

# Semantic Object Search in Large-scale Indoor Environments

Manabu Saito, Haseru Chen,  
Kei Okada, Masayuki Inaba  
Johou Systems Kougaku Laboratory  
The University of Tokyo  
{saito, chen, k-okada, inaba}@jsk.t.u-tokyo.ac.jp

Lars Kunze, Michael Beetz  
Intelligent Autonomous Systems Group  
Technische Universität München  
{kunzel, beetz}@cs.tum.edu

**Abstract**—Many of today’s mobile robots are supposed to perform everyday manipulation tasks autonomously. However, in large-scale environments, a task-related object might be out of the robot’s reach, that is, the object is currently not perceivable by the robot. Hence, the robot first has to search for the object in its environment before it can perform the task.

In this paper, we present an approach for object search in large-scale environments using different search strategies based on semantic environment models. We demonstrate the feasibility of our approach by integrating it into a robot system and by conducting experiments where the robot is supposed to search for objects within the context of fetch-and-delivery tasks within a multi-level building.

## I. INTRODUCTION

Autonomous mobile robots that are to perform everyday manipulation tasks need the appropriate capabilities to obtain task-related objects from the environment. That is, object search is a prerequisite to autonomous object manipulation.

When we look at the performance of humans in object search tasks, it seems, that in most cases humans can find objects in their environment relatively effortless and at very high success rates. This is not only because humans have excellent perception capabilities, but also because humans can rely on a large body of commonsense knowledge to conduct the search for objects more effectively.

Although perception plays a key role in object search within robotics, it is also very important to employ semantic knowledge to narrow down the search space, especially in large-scale environments. Pruning the search space is mainly important because of two reasons, first, robot perception is computational expensive, and second, if robots have no clue about potential object locations, objects will only be found by employing exhaustive search methods or by chance.

In the present work, we investigate the problem of how robots can use semantic environment models to perform object search tasks in large-scale environments more effective and efficient. The contributions of this work are as follows. First, we extend the representations and reasoning methods for semantic maps that have been introduced in [1] in order to account for large-scale indoor environments, i.e. multi-level buildings. Second, we utilize commonsense knowledge acquired from Internet users to bootstrap probabilistic models about typical locations of objects which can be updated during the robot’s lifetime according to its observations. Third, we present several object search strategies using the above models while also considering the current context

of the robot. Finally, we integrate the developed methods within a robot system and provide experimental results for object search in fetch-and-delivery tasks within a multi-level building. Additionally, we show how the semantic search for objects is integrated into an iPad user interface.

In the remainder of the paper we first describe the fetch-and-delivery scenario in more detail in Section II. Then, we explain how we represent the semantic environment models in Section III, and how we use them within various search strategies in Section IV. The integration of these methods with a robot system is presented in Section V. Experimental results are provided in Section VI. Finally, we put the paper into the context of related work in Section VII, before we conclude in Section VIII.

## II. THE FETCH-AND-DELIVERY SCENARIO

In fetch-and-delivery tasks robots are, for example, supposed to serve food and drinks, deliver letters, or collect used cups from workplaces and take them to the kitchen. An essential sub-task of such tasks is to search for the task-related objects in the environment. In this paper, we investigate two scenarios of object search in the context of fetch-and-delivery tasks. In the first scenario the robot is supposed to find cups in its environment and bring them back to its starting position. In the second scenario, we ask the robot to get us a sandwich.

Let’s first look at the cup scenario in more detail. For example, consider a situation where a robot is supposed to fetch a particular cup, let’s say Michael’s cup. In order to find the cup in its environment the robot first replaces the possessive attribute with one or more perceptual attributes, e.g., *Owns(Michael, Cup)* is replaced by *HasLogo(Cup, PR2)*. Either the robot knows about the perceptual attributes of the cup because of its own perceptual experience or this information has to be provided somehow externally. Then the robot tries to retrieve the set of known cup instances from its belief state that fulfill the partial description, e.g. *HasLogo(Cup, PR2)*, or at least, do not contradict it. However, if the robot does not know about any cup instance in the environment, it has to rely on more general knowledge about cups. For example, the robot could infer that cups are typically stored in the cupboards of a kitchen and that they are occasionally located in a dishwasher. Furthermore, if we assume that similar objects are placed next to each other, the robot could reason that cups are similar to glasses

and that therefore the robot could try to find the cup nearby instances of glasses it knows about. In the next step, the robot retrieves the poses of the potential cup locations as well as the poses from which the cups might be perceivable from the semantic map. The latter poses are used as goal locations for the autonomous navigation of the robot. In order to find the cup, the robot moves to the respective goal locations while taking path costs (or the expected rate of success) into account. Having reached a potential cup position the robot uses perception routines for detecting and localizing the cup in the environment. If a cup is detected and fulfills the partial object descriptions the robot applies more specialized perception routines in order to gather all relevant information needed for effectively manipulating the cup. However, if no cup was found, the robot will repeat the procedure until it has searched the remaining potential cup locations.

In the second scenario, the robot is supposed to find and deliver a sandwich. Obviously, similar search strategies like explained above can also be employed. For example, the robot can infer that sandwiches are perishable and that therefore they are typically stored in a fridge.

However, with this example we want to point to the problem that an object might even not exist at the time when looking for it. Some objects in our daily life will only come into existence if we trigger the right processes. For example, food items like sandwiches are created in meal preparation tasks. Mostly these tasks or processes which create instances of certain object types take place at specific locations, e.g., meal preparation tasks are typically carried out in a kitchen or a restaurant. Having this kind knowledge, the robot can go to the designated places and trigger the appropriate processes. Our point here is, that knowledge about the environment is not only beneficial for limiting the search space but in some cases it is even a necessary condition to locate objects.

### III. SEMANTIC ENVIRONMENTS MODELS

In this section, we explain the underlying representations and the knowledge that is represented in the semantic environment models. Figure 1 gives an overview of the different ontological concepts and relations, whereby we distinguish mainly between three types of relations: assertional, computable, and probabilistic. In the following we first explain the ontological representation and afterwards we elaborate on the acquisition and formalization of a probabilistic model for commonsense reasoning about object locations.

#### A. Semantic Maps of Large-scale Environments

In this work, we basically build on the representation formalisms as described in [1]. The underlying representation is based the Web Ontology Language (OWL). In OWL, classes are defined within a taxonomy and derive all the properties of their super-classes. Relations between classes are described by properties. Objects are represented as instances of classes and can also have relations with other instances. The upper ontology is derived from OpenCyc<sup>1</sup>.

With respect to semantic environment maps, this for-

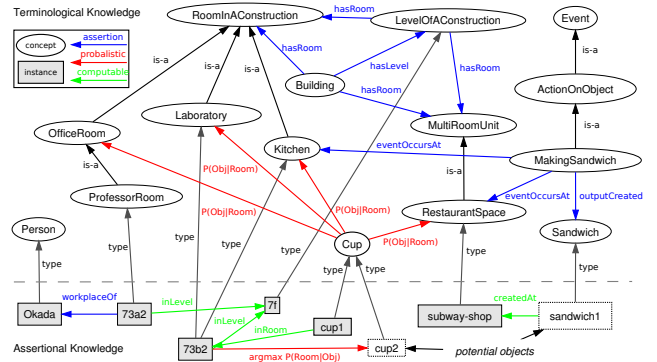


Fig. 1. Simplified overview of concepts and relations of the semantic model for large-scale environments.

malism allow us to structure the knowledge about classes hierarchically, e.g. a *ProfessorOffice* is an *Office* which is a *RoomInAConstruction*. Given the mechanism of (multiple-) inheritance a class derives all the properties of its super-class(es), e.g., *ProfessorOffice* have also the property *roomNumber* since it has been defined for *RoomInAConstruction*. Objects in the map are described by instances and have a certain *type*, e.g. *Office*, properties to other instances, e.g. *workplaceOf*, and simple data properties like *widthOfObject*. The spatial extensions of an object are described by their bounding box, i.e. depth, width, and height. Although this representation is very simple, it allows us to reason about several spatial relations between objects like *in-Container* or *on-Physical* more efficient. Furthermore, objects are related to instances of perception events. These events contain information about the object's pose at a point in time. Thereby we can track the pose of an object over time and answer questions like where was an object five minutes ago.

In [1], the map representation is mainly focused on small environments like rooms, especially kitchens. However, in the context of fetch-and-delivery tasks it is important to represent also the knowledge about environments at a larger scale. Therefore we introduced concepts like *Building*, *LevelOfAConstruction*, *RoomInAConstruction*, and *Elevator*. Figure 1 show a small excerpt of the extended ontology. Additionally, we introduced properties like *floorNumber*, *roomNumber*, *inLevel*, *inRoom*, and *workplaceOf*. Whereas properties like *roomNumber* are directly asserted to particular instances (assertional property), other properties like *inRoom* are computed based on the actual physical configuration of the environment (computable property), i.e. the current pose and dimensions of an object are taken into account. Thereby we can infer whether spatial relations between objects like *in* and *on* hold at some point in time.

In this work, we created a semantic environment map of the Engineering Building No. 2 at the Hongo Campus of The University of Tokyo. Figure 2 visualize some aspects of the semantic map including levels, rooms, furniture and places.

#### B. Commonsense Reasoning about Objects Locations

In this section, we explain how we bootstrap a probabilistic model for reasoning about object locations. For example, if

<sup>1</sup><http://www.opencyc.org/>

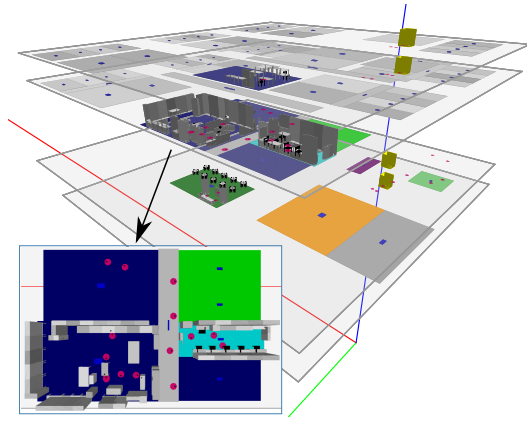


Fig. 2. Engineering Building No. 2, Hongo Campus, University of Tokyo. The map comprises several floors, elevators (yellow boxes), rooms with different types, furniture, and a subway restaurant.

the robot has no information about instances of a certain object type, it should look for these instances at locations where the probability to find this type of object is maximal.

In [2], we investigated how commonsense knowledge that was acquired from Internet users within the Open Mind Indoor Common Sense (OMICS) project [3] can be transformed from natural language to formal representations and integrated into a robot's knowledge base.

The *locations* relation in the OMICS database describes objects and their typical locations, i.e. rooms. Following our previous approach, we first transform the natural language database entries to ontological concepts. For example, the tuple  $(mug, kitchen)$  is mapped to well-defined ontological concepts  $(Cup, Kitchen)$ . Then, we calculate the conditional probability of an object given the room by counting the database entries as suggested by [3], i.e.:

$$P(x_i|\omega) = (C(x_i, \omega) + \lambda) / (C(\omega) + \lambda n)$$

where  $x_i$  denotes an object,  $\omega$  denotes a room, and  $\lambda$  denotes the parameter according to Lidstone's law. The  $\lambda$  parameter basically influences how the probability distribution account for unseen tuples. In our experiments, we set  $\lambda = 0.5$  (Jeffrey-Perk's law).

In total *locations* database table has more than 3500 entries. However, since we could not map all entries to ontological concepts and we restricted the set of rooms concepts to nine different types that fit our map, we used only 448 entries for calculating the probabilistic properties.

#### IV. SEMANTIC OBJECT SEARCH

In this section, we first present various search strategies when looking for an object, and second, we explain how the search context influences the robot's behavior.

##### A. Search Strategies

In the introduction of this paper we already mentioned the excellent ability of humans to find objects, sometimes even in previously unknown environments. Among other reasons, this is because humans make use of different strategies when looking for an object. Therefore, robots should

also be equipped with a repertoire of strategies they can employ, depending on what knowledge they have about the environment. For example, a robot that has knowledge about some instances of a sought object class should exploit this information before considering more general knowledge. However, if a robot does not have any information about object instances, it has to rely on common sense. How these different kinds of search strategies should be orchestrated is another interesting problem which beyond the scope of this paper. Hence, in this work we assume that different searches are conducted in a kind of *try-in-order* procedure.

In general, we identified three different situations in which a robot can be when looking for an object:

- 1) the object is currently not perceivable by the robot
- 2) the object is perceivable by the robot
- 3) the object is physically attached to the robot

The search for an object can basically be started in all three of these situations. If the robot already holds the sought object in its hand, it can immediately finish the search successfully. If the robot already perceives the sought object within the environment, it has to localize the object effectively for its manipulation routines and pick it up. If the robot do not perceive the object at all, the robot has to reason about the potential locations of the object, go to the locations, try to perceive the object, and if it can perceive the object try to pick it up, otherwise, the robot has to continue at the next potential location. If the robot is not able to find the object at any potential location, the robot could either switch to another search strategy, ask for help, or eventually give up the search.

In the following we explain some basic predicates that have been implemented in PROLOG in order to search for objects with different strategies using the semantic environment models described in the previous section.

**locatedAt(Obj, Pose)** denotes the pose of an object. Poses are described by the translation and rotation of an object decoded in a  $4 \times 4$  transformation matrix.

**lookForAt(Obj, Pose)** denotes the pose where an object instance might be perceivable.

**hasType(Obj, Type)** denotes the type of an object. When only providing a type all object instances with the specified type are mentally retrieved from the map.

**similarTypes(Type, SimilarTypes)** denotes the semantic relatedness between an object type and other types in the ontology. The relatedness is calculated based on the *wup* similarity measure as explained in [1]. We see basically see two possibilities for using the similarity between objects in the context of search: (1) objects that are similar are often placed together, i.e. a robot can look at places of similar objects, if it has no information about the object itself, and (2), if an object cannot be found in the environment, the robot could look for a similar object instead, e.g. if the robot cannot find any cup, it could look for a glass.

Since eventually we want to localize objects of the similar types in the environment map, only types of map

instances are considered for the calculation. Finally, the similar types are sorted with respect to the *wup* measure. **similarObj(Type, Obj)** retrieves an object instance from the map that is similar to a certain type. The predicate considers only instances in the environment map. Eventually the predicate returns all instances described in the map, for retrieving only a subset we have implemented more specialized predicates (see below).

**mostSimilarObj(Type, Obj)** retrieves only the object instances of the most similar object type from the map.

**kMostSimilarObj(K, Type, Obj)** retrieves the object instances of the k-most similar object types from the map.

**createdAt(Type, Loc)** denotes a location where instances of a given type will typically be created, e.g. a *Sandwich* will typically be created in a *Kitchen* or a *Restaurant*. Within the semantic environment map these locations are related to events using the *eventOccursAtLocation(Event, Loc)* property and if these events have a property like *outputCreated(Event, Type)* then the predicate *createdAt(Type, Loc)* holds.

**locationOf(Loc, Type, P)** denotes the probability  $P$  to encounter a given object type at a location. The probability is calculated from the probabilistic model explained in Section III-B using Bayes' rule:

$$P(\omega|x) = \frac{P(x|\omega)P(\omega)}{\sum_i P(x|\omega_i)P(\omega_i)}$$

To retrieve the location with the highest probability we simply apply the *argmax* operator  $\operatorname{argmax}_{\omega \in \Omega} P(\omega|x)$ .

Given these basic predicate definitions it is already possible to implement different kinds of search strategies when looking for an object. The most noticeable difference is that some predicates use knowledge about known instances in the robots environment whereas others only consider general knowledge about classes.

### B. Search Context

This section describes how the search results are affected by the situational context of the robot. In this paper, the context is determined by the robot's current pose.

In the previous section, we explained some basic predicates a robot can use to retrieve objects and locations from the semantic environment map. However, the decision to which location the robot will move to and look for an object depends also on its current position. That is, instead of retrieving only instance-by-instance from the map and move to the different locations successively, the robot rather retrieves the set of all instances at once and takes the respective path costs into account. More generally, these costs should include the probability of success to find an object at a location. Though, in this work we only consider a rough heuristic to calculate the path cost from the current position of the robot to a goal location. For the heuristic we distinguish two situations:

- 1) robot pose and goal pose are in the same level
- 2) robot pose and goal pose are in different levels

If the robot pose and the goal pose are in the same level, the path cost is determined simply by the Euclidean distance between the two poses. Obviously, the path cost calculation can also be approximated by using path planner on 2D occupancy grid maps.

If the robot pose and the goal pose are not in the same level, the overall path cost is determined by the sum of three cost calculations: (1) the path cost from the current position to the elevator in the same level, (2) the cost using the elevator, and (3) the path cost from the elevator to the goal pose. The costs (1) and (3) are calculated as explained above. The elevator cost is determined by the distance of levels. However, the cost model for using an elevator could be replaced by a more complex model considering for example the waiting time, intermediate stops, and the hour of the day. Figure 3 visualizes the path costs from room 73a4 in floor 7 to other rooms in the building.

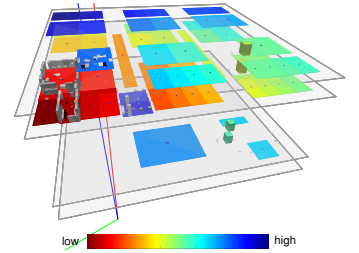


Fig. 3. Path costs visualized as heatmap calculated from room 73a4, 7th floor (dark red).

as explained above. The elevator cost is determined by the distance of levels. However, the cost model for using an elevator could be replaced by a more complex model considering for example the waiting time, intermediate stops, and the hour of the day. Figure 3 visualizes the path costs from room 73a4 in floor 7 to other rooms in the building.

## V. THE ROBOT SYSTEM

Within the experiments, we use the PR2<sup>2</sup> robot platform and a ROS<sup>3</sup>-based software infrastructure. An overview of the different software components is shown in Figure 4.

Basically, the robot's main control program is responsible for performing the fetch-and-delivery tasks. It receives a task goal from an iPad interface and performs the task autonomously. Alternatively, the task can be carried out in interaction with a user. After having received the task goal, i.e. an object to search for, one of the search strategies is used to query the semantic environment models for potential object locations. When the task-level control program recognizes that the robot has to change the floor, the relevant information is retrieved from the semantic environment model, e.g. the number of the current and the target floor, and the position of elevator control panels. These information are then send to the inter-floor navigation module to navigate the robot to the respective locations. At the goal location the robot tries to detect the object under request by using a sift-based template matching approach. Finally, the robot uses motion planning to figure out how to grasp the object.

## VI. PRELIMINARY EXPERIMENTAL RESULTS

In this section, we present the preliminary results of two example scenarios to demonstrate how a robot can use the semantic environment models in fetch-and-delivery tasks.

### A. Cups

In the first scenario, we look at situations in which a robot is supposed to find cups in the environment.

<sup>2</sup><http://www.willowgarage.com/pages/pr2/overview>

<sup>3</sup><http://www.ros.org/>

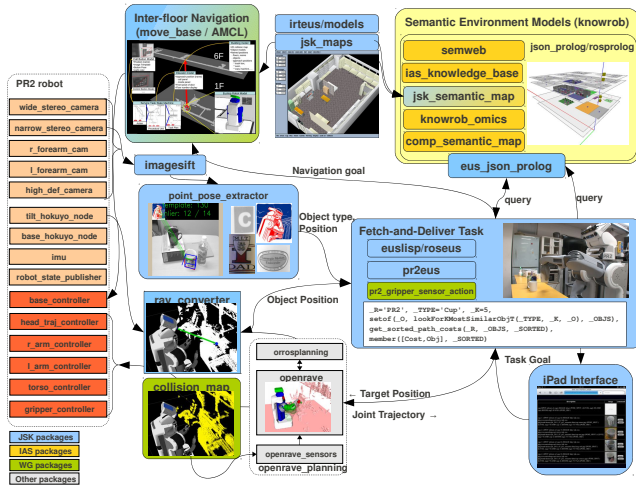


Fig. 4. Overview of the main software components realizing the execution of fetch-and-delivery tasks.

In the first situation the robot is asked to look for a particular cup, a cup that has a certain logo on it. Let's assume the robot knows some cup instances in the environment, and it also has a model of the logo. However, this logo is not associated to any of the cups, i.e., the following query fails:

```
?- hasType(C, 'Cup'), hasLogo(C, 'CMU').
```

Hence, the robot has to move to all possible cup locations and try to match the logo with one of the cups perceptually. The following PROLOG query shows how the robot can retrieve the positions of potential cup locations from the semantic map and calculate their respective path costs. The list of ordered poses is then passed to the navigation framework which uses the *lookForAt(Obj, Pose)* predicate to determine the navigation goals.

```
?- findall(Pose, (hasType(Obj, 'Cup'),
                 not(hasLogo(Obj, AnyLogo)),
                 locatedAt(Obj, Pose)), PList),
   calc_path_costs('current-pose', PList, SortedPoses).
```

Figure 5 visualizes the query result in the semantic map and show some images of the carried out robot experiment. After detecting a cup with a different logo in the first room, the robot navigates to another room where it eventually find the sought cup<sup>4</sup>. We varied the experiment by placing the individual cups at the different locations in various permutations or removing individual cups completely.

In addition to the autonomous object search, we also developed an interactive mode where users can search for objects using an iPad interface. Basically, users can ask for an object and receive a list of possible object locations. After taking a user's request, the system searches the semantic environment models for possible locations

Name	Level	Room	Pose	Cost	Image	Command
colleysty	7f	73a3	[[0.0, 0.0, 0.0]   180(deg)]	2.53224		
tea	7f	73b2	[[0.0, 0.0, 0.0]   0(deg)]	7.05173		
sumo	7f	73b2	[[0.0, 0.0, 0.0]   0(deg)]	8.89327		
cmu	7f	floor	[[0.0, 0.0, 0.0]   90(deg)]	9.00379		
mit	7f	73b2	[[0.0, 0.0, 0.0]   0(deg)]	10.09		
C	8f	83b1	[[0.0, 0.0, 0.0]   0(deg)]	75.99		

Fig. 6. iPad interface.

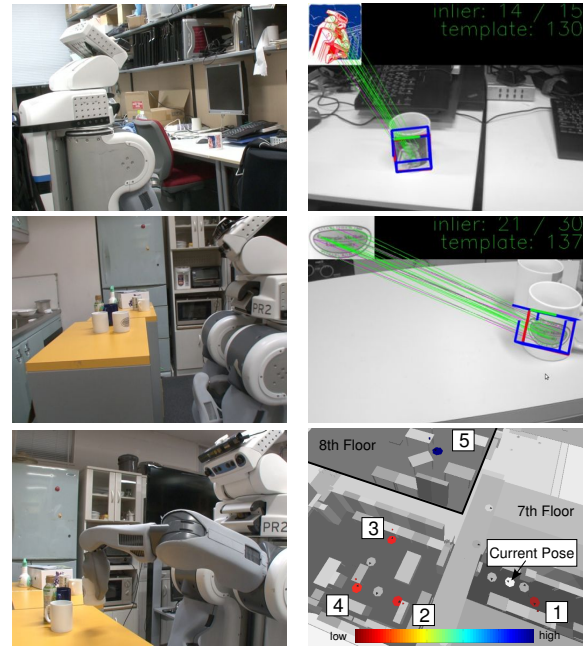


Fig. 5. Row 1: Robot at place (1). Found cup, but has PR2 logo. Row 2: Robot at place (2). Found cup with desired CMU logo. Row 3: Robot picks up cup. Visualized query results in semantic map.

using a selected search strategy, generates a list of potential locations including information about floor, room, costs, and if available an image of the object's visual appearance. The user can then decide from which location the robot should fetch the object. Figure 6 shows a query result for *Cup*.

In the previous situation we assumed that the robot had some knowledge about cup instances. However, if the robot has no such information it has to rely on more general knowledge. The following query illustrates how the robot retrieves potential locations from the probabilistic models.

```
?- findall([P, Type,
           locationOf(Type, 'Cup', P),
           PTLList),
          argmax(PTList, RoomType),
          findall(R, hasType(R, RoomType), Rs),
          member(Room, Rs),
          inRoom(Spot, Room), hasType(Spot, 'Place').
```

First, the robot finds all tuples of probability (*P*) and room type (*Type*), given that it is looking for a *Cup*. Figure 7 shows some probabilities that a type of object can be found in a room. Second, it extracts only the tuple with the highest probability, i.e. *Kitchen* (given *Cup*). Third, it retrieves all room instances of this *Type* from the semantic map, and finally, it considers all known places in these room instances. At each place the robot tries to perceive a cup at different angles.

Using the above query the robot gets the information that a *Cup* instance might be found in a *Kitchen*. So, it has to search for a cup at all places in the kitchen. However, not all places in the kitchen make sense to look for a cup, e.g. one place is in front of a TV. Using additional knowledge, for example, that the places should be in front of a table, shelf, or cupboard, can further reduce the number of places.

<sup>4</sup>Video: <http://www.jsk.t.u-tokyo.ac.jp/~kunzel/sos.html>

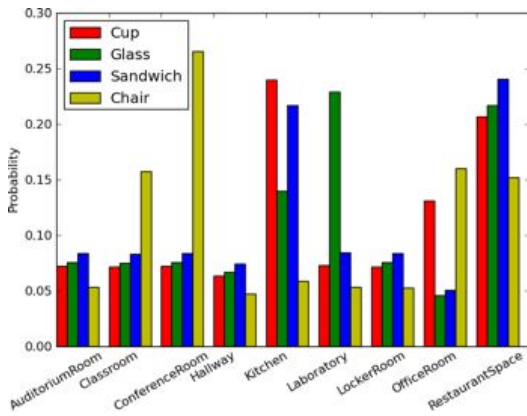


Fig. 7. Examples of probabilities to find a given object in certain room, i.e.  $P(\omega|x)$ . The initial configuration is bootstrapped from the OMICS database.

### B. Sandwiches

In this scenario the robot is asked to get a sandwich. Let's assume the robot does not have any knowledge about a *Sandwich* instance. So, it calculates which objects (it knows about) are similar to sandwiches using the following query:

```
?- mostSimilarObj('Sandwich', Obj).
```

where *Obj* is bound to *sushi1*. This instance is of type *Sushi* and it is similar to *Sandwich* because of the common super-class *Food-ReadyToEat*. With its spatial reasoning capabilities the robot finds out that *sushi1* is currently located in *frigde1* in room *73b2*.

Another possibility to find a sandwich is to search at locations where instances of sandwiches are created, e.g. kitchens or restaurants. Using the query

```
?- createdAt('Sandwich', Loc).
```

the robot infers that sandwiches might be created in room *73b2* which is of type *Kitchen* and/or *subway-shop* which is of type *FastFoodRestaurant*.



Fig. 8. PR2 using an elevator, approaching subway shop, and ordering a sandwich.

## VII. RELATED WORK

In recent years, the usage of semantic information has become more and more prominent in the context of robotics.

In [4], object-room relations are exploited for mapping and navigation tasks. Such kind of knowledge is often represented by the means of Description logics [5], [6], or probabilistic models [7], [8].

How to use semantic knowledge within object search tasks is explored by [9], [10], [11]. Our approach can be considered in a similar line of research. Probabilistic models are used in [9] to guide a simulated robot during object search tasks in structured indoor environments, namely supermarkets. Work by [10] utilizes also probabilistic methods

on a robot system to make inferences about the spatial relations between object instances in the environment. Similar to our approach, [11] bootstraps commonsense knowledge from the OMICS database to initialize the probabilistic representations. However, other forms of reasoning are not investigated in their robot experiments.

## VIII. CONCLUSION AND FUTURE WORK

In this paper, we investigated how semantic information about the robot's environment can be used in object search tasks. To this extent, we extended the representation formalisms introduced in previous work ([1]) and implement several PROLOG programs to infer the potential locations of an object. As proof-of-concept, we integrated the developed methods into a robot system that searches objects within a multi-level building in the context of fetch-and-delivery tasks. The realized system is able to navigate to inferred objects location using different kinds of semantic information.

In future work, we will include further probabilistic models about spatial relations like *in*, *on*, and *next-to* in order to restrict the search space of objects and thereby make the search tasks even more efficient. Additionally, we will conduct more experiments to evaluate the outcomes of search tasks more systematically.

## REFERENCES

- [1] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "KNOWROB-MAP – Knowledge-Linked Semantic Object Maps," in *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.
- [2] L. Kunze, M. Tenorth, and M. Beetz, "Putting People's Common Sense into Knowledge Bases of Household Robots," in *33rd Annual German Conference on Artificial Intelligence (KI 2010)*. Karlsruhe, Germany: Springer, September 21-24 2010, pp. 151–159.
- [3] R. Gupta and M. J. Kochenderfer, "Common sense data acquisition for indoor mobile robots," in *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004, pp. 605–610.
- [4] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigo, and J. González, "Multi-hierarchical semantic maps for mobile robotics," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, CA, 2005, pp. 3492–3497.
- [5] H. Zender, O. Martínez Mozos, P. Jensfelt, G. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, 2008.
- [6] K. Sjö, H. Zender, P. Jensfelt, G.-J. M. Kruijff, A. Pronobis, N. Hawes, and M. Brenner, "The explorer system," in *Cognitive Systems*, H. I. Christensen, G.-J. M. Kruijff, and J. L. Wyatt, Eds. Springer, 2010.
- [7] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Handling uncertainty in semantic-knowledge based execution monitoring," in *IROS*. IEEE, 2007, pp. 437–443.
- [8] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots - an object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359–371, 2007.
- [9] D. Joho and W. Burgard, "Searching for objects: Combining multiple cues to object locations using a maximum entropy model," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010, pp. 723–728.
- [10] A. Aydemir, K. Sjö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011.
- [11] M. Hanheide, C. Gretton, and M. Göbelbecker, "Dora, a robot exploiting probabilistic knowledge under uncertain sensing for efficient object search," in *Proceedings of Systems Demonstration of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, Freiburg, Germany, June 2011.