A Hierarchical Approach to Active Semantic Mapping Using Probabilistic Logic and Information Reward POMDPs

Tiago S. Veiga, Miguel Silva, Rodrigo Ventura, Pedro U. Lima
Institute for Systems and Robotics, Instituto Superior Técnico
Universidade de Lisboa, Lisboa, Portugal
{tsveiga,rodrigo.ventura}@isr.tecnico.ulisboa.pt
{miguel.oliveira.silva,pedro.lima}@tecnico.ulisboa.pt

Abstract

Maintaining a semantic map of a complex and dynamic environment, where the uncertainty originates in both noisy perception and unexpected changes, is a challenging problem. In particular, we focus on the problem of maintaining a semantic map of an environment by a mobile agent. In this paper we address this problem in an hierarchical fashion. Firstly, we employ a probabilistic logic model representing the semantic map, as well as the associated uncertainty. Secondly, we model the interaction of the robot with the environment with a set of information-reward POMDP models, one for each partition of the environment (e.g., a room). The partition is performed in order to address the scalability limitations of POMDP models over very large state spaces. We then use probabilistic inference to determine which POMDP and policy to execute next. Experimental results show the efficiency of this architecture in real domestic service robotic scenarios.

Introduction

In this paper we present an efficient way to accurately represent cognitive knowledge about the environment through the use of a semantic map (Pangercic et al. 2012). One challenge in the use of semantic maps is to keep an updated information about the environment, as well as, to have a reduced uncertainty about the world state. Moreover, it can also use that information to influence its behavior in order to carry out some tasks.

To address this problem, we propose an hierarchical approach composed of two layers: a (1) execution layer based on sequential decision making under uncertainty models, and a (2) coordination and knowledge representation layer based on probabilistic logic.

At the execution layer, we split the world into regions, dubbed partition, and the problem of decision making into multiple subproblems. Each subproblem is responsible for a partial representation of the global world and to make decisions within the scope of that partition. The decision maker mechanism used to make decisions in each partition is an information-reward POMDP (Spaan, Veiga, and Lima 2015), as it provides a principled framework to model decision making under uncertainty and to probabilistically represent the uncertainties in the system.

At the coordination and knowledge representation layer, we employ a probabilistic logic approach (De Raedt and Kimmig 2015), that keeps an updated belief about the global world state, and uses that information to coordinate the execution layer. Through a probabilistic inference process, the knowledge representation engine is also able to fully generate each POMDP model of the decision maker, given the world model that it receives as input. For simplicity, we will refer to this layer as the knowledge layer throughout the paper.

Inspired by the RoboCup@Home competition scenarios (van der Zant and Wisspeintner 2007), we evaluated an instantiation of this architecture in a domestic scenario. Here, a robot learns a semantic map without prior knowledge and keeps this information updated, even under human-made changes in the environment. We show that we are able to keep a low uncertain information over time, that the system is able to recover from changes in the environment and is robust to false observations.

We start by reviewing the probabilistic programming and decision-making with POMDPs frameworks, then give a general overview of the proposed architecture and more details of its execution. Finally we present the experimental setup and results and draw final conclusion.

Related Work

Semantic mapping was developed as a mean to provide an abstraction of spatial information in robotics and incorporate common-sense knowledge (Kostavelis and Gasteratos 2015). Hierarchical representations are a common approach to establish spatial relations. For instance, objects have properties that distinguish them and can be located at some places. This way, semantic information from sensors was used for navigation tasks (Galindo et al. 2005) or layered representations of knowledge that consider the uncertainty on the dynamics of the system and build probabilistic conceptual maps (Pronobis et al. 2010; Pronobis and Jensfelt 2012). Closer to our work, probabilistic conceptual maps and probabilistic planning have been also combined in object search tasks (Hanheide et al. 2011) that actively reasons about the current environmental information to achieve goals. Probabilistic logic and decision-making were combined to decouple the global information update and the local decision-making processes (Veiga et al. 2016), with application to single regions. In this paper the agent takes into account the current quality of information, and the current system objective, to decide which region of the environment to explore and to plan actions locally without any simplification in the decision-making process, making the decision-making process more scalable. Robot task planning benefits from semantic information, providing the robot with abilities to perform its tasks more efficiently (Galindo et al. 2008). In this work we are able to tackle planning towards a physical goal, as well as planning for information gathering as a mean to improve the current semantic map information.

Background

Probabilistic Logic Programming

The introduction of probabilities in logic programming allows to encode the inherent uncertainty present in real-life situations. Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities, supporting probabilistic inference and learning. We focus on the probabilistic logic programming language ProbLog, which is a probabilistic extension of Prolog.

A ProbLog program has a set of ground probabilistic facts and a logic program, that is, a set of rules and non probabilistic facts (De Raedt, Kimmig, and Toivonen 2007). The last one is the same as in logic programming. A ground probabilistic fact is a fact `f` with no variables and probability `p` and can be written as `p::f`. It is also possible to write an intentional probabilistic fact, which compactly specifies an entire set of ground probabilistic facts. ProbLog also allows the use of annotated disjunctions (Dries et al. 2015), like the sentence `0.15::bird(V); 0.09::mammal(V); 0.5::fish(V) :- vertebrate(V)`, with the structure $\mathtt{p_1} :: \mathtt{h_1}$; ... ; $\mathtt{p_n} :: \mathtt{h_n}$: $-$ `body`.

A ProbLog program specifies a probability distribution over states. Given a set of features of interest, we may define state variables and construct a probabilistic logic program with probabilistic facts for each state variable. With this, and given evidence from the environment, we may compute a probability for all possible values of state variables and, thus, build a belief for the whole system.

Partially Observable Markov Decision Processes (POMDPs) A POMDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, O, T, \Omega, R, h, \gamma \rangle$ (Kaelbling, Littman, and Cassandra 1998). At any time step the environment is in a state $s \in S$, the agent takes an action $a \in A$ and receives a reward $R(s,a)$ from the environment as a result of this action, while the environment switches to a new state s' according to a known stochastic transition model $T : p(s'|s,a)$. After transitioning to a new state, the agent perceives an observation $o \in O$, that may be conditional on its action, which provides information about the state s' through a known stochastic observation model $\Omega : p(o|s',a)$. The agent's task is defined by the reward it receives at each timestep t and its goal is to maximize its expected long-reward $E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where γ is a discount rate, $0 \leq \gamma < 1$.

Given the transition and observation models a POMDP can be transformed to a belief-state MDP in which the agent summarizes all information about its past using a belief vector $b(s)$.The initial state of the system is drawn from the initial belief b_0. Every time the agent takes an action a and observes o, b is updated by Bayes' rule:

$$b_a^o(s') = \frac{p(o|s',a)}{p(o|a,b)} \sum_{s \in S} p(s'|s,a) b(s), \tag{1}$$

where $p(o|a,b)$ is a normalizing constant.

The agent behavior is given by the policy, that maps beliefs to actions. The value of a policy π is defined by a value function V:

$$V^*(b) = \max_{a \in A} \Big[\sum_{s \in S} R(s,a) b(s) + \gamma \sum_{o \in O} p(o|b,a) V^*(b_a^o) \Big], \tag{2}$$

with b_a^o given by (1). The optimal policy that maximizes the value function is denoted as π^*. The POMDP value function has been proven to be piecewise linear and convex (PWLC), which is used by most algorithms to efficiently represent it.

Factored models are used and exploited in POMDP representation and solving. By factorizing the state, observation and/or actions spaces we may represent these spaces as a cross-product of variables.

POMDPs for Active Perception In the classical POMDP definition rewards are defined over states and actions. However, when transforming it in a belief-MDP the current reward is given by the expected immediate reward for the current belief:

$$r(b,a) = \sum_{s \in S} b(s) R(s,a) \tag{3}$$

From the perspective of active perception, as the belief is a probability distribution over the state space, it is natural to define the quality of information based on it. We could use the belief to define a measurement of the expected information gain when executing an action. However, with reward models directly defined over beliefs the value function is no longer PWLC. Some extensions to the classic POMDP formulation allow to formulate POMDP models for information gain and keeping PWLC value function, such as POMDP-IR (Spaan, Veiga, and Lima 2015) and ρPOMDP (Araya et al. 2010).

We use the POMDP-IR formulation, which extends the original action space with information rewarding actions. These actions do not influence the physical state of the environment and are run in parallel with the original domain actions, but appropriately reward the system to reach low-uncertain levels of belief for features of interest in the environment.

Architecture Description

The proposed approach includes a coordination and knowledge representation layer, that infers its knowledge from observations perceived by the agent's sensors, and an execution layer, that models the interaction between the agent and the environment given the current knowledge in the knowledge representation layer. To overcome scalability issues the world is divided into partitions and a set of POMDP models are derived, one for each partition. Overall, both layers complement each other and provide a more efficient and scalable framework for object search in an environment.

The world model must be compatible between both layers to ensure that they can effectively communicate and get information from the environment. A consequence is that the POMDP generation is made through the upper layer taking into account the global world model. Also, at execution time, the knowledge engine must correctly select which partition of the environment to explore and, therefore which POMDP to select. This means that there are two levels of decision: a macro level in which the knowledge engine selects one POMDP (this is equivalent to decide which partition of the environment to explore) and a micro level in which each POMDP drives the agent through the environment.

In the following of this section we will describe in detail each one of these layers and how they tackle these challenges.

Coordination and Knowledge Representation Layer

The knowledge layer generates a knowledge base from an initial world model that consists of a list of instances, their properties and relations between them. Instances are any kind of physical entity existing in the environment, which can have relations between them. For instance, in a domestic semantic mapping context this can be considered as a list of objects, furniture and rooms with their characteristics, such as position, volume, size and others.

This is then used by a reasoning engine, based in the ProbLog language, to create a set of rules and probabilistic facts that will later be used to perform inference with the observations perceived from the environment and to generate models for the decision maker.

The knowledge layer considers that, at each time, a state S can be defined as the joint discrete probability distribution of a set $\mathcal{X} = \{X_1, X_2, \ldots, X_{K_T}\}$ of independent discrete random variables. Each state S in the state space $\mathcal{S}$ can be defined as:

$$S = X_1 \times X_2 \times \cdots \times X_i \cdots \times X_{K_T} \tag{4}$$

Each state variable X_i can take any value in its domain D_i. Then, the size of the state space $|\mathcal{S}|$ is equal to the combinations of the domain D_i of each state variable X_i,

$$|\mathcal{S}| = \prod_{i=1}^{K_T} D_i. \tag{5}$$

Execution Layer

The execution layer implementation is based on the POMDP framework and provides the system with intelligent decisions to efficiently construct a semantic map. On the other hand, the state knowledge representation of POMDPs is expanded by the knowledge layer.

Each POMDP n of the execution layer has a set of states $\mathcal{S}^n$. Each state S^n in $\mathcal{S}^n$ is defined as the joint discrete probability distribution of a set $\mathcal{X}^n$ of independent discrete random variables. It is important to notice that,

$$|\mathcal{X}^n| \leq |\mathcal{X}| \tag{6}$$

and that each variable $X_i^n \in \mathcal{X}^n$ has a match with the variable $X_i \in \mathcal{X}$, because they are representing the same feature in the world model. However, they are not the same because they have different domains. The domain of X_i^n is D_i^n and it is adapted to the partition of the respective POMDP. It should be noted that there is an important characteristic of the relation between X_i and X_i^n domains, given by Equation (7), because the partition in each POMDP is restricted with respect to the global world.

$$|D_i^n| \leq |D_i|, \tag{7}$$

Conditions presented in Equations (6) and (7) are the reason for the dimension of $\mathcal{S}^n$ to be smaller than $\mathcal{S}$.

To construct the domain of the variables D_i^n in a POMDP, domain values that are not available on that specific subworld must be represented because those values are still valid in the global world representation. For that reason, all those values can be aggregated in a single value, here designated as *none*. It represents those values, but does not discriminate each one individually, minimizing the number of POMDP states, as desired. Every time the knowledge layer calculates the belief b^n of each POMDP *n* it generates the new probability distribution for each state variable X_i^n of POMDP *n*. For that purpose, a function $f_{n,i}$ for each state variable X_i^n of each POMDP is considered, that associates each element of the domain D_i to a single element of the domain D_i^n,

$$f_{n,i} : D_i \rightarrow D_i^n. \tag{8}$$

If $D_i = D_i^n$, $f_{n,i}$ is an endofunction, however, the most common case is to have $D_i^n \subseteq D_i \cup \{none\}$ where *none* represents the set of elements $D_i \setminus D_i^n$ and then

$$P(X_i^n = x_i^n) = \begin{cases} \sum_{x_i \in D_i \setminus D_i^n} P\left(X_i = x_i\right) & \text{, if } x_i^n = none \\ P\left(X_i = x_i^n\right) & \text{, otherwise} \end{cases} \tag{9}$$

Execution Flow

In this section we outline key features of the proposed approach, in particular automatic generation of POMDP models, POMDP selection and update of the global belief in the coordination layer.

At execution time, the knowledge layer keeps a global probabilistic representation of the world state, initialized as a uniform distribution, and decides which partition of the environment to explore by selecting a POMDP to which the control of the agent is assigned. This POMDP executes until some stopping criteria is met and returns the received observations to the knowledge layer, that updates the global belief and again selects the next partition to explore. A schematic of the execution flow is outlined in Figure 2.

POMDP generation

At initialization, the system receives a world model from the system designer that is read by the knowledge layer and

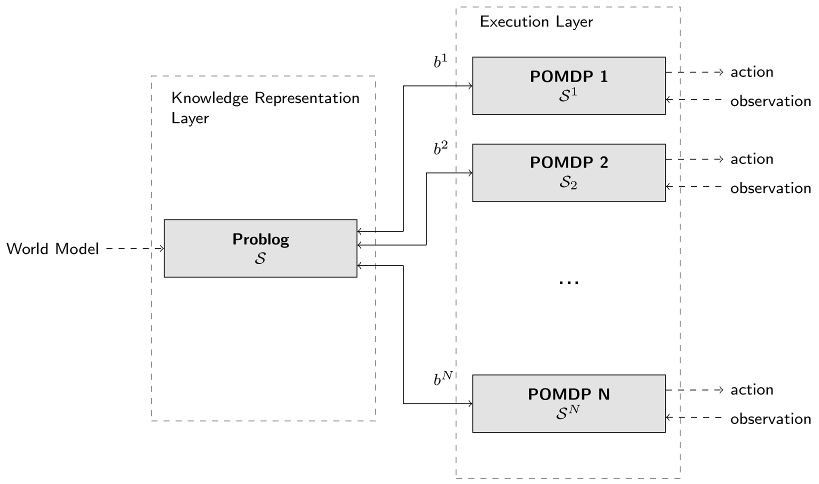


Figure 1: Proposed architecture. $\mathcal{S}$ is the global state space, $\mathcal{S}^i$, $i = 1, \ldots, N$ and b^i, $i = 1, \ldots, N$ are, respectively, the state spaces and beliefs for each partition of the environment.

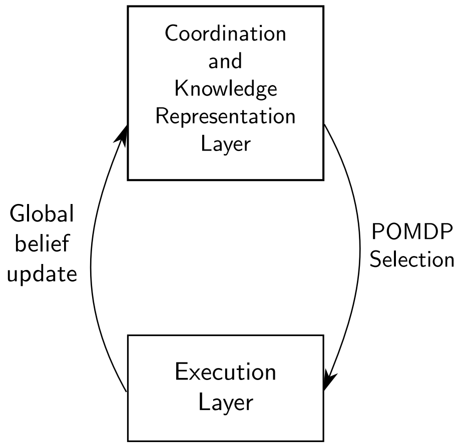


Figure 2: Execution flow of our approach. Execution layer interacts with the environment while the Coordination and Knowledge Representation Layer keeps a higher level of abstract information.

fed into a POMDP generator, which automatically builds the POMDP models for each partition. Finally, optimal policies for those POMDPs are computed by POMDP solving algorithms.

The partition criterion is defined *a priori*. Each tuple $\langle \mathcal{S}^n, \mathcal{A}^n, O^n, T^n, \Omega^n, R^n \rangle$, that defines the POMDP *n*, is completely defined by the knowledge layer. This partition into smaller POMDP models reduces the computational complexity of computing policies. The dimension of $\mathcal{S}^n$ for each POMDP is smaller than $\mathcal{S}$, that considers all the possible world states.

POMDP selection

A macro decision is made every time the robot finishes exploring a region of the environment. The knowledge layer must decide which region to explore next, taking into account the global belief *b*. For that, it receives as inputs the value functions of each POMDP model ($V_1, \ldots, V_N$) and the current belief (b_t) and compares how beneficial it is to follow the POMDP of each region. Because all POMDPs are generated through the same engine in the knowledge layer we ensure that the structure of the model is the same across different POMDPs we the performance of optimal policies for different partitions are comparable, according to some metric defined according to the system's main goal. In this paper we focus on two main goals: *i)* actively learn or update information on a semantic map; *ii)* carry out a particular physical task; the selection scheme is resumed in Algorithm 1.

```
**Algorithm 1** POMDP selection algorithm
**Input:** $V_1, \ldots, V_N, b_t$
  **if** objective is carry task **then**
    $n^* = \text{argmax}_n V_n(b_t^n)$
  **else if** objective is information gain **then**
    $n^* = \text{argmin}_n V_n(b_t^n)$
  **end if**
  **return** $n^*$
```

The first is an instance of an active perception scenario, meaning that the agent should prefer to explore regions with less certain information. Because we use the POMDP-IR framework, that rewards the model positively to achieve low uncertainty beliefs, the value function is an indirect measure of the quality of information in the belief. Also, the POMDP policy maximizes the value function in the long run and, due to the convexity property of the value function, it drives the belief to regions on the extremes of the belief simplex. Therefore, choosing the POMDP with the lowest value for the current belief means that the potential information gain will be higher. An alternative to this could be to choose the POMDP of the room with highest belief entropy, but then we would not take into account the dynamics of the model to gather information, which is indirectly represented in the value function.

If the goal is to perform a physical defined task, such as carrying an object from one point to the other, than the best option is to select the POMDP with the highest value function for the current belief, similarly to auctioned POMDPs (Capitán et al. 2013).

Global belief update

While exploring the environment the global belief is updated in the knowledge layer with observations communicated by the execution layer. The belief is first smoothed through an exponential decay and then updated through an inference process that uses the received observations as evidence to the probabilistic logic program. The probability distribution of each variable X_i is smoothed as in Equation (10), where $P_{previous}(X_i = x)$ is the value of the probability distribution of X_i at the previous time step, $P_{uniform}(X_i = x)$ is the probability value for a uniform distribution and λ is the decay rate.

$$\begin{aligned} P\left(X_i = x_i\right) &= P_{uniform}\left(X_i = x_i\right) + \\ &+ \left[P_{previous}\left(X_i = x_i\right) - P_{uniform}\left(X_i = x_i\right)\right] e^{-\lambda t} \end{aligned} \tag{10}$$

Then, the observation o is used in an inference process to complete the global belief update. The global belief b and the belief of partition n, b^n, are defined as the joint probability distribution of the set of state space variables $\mathcal{X}$ and $\mathcal{X}^n$. Given the independence between state variables in $\mathcal{X}^n$ and in $\mathcal{X}$, updating the belief b with the new belief b^n is equivalent to update the probability distribution of each variable X_i, given the probability distribution of the respective X_i^n, and then calculate the joint probability distribution of the variables updated in the set $\mathcal{X}$. The probability distribution of the state variables X_i remains the same when there is no correspondent X_i^n in the set $\mathcal{X}^n$ and the remain variables X_i are updated individually, taking into account the probability distribution that comes up from the belief of POMDP n. This probability represents the posterior of the variable X_i^n given the current belief an observation o. Considering $P\left(x_i\right) = b_{t-1}(x_i)$ as the prior probability distribution of X_i, the posterior probability $P\left(x_i|b_{t-1}, o\right) = b_t(x_i)$ can be derived to the Equation (11), where $f_{i,n}$ is the function that associates each element in D_i to an element in D_i^n, for the POMDP n, t and $t-1$ are, respectively, the current and previous time step.

$$P\left(x_i|b_{t-1}, o\right) = \frac{P\left(x_i\right) P\left(x_i^n|b_{t-1}^n, o\right)}{\sum\limits_{x_i \in D_i} P\left(x_i^n|x_i\right) P\left(x_i\right)} \tag{11}$$
$$, \text{with } x_i^n = f_{i,n}\left(x_i\right)$$

Considering that $P\left(x_i^n|x_i\right)$ is given by Equation (12),

$$P\left(x_i^n|x_i\right) = \begin{cases} 1 & , \text{if } x_i^n = f_{i,n}\left(x_i\right) \\ 0 & , \text{otherwise} \end{cases} \tag{12}$$

then, Equation (11) can also be written as in (13).

$$P\left(x_i|b_{t-1}, o\right) = \begin{cases} P\left(x_i^n|b_{t-1}^n, o\right) & , \text{if } x_i \in D_i \cap D_i^n \\ \dfrac{P\left(x_i\right) P\left(x_i^n|b_{t-1}^n, o\right)}{\sum\limits_{x_i \in D_i \setminus D_i^n} P\left(x_i\right)} & , \text{otherwise} \end{cases} \tag{13}$$

where $P\left(x_i^n|b_{t-1}^n, o\right) = b_t^n\left(x_i^n\right)$ is the current belief of POMDP n. This ensures that the belief of the value *none* in each partition belief updates proportionally the values on the global belief that were aggregated in there.

Experimental Setup

For experiments we considered a robot in a domestic environment, where it must locate some objects and, possibly, perform some tasks with them. A map of the environment is presented in Figure 3

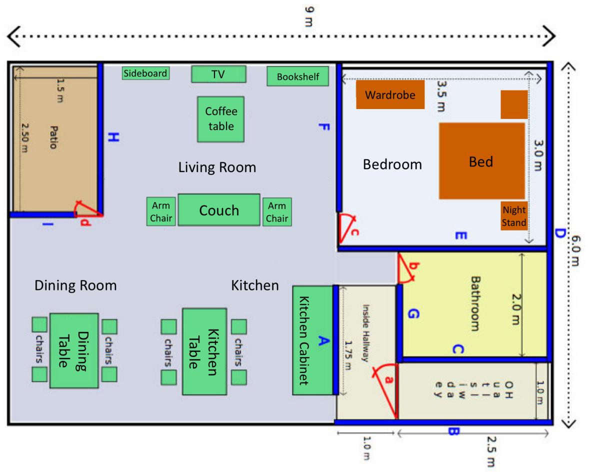


Figure 3: Map of the domestic environment used in our experimental results.

Model

The world model consists of an apartment scenario with all its rooms, furniture, objects and their respective characteristics and relations. Each POMDP model of the execution layer is defined as follows.

States and transitions The model includes one state variable for the position of the robot and one state variable for the position of each object that can be located in the room. Objects can be located in any of k waypoints through which the robot navigates. For the purpose of our task we consider a simple deterministic state transition model both for robot movement and object location, although POMDPs generalize for more complex dynamics. In practice, this means that we consider that the robot's navigation is always successful and that objects remain static in their initial position during each run. Those are realistic assumptions given that, in one hand, nowadays there are reliable and accurate navigation algorithms that work well in our kind of environments and can be separated from the decision-making task. In the other hand, in these environments it is not expected that objects are relocated too often while the robot performs a search task and, even if it happens, it will be detected in the following searching episode. Moreover, the knowledge layer already includes a decay in the belief update.

Observations There is an observation variable for each object of interest ($1 \leq i \leq m$). Every time a perception action is triggered each of the observation variables takes a value *yes* or *no*, indicating whether that object was detected by the onboard camera or not. Depending on the distance between placements we assign a probability of observing an object in a location different from the actual one that decays with the distance to the current location. Furthermore, volume of objects are also considered and larger objects have higher probability of being correctly observed.

Actions Following the POMDP-IR framework used, there are two kind of actions: domain A_d and information-rewarding A_{IR_i} actions. Domain actions are those which physically interact with the environment. In our model it includes moving actions, that guide the robot through the different locations of interest in the environment, a perception action that uses the camera to detect objects in the actual location, and an additional *doNothing* action that indicates the end of a searching episode. The set of information-rewarding actions include one for each object of interest. This extra actions indicate whether an object is believed to be in some location in the actual room, not found in this room, or null if there is not enough information.

Rewards The set of rewards used follow the set of actions previously mentioned. Therefore, the model includes a cost for moving actions, based on the distance between waypoints, and a cost for perception actions. There is neither a cost nor reward for *doNothing* action, to motivate the end of an episode every time the agent achieved a low-uncertainty information. For information-gaining actions we include rewards that are positive if the belief for a particular state variable is above a threshold of $\beta = 0.9$. This will guide the system to increase the available information about object locations.

Results

In our set of experiments we use the Hellinger distance between the current belief of the system and a perfect belief, that is, the belief which would be 1 for the true state of the environment and 0 otherwise, as a metric for the quality of the information. For two discrete probability distributions $U = (u_1, ..., u_n)$ and $V = (v_1, ..., v_n)$, the Hellinger distance is defined as

$$H(U,V) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{u_i} - \sqrt{v_i})^2}, \qquad (14)$$

A possible alternative would be to measure the entropy of the current belief but, in that case, we would be measuring only how much informative is the belief, without considering how close it would be to the true state of the environment. Instead, we can do that with the Hellinger distance.

Semantic mapping [1]

In the first set of experiments, we tested the framework in a domestic semantic mapping scenario in which the system wants to learn the position of a set of objects in the environment and, subsequently, keep track of their location. The model considers a set of possible locations for the objects based on the furniture existing at the house and received initially through the world model. These locations are detailed in Table 1.

Kitchen	Living Room	Dining Room	Bedroom
kitchen table kitchen cabinet	coffee table sideboard bookshelf	dining table	bed nigh stand

Table 1: Placements in the semantic map experiments.

Static objects First, we test with objects that remain static throughout the run. The initial distribution of each object state variable is uniform, therefore the initial Hellinger distance is 0.8 for all objects. Figures 4a and 4b show, respectively, the evolution of the Hellinger distance for the belief of each object and the value function for each partition of the map at each decision step.

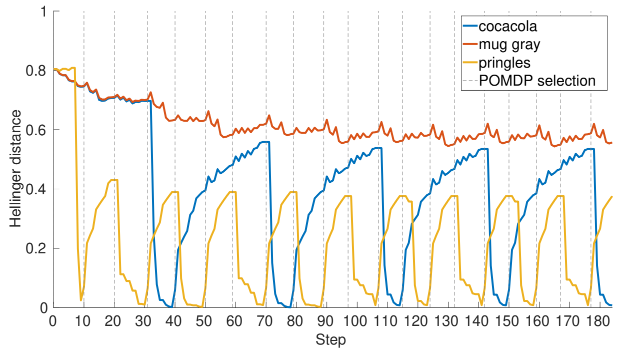
The agent moves between the living room, bedroom and kitchen, while it never visits the dining room. Based on the information it has from the rest of the placements, the agent is able to infer whether objects are in the dining room because there is just one placement there. That is why the value of the policy for the dining room is never the lowest of all rooms. On the other hand, if the agent decided to explore the dining room, the belief of *mug gray* would increase while the belief of the remain objects would decrease due to belief decay, leading the global belief farther from a perfect belief and, therefore the whole system with more uncertain information.

Dynamic objects We repeated the same initial configuration but now changing objects' location along the run. The true localization of objects are presented in Table 2.

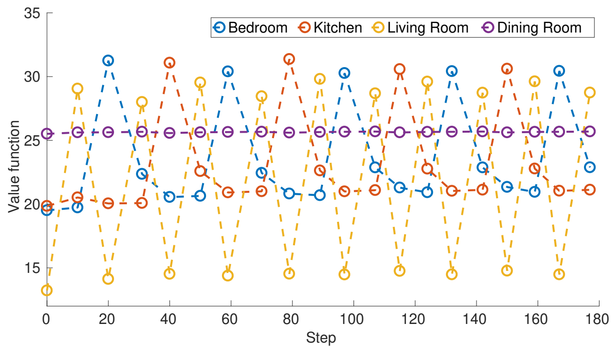
Figures 4c and 4d present the Hellinger distance between the current and perfect beliefs and the value function at decision steps.The robot visits the living room more often because it is the room with more placements and the belief decays overtime. Again, the Hellinger distance decreases over time as the robot explores the environment. Note that initially the information about the *mug gray* does not decrease at the same pace as the other objects, due to the fact that it is located in the dining room. The belief decay leads the belief of well localized objects to spread before each decision step of the coordination layer and the system always decides to explore other rooms with more placements and, therefore, more potential to gather information.

In general, after every modification in objects location there is a peak in the Hellinger distance, meaning that the be-

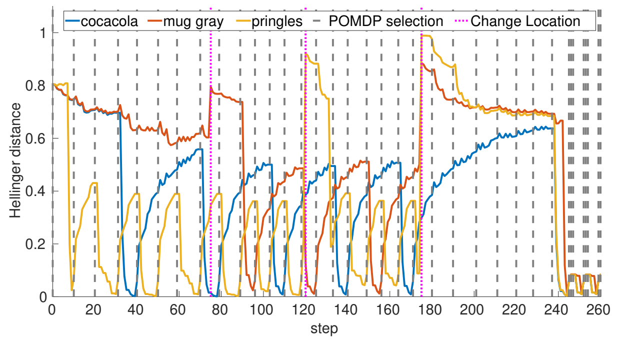
[1]Additional video experiments may be found at https://youtu.be/eLUWA_QkVuo.



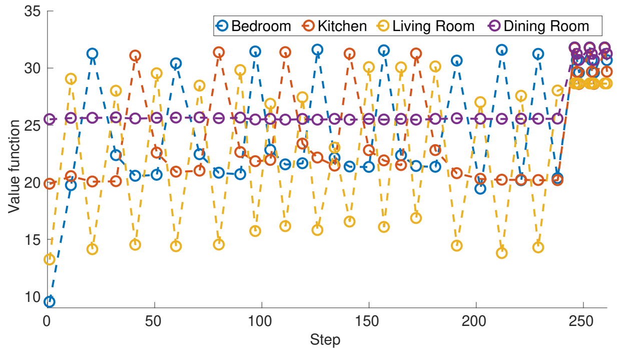
(a) **Static objects**: Hellinger distance.



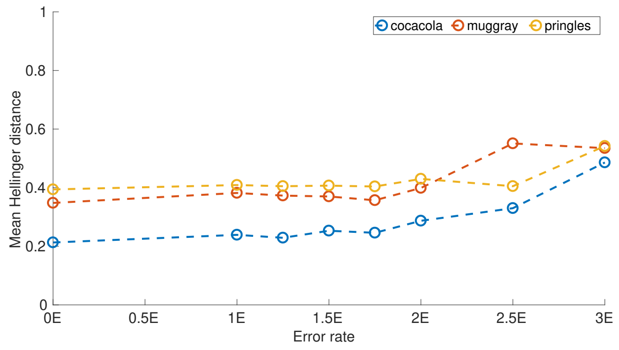
(b) **Static objects**: value function at decision steps.



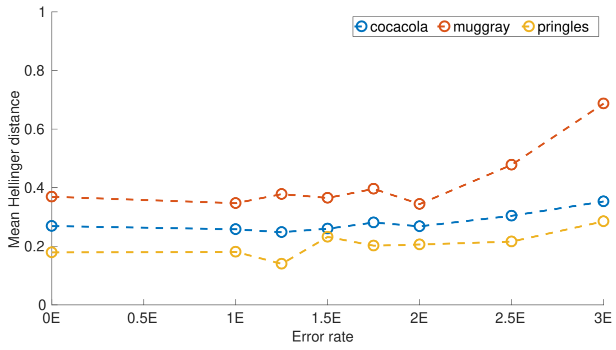
(c) **Dynamic objects**: Hellinger distance.



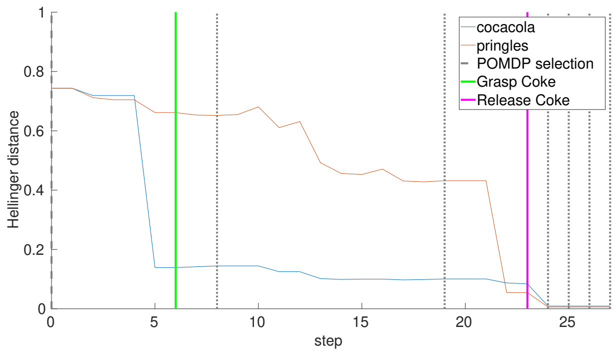
(d) **Dynamic objects**: value function at decision steps.



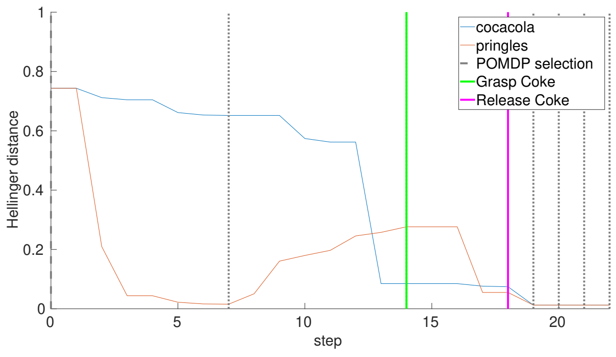
(e) **Incorrect observations robustness**: mean Hellinger distance over 200 steps. E is error rate in predicted in observation model.



(f) **Incorrect observations robustness**: Hellinger distance over 200 steps. E is error rate in predicted in observation model.



(g) **Carrying out a task**: Hellinger distance.



(h) **Carrying out a task**: Hellinger distance.

Figure 4: 4a,**4b**,**4c**,**4d** Mug gray: *dining table*, Coca cola: *kitchen table*, Pringles: *coffee table*, **4e** Mug gray: *bed*, Coca cola: *coffee table*, Pringles: *kitchen cabinet*, **4f** Mug gray: *night stand*, Coca cola: *bookshelf*, Pringles: *sideboard*, **4g** Coca cola: *sideboard*, Pringles: *dining table*., **4h** Coca cola: *kitchen table*, Pringles: *coffee table*.

steps	object	location
0-75	mug_gray	dining table
	cocacola	kitchen table
	pringles	coffee table
76-120	mug_gray	night_stand
	cocacola	kitchen table
	pringles	coffee table
121-175	mug_gray	night_stand
	cocacola	kitchen table
	pringles	bookshelf
176-260	mug_gray	kitchen cabinet
	cocacola	kitchen table
	pringles	kitchen table

Table 2: Human changes in the location of objects during the *Dynamic objects* experiment.

lief is inaccurate, but recovered after some steps. A final note for the behavior in the last steps of the experiment, when all objects are located in the same room. Due to this fact the system receives newer observations of the objects before the decay spreads the belief too much and, therefore, beliefs are more accurate in this case.

Incorrect observations robustness Finally, we tested the robustness against incorrect observations. Despite the expected error rate in the observation model, conditions may change overtime and in the long run different error rates arise. Therefore, we executed tests for two different configurations of objects location with false positive and negative rates different from the ones predicted in the model. Figures 4e and 4f show the mean Hellinger distance for a run of 200 steps. With actual error rates up to 2 times higher than the predicted in the observation model the system is still able to maintain a mean Hellinger distance over the episode similar to the original one. This shows that even with real conditions different from the ones predicted in the model the system is still able to perform well and maintain a good information about the environment state.

Carrying out a task

As mentioned this approach is also useful in situations where the goal of the agent is to carry out a specific task. In this case the agent will do information gathering only if that is useful for task performance. We tested this behavior in a scenario in which the robot is instructed to move the *cocacola* close to the *pringles*, without a priori knowledge of the localization of both objects. Two different initial configurations of the location of objects were considered, with the placements in Table 3.

Kitchen	**Living Room**	**Dining Room**	**Bedroom**
kitchen table	coffee table	dining table	
kitchen cabinet	sideboard		

Table 3: Placements in the *carrying out a task* experiments.

Here, the partition to explore is select from the POMDP with the highest value function for the current belief. In Figure 4g the robot finds the *cocacola* in the first room that it visits, grasps it and keeps it in the gripper until it finds the *pringles* that are in the dining room. After grasping the *cocacola* it first visits the kitchen because there are two placements and, therefore, the probability of finding objects there is higher, which implies that the expected rewards are also higher. In Figure 4h, the robot first finds the *pringles* and, as soon as it finds the *cocacola*, goes back to the placement where the *pringles* were found in order to reach the goal.

Discussion

In general, the agent has an active behavior that keeps the level of uncertainty reduced. When there is any change in the true state of the environment, our approach is able to minimize the uncertainty about the world state in a short amount of time and in different conditions: moving the object to a placement inside the same room or to a different room, moving one or multiple objects simultaneously, etc. Our approach deals well with incorrect observations, even when error rates are as high as twice the expected by the model.

We observed behaviors such as deciding to explore more often rooms with more placements because there is a higher chance of finding them there. The *dining room* only has one placement and the system can infer the probability of the object being there without actually visiting it.

Conclusions

In this work we present an integrated and efficient approach to represent cognitive knowledge about uncertain environments. By exploiting the advantages of probabilistic programming and decision-theoretic planning an hierarchical architecture is able to mitigate the disadvantages of both. With this architecture POMDP models are automatically generated for the semantic mapping application using the world model as an input to infer the states, observation, transition and rewards function of each POMDP, thus avoiding the explicit declaration of the model.

We show the approach performance in an application to a semantic map scenario, in which the system is able to maintain a probabilistic representation of the location of objects, even after human changes in the environment and, if needed, use that information to carry out a task. The presented architecture is also robust to incorrect observations.

In future work, we would like to extend this work to multiple robots exploring multiple rooms, where cooperation must be taken into account. Also, learning the model, in particular the transition, observation and reward functions directly from the environment through inverse reinforcement learning processes can lead to better representation of the environment model.

Acknowledgments

This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) through the strategic project UID/EEA/50009/2019, and by the European Union's Horizon 2020 research and innovation program under grant agreement No 825619.

References

Araya, M.; Buffet, O.; Thomas, V.; and Charpillet, F. 2010. A POMDP Extension with Belief-dependent Rewards. In *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc. 64–72.

Capitán, J.; Spaan, M. T. J.; Merino, L.; and Ollero, A. 2013. Decentralized multi-robot cooperation with auctioned POMDPs. *International Journal of Robotics Research* 32(6):650–671.

De Raedt, L., and Kimmig, A. 2015. Probabilistic (logic) programming concepts. *Machine Learning* 100(1):5–47.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. 7:2462–2467.

Dries, A.; Kimmig, A.; Meert, W.; Renkens, J.; Broeck, G. V. D.; Vlasselaer, J.; and Raedt, L. D. 2015. ProbLog2 : Probabilistic logic programming. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III*. 312–315.

Galindo, C.; Saffiotti, A.; Coradeschi, S.; Buschka, P.; Fernández-Madrigal, J. A.; and González, J. 2005. Multi-hierarchical semantic maps for mobile robotics. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (3):3492–3497.

Galindo, C.; Fernández-Madrigal, J.-A.; González, J.; and Saffiotti, A. 2008. Robot task planning using semantic maps. *Robot. Auton. Syst.* 56(11):955–966.

Hanheide, M.; Gretton, C.; Dearden, R.; Hawes, N.; Wyatt, J.; Pronobis, A.; Aydemir, A.; Göbelbecker, M.; and Zender, H. 2011. Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. *IJCAI International Joint Conference on Artificial Intelligence* 2442–2449.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Kostavelis, I., and Gasteratos, A. 2015. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems* 66:86–103.

Pangercic, D.; Pitzer, B.; Tenorth, M.; and Beetz, M. 2012. Semantic Object Maps for robotic housework - Representation, acquisition and use. *IEEE International Conference on Intelligent Robots and Systems* 4644–4651.

Pronobis, A., and Jensfelt, P. 2012. Large-scale semantic mapping and reasoning with heterogeneous modalities. *Proceedings - IEEE International Conference on Robotics and Automation* 3515–3522.

Pronobis, A.; Sjöö, K.; Aydemir, A.; Bishop, A. N.; and Jensfelt, P. 2010. Representing spatial knowledge in mobile cognitive systems. *Intelligent Autonomous Systems 11, IAS 2010* 133–142.

Spaan, M. T.; Veiga, T. S.; and Lima, P. U. 2015. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems* 29(6):1157–1185.

van der Zant, T., and Wisspeintner, T. 2007. *Robotic Soccer*. Itech Education and Publishing. chapter RoboCup@Home: Creating and Benchmarking Tomorrows Service Robot Applications, 521–528.

Veiga, T. S.; Miraldo, P.; Ventura, R.; and Lima, P. U. 2016. Efficient object search for mobile robots in dynamic environments: Semantic map as an input for the decision maker. *IEEE International Conference on Intelligent Robots and Systems* 2016-Novem:2745–2750.