long term model of how different modalities are related.

What are the open research issues? First of all our approach to novelty is limited. The work on KDE models provides a particular approach to this, in a particular domain, but it is far from complete. There is also a question about how constrained the tacit design knowledge makes the self-extension. At the moment Dora, and to a lesser extent George extend their models within knowledge spaces that are quite well defined. The Dora design tacitly assumes that placeholders will become places, and George has the visual features necessary to learn the correct associations with words describing colour and shape. In addition the typology we have described for different types of incompleteness is only a beginning. Most challengingly, however, we have not yet dealt with the representation of different kinds of outcome or causal incompleteness. It is in general very difficult to model and reason about these in worlds with noisy observations and noisy actions. This is because an unexpected outcome could be due to observation noise, action noise, or true novelty. Variations on latent variable models such as factored POMDPs provide a probabilistic approach, but these are notoriously difficult to learn and reason with. To identify hidden causes in models of actions is also difficult. Suppose an action of a robot fails, such as a grasping action? This could be because of picking a poor grasp position, failing to grip strongly enough, or estimating wrongly where the obejct was. These possible causes can be distinguished if the robot has the a priori notion that they are possible causes of grasp failure, but in general we want the robot to be able to discover for itself that they are possible causes. This degree of open-endedness will take many years to tackle.

In summary if an approach to self-extension based on self-understanding is to be promising as a long term approach, then we need to find ways of representing and reasoning about much more difficult knowledge gaps. We believe we have developed the first part of such an approach, and that these are indeed challenging, but achieveable goals.

ACKNOWLEDGMENT

The authors gratefully acknowledge support of the EC FP7 IST project CogX-215181.

REFERENCES

[1] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Advanced Engineering Informatics*, vol. 24, pp. 27–39, 2010.

[2] D. Roy, "Semiotic schemas: A framework for grounding language in action and perception," *Artificial Intelligence*, vol. 167, no. 1-2, pp. 170–205, 2005.

[3] R. Engel and N. Pfleger, "Modality fusion," in *SmartKom: Foundations of Multimodal Dialogue Systems*, W. Wahlster, Ed. Berlin: Springer, 2006, pp. 223–235.

[4] H. Jacobsson, N. Hawes, G.-J. Kruijff, and J. Wyatt, "Crossmodal content binding in information-processing architectures," in *Proc. of the 3rd International Conference on Human-Robot Interaction (HRI)*, 2008.

[5] J. Kelleher, "Integrating visual and linguistic salience for reference resolution," in *Proceedings of the 16th Irish conference on Artificial Intelligence and Cognitive Science (AICS-05)*, N. Creaney, Ed., 2005.

[6] E. Punskaya, "Bayesian approaches to multi-sensor data fusion," Master's thesis, Cambridge University Engineering Department, 1999.

[7] A. Pronobis, K. Sjöö, A. Aydemir, A. N. Bishop, and P. Jensfelt, "Representing spatial knowledge in mobile cognitive systems," Kungliga Tekniska Högskolan, CVAP/CAS, Tech. Rep. TRITA-CSC-CV 2010:1 CVAP 316, March 2010.

[8] J. Folkesson, P. Jensfelt, and H. Christensen, "The m-space feature representation for slam," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1024–1035, Oct. 2007.

[9] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *4th International Symposium on Robotics Research*, 1987.

[10] A. Aydemir, A. Bishop, and P. Jensfelt, "Simultaneous object class and pose estimation for mobile robotic applications with minimalistic recognition," in *Proc. of the International Conference on Robotics and Automation (ICRA'09)*, 2010.

[11] González-Banos and Laser, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001.

[12] J. Castellanos, R. Martinez-Cantin, J. Tardos, and J. Neira, "Robocentric map joining: Improving the consistency of EKF-SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, January 2007.

[13] A. N. Bishop and P. Jensfelt, "A stochastically stable solution to the problem of robocentric mapping," in *Proc. of ICRA'09*.

[14] ——, "Stochastically convergent localization of objects and actively controllable sensor-object pose," in *Proc. of the 10th European Control Conference (ECC'09)*.

[15] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. accepted, 2009.

[16] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research (IJRR)*, vol. 29, no. 2-3, pp. 298–320, February 2010.

[17] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. M. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, June 2008.

[18] C. Bäckström and B. Nebel, "Complexity results for SAS^+ planning," *Computiational Intelligence*, vol. 11, no. 4, pp. 625–655, 1995. [Online]. Available: ftp://ftp.informatik.uni-freiburg.de/papers/ki/backstrom-nebel-ci-95.ps.gz

[19] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Journal of Autonomous Agents and Multiagent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[20] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning About Knowledge*. MIT Press, 1995.

[21] H. J. Levesque, "What is planning in the presence of sensing?" in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-96)*. AAAI Press, 1996, pp. 1139–1146.

[22] R. Petrick and F. Bacchus, "A knowledge-based approach to planning with incomplete information and sensing." in *Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS-02)*, 2002.

[23] B. J. Grosz and S. Kraus, "Collaborative plans for complex group action," *Artificial Intelligence Journal*, vol. 86, 1996. [Online]. Available: citeseer.ist.psu.edu/grosz96collaborative.html

[24] K. E. Lochbaum, "A collaborative planning model of intentional structure," *Computational Linguistics*, 1998.

[25] M. Kristan and A. Leonardis, "Multivariate online kernel density estimation," in *Computer Vision Winter Workshop*, 2010, pp. 77–86.

[26] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, 2009.
[27] D. Skočaj, M. Kristan, and A. Leonardis, "Continuous learning of simple visual concepts using Incremental Kernel Density Estimation," in *VISSAP 2008*, 2008, pp. 598–604.
[28] L. P. Beaudoin and A. Sloman, "A study of motive processing and attention," in *Prospects for Artificial Intelligence: Proc. of AISB-93*, A. Sloman, D. Hogg, G. Humphreys, A. Ramsay, and D. Partridge, Eds. Amsterdam: IOS Press, 1993, pp. 229–238.
[29] M. Ozuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[30] M. Hanheide, N. Hawes, J. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, and G.-J. Kruijff, "A framework for goal generation and management," in *Proceedings of the AAAI Workshop on Goal-Directed Autonomy*, 2010.
[31] D. Skočaj, M. Kristan, and A. Leonardis, "Formalization of different learning strategies in a continuous learning framework," in *EPIROB'09*, 2009, pp. 153–160.