RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach

Luis Riazuelo, Moritz Tenorth, Daniel Di Marco, Marta Salas, Dorian Gálvez-López, Lorenz Mösenlechner, Lars Kunze, Michael Beetz, Juan D. Tardós, Luis Montano, J. M. M. Montiel

***Abstract*—The vision of the RoboEarth project is to design a knowledge-based system to provide web and cloud services that can transform a simple robot into an intelligent one. In this work we describe the RoboEarth semantic mapping system. The semantic map is composed of (1) an ontology to code the concepts and relations in maps and objects, and (2) a SLAM map providing the scene geometry and the object locations with respect to the robot. We propose to ground the terminological knowledge in the robot perceptions by means of the SLAM map of objects. RoboEarth boosts mapping by providing: (1) a subdatabase of object models relevant for the task at hand, obtained by semantic reasoning, which improves recognition by reducing computation and the false positive rate; (2) the sharing of semantic maps between robots, and (3) software as a service to externalize in the cloud the more intensive mapping computations, while meeting the mandatory hard real time constraints of the robot.**

To demonstrate the RoboEarth cloud mapping system, we investigate two action recipes that embody semantic map building in a simple mobile robot. The first recipe enables semantic map building for a novel environment while exploiting available prior information about the environment. The second recipe searches for a novel object, with the efficiency boosted thanks to the reasoning on a semantically annotated map. Our experimental results demonstrate that by using RoboEarth cloud services, a simple robot can reliably and efficiently build the semantic maps needed to perform its quotidian tasks. In addition, we show the synergetic relation of the SLAM map of objects that grounds the terminological knowledge coded in the ontology.

***Note to Practitioners*— RoboEarth is a cloud-based knowledge base for robots that transforms a simple robot into an intelligent one thanks to the web services provided. As mapping is a mandatory element on most of the robot systems, we focus on the RoboEarth semantic mapping for robot systems, showing the benefits of the combination of SLAM (Simultaneous Localization And Map building), and knowledge-based reasoning. We show the qualities of our system by means of two experiments: (1) building a map of a novel environment boosted by prior information and (2) efficient searching for a novel object thanks to the knowledge-based reasoning techniques. We can conclude that RoboEarth enables the execution of the proposed methods as web and cloud services that enable advanced perception in a simple robot.**

***Index Terms*—Semantic mapping, cloud mapping, knowledge representation, visual SLAM, object recognition.**

Luis Riazuelo, Marta Salas, Dorian Gálvez-López, Juan D. Tardós, Luis Montano, J. M. M. Montiel are with the Aragón Institute of Engineering Research (I3A), University of Zaragoza, Spain. {riazuelo, msalasg, dorian, tardos, montano, josemari}@unizar.es

Moritz Tenorth, Michael Beetz are with the Universität Bremen, Germany. {tenorth, beetz}@cs.uni-bremen.de

Daniel Di Marco is with the Universität Stuttgart, Germany. dimarco@ipvs.uni-stuttgart.de

Lorenz Mösenlechner is with the Technische Universität München, Germany. {moesenle}@cs.tum.edu

Lars Kunze is with the Intelligent Robotics Lab, University of Birmingham, United Kingdom. {l.kunze}@cs.bham.ac.uk

I. Introduction

THE ability to efficiently create semantic environment models and to use them intelligently to locate objects will become increasingly important as more and more robots enter human living and working environments. To successfully operate in such environments, robots will have to face the *open-world challenge*, i.e. they will need to be able to handle large numbers of (novel) objects located in various places on top of or inside furniture, and they need to quickly become acquainted with novel environments.

This poses several challenges for today's robots, for example: How can the visual perception system handle large numbers of object models without slowing down recognition or detecting more false positives? How can a robot efficiently explore an environment to create a map of the objects therein? Which are the most important objects to look out for? How can the robot exploit common-sense knowledge to guide its search for novel objects? How can it profit from information collected by other robots? We believe that finding solutions to these problems will be crucial to scale object search tasks from restricted and well-known laboratory environments to more open and diverse scenes.

We investigate a Web-enabled and knowledge-based approach to semantic mapping in order to build models of the environment and explore the role that cloud services can play in this mapping approach. The use of these cloud services has recently opened a new line of research in robotics called *Cloud Robotics* [1]. In [2] different architectures based on a knowledge-based solution have been presented in industrial robotized automation systems. [3] and [4] explore the use of a Cloud Computing for offloading intensive computing tasks like vision-based algorithms and grasp planning respectively. In particular, we consider a simple robot that has access to the cloud-based RoboEarth knowledge base [5], and evaluate how access to such a cloud-based knowledge base can help robots with their tasks. RoboEarth enables robots to upload and download "action recipes", models of objects they have created and maps of environments. By intelligently selecting only those pieces of information that are needed for the current task, robots can keep their local knowledge bases and object model database small and efficient, while having much larger information resources in the background.

All pieces of information in RoboEarth are semantically annotated, i.e. they are described in a formal, logical language [6] and are linked to an ontology. To achieve platform independence, these annotations include a specification of

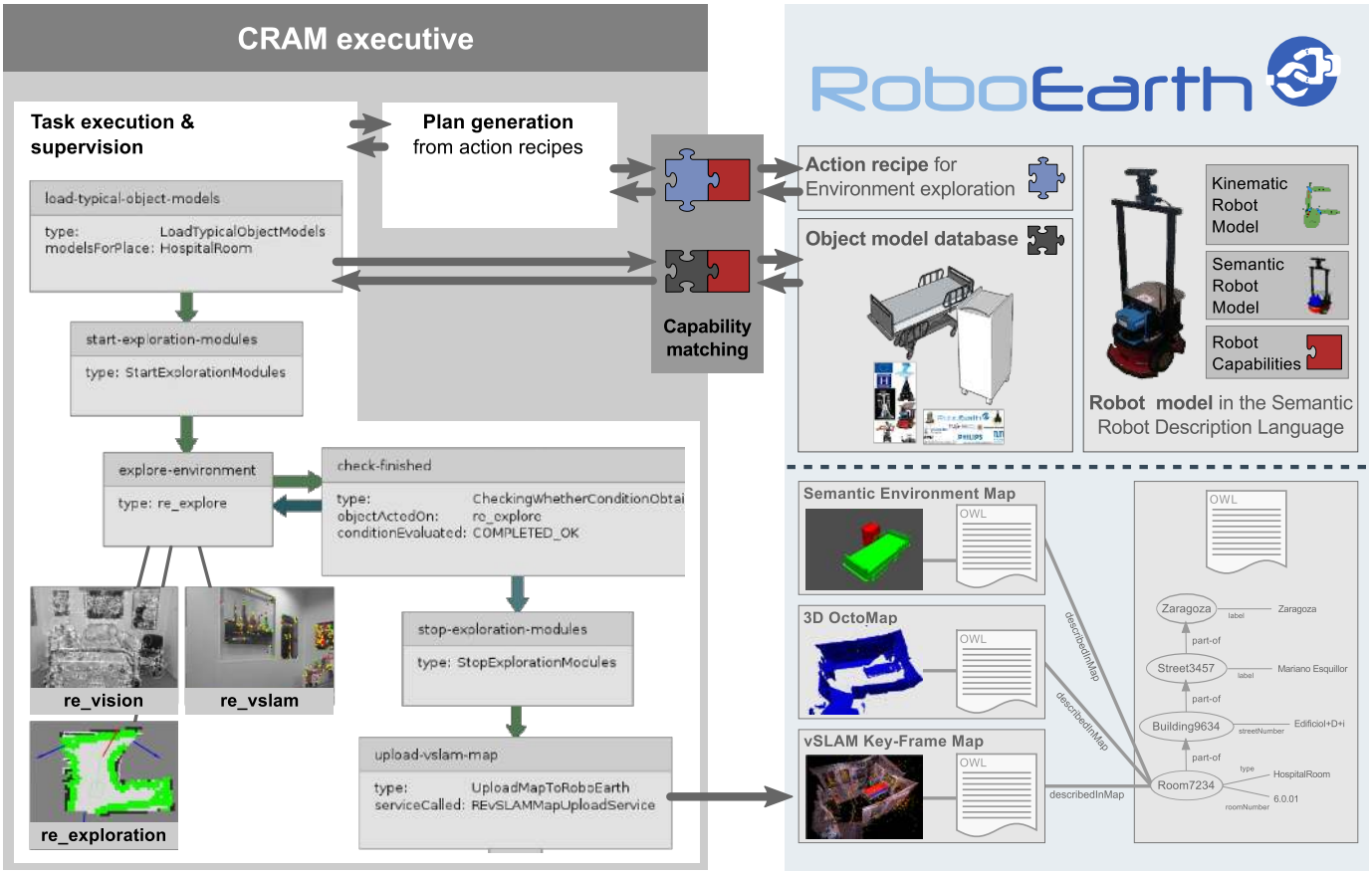

Fig. 1. Overview of the proposed system. In the beginning, the RoboEarth knowledge base (right) contains only the elements above the dotted line: An action recipe describing the exploration task, a set of object models and the robot's SRDL description. When a robot requests an action recipe, it is matched against its capability model and, if all required capabilities are available, a plan is generated. During execution of this plan (left part), the robot first downloads a set of object models that are to be expected in this environment and uses these models to build a semantic map. After execution, it uploads the generated set of maps to RoboEarth (lower part of the right block) to make them available to other robots.

which capabilities a robot needs to have in order to execute a task. When searching for suitable "action recipes", a robot can match this specification against a formal model of its own components and capabilities described in the Semantic Robot Description Language (SRDL, [7]). If necessary components or capabilities are missing on the robot, the recipe cannot be executed and is not considered for download. If all required capabilities are available, the robot model is used to generate a plan that is tailored to the hardware of the respective robot. The semantic annotations further enable robots to perform logical inference, for instance to decide which are the most likely objects in a room (and only download their models to their local database), or where novel objects are likely to be found (and guide the search accordingly).

In order to apply abstract knowledge to operation in the real world, it needs to be *grounded* [8] in the robot's perception system and its knowledge about the environment. In this article, we propose to link the knowledge base with a visual SLAM system that provides accurate and continuous asynchronous perception. The system is integrated with an object recognition module that identifies objects based on a local database of object models. The main contributions of this work are (1) a semantic mapping method resulting from the synergistic integration of a visual SLAM map of objects with the RoboEarth ontology; (2) knowledge-based methods for using prior information, exemplified in the selection of object models for exploration and in the guidance of a robot when searching for a novel object; and (3) methods for embodying the semantic map building and exploitation in a simple robot using RoboEarth cloud services.

The remainder of the paper is organized as follows: We start with an overview of related work on searching for objects, explain the structure of our system as well as the two main tasks it performs: The creation of an initial semantic map building and knowledge-guided object search. We then present the system's components in more detail, describe the experiments we have performed, and finish with our conclusions.

II. Related Work

Several proposals have been made for building maps of objects. Objects from a database are recognized and located in [9] where polyhedral CAD object models are recognized in single RGBD images. Similarly, using point clouds, in [10] geometrical primitives are segmented assuming they correspond to scene objects. Combining visual SLAM with object recognition to produce maps of objects has recently gained more attention for pure visual RGB sensors in [11], [12], and for RGBD in [13]. Several approaches have been

made to endow maps with reasoning capabilities. A Bayesian network classifier is proposed in [14] to encode the relations between objects in a scene and the objects typically present in a type of room. An ontology-based approach is proposed in [15], [16] to represent knowledge about the elements in a map. The knowledge-based maps by Zender et al. [17] provide grounding by combining place recognition from 2D laser maps and object recognition. An exploration method similar to ours has been proposed in [18]. Our contribution is to combine a knowledge-base with a visual SLAM map of objects to ground the robot perceptions to implement the RoboEarth Web and cloud mapping services. For the estimation of this semantic map, we propose the use of action recipes that describe how to explore the free space while searching for objects in a local database using an object recognition algorithm.

Structured object search and reasoning about likely object locations have been an active research topic over the past years. Much of the work has explored vision-based methods to search for objects in a top-down manner based on saliency and visual attention mechanisms [19], [20], [21]. Having a (partial) semantic map allows a robot to apply background knowledge for directing the search. One possibility is to learn co-occurrence statistics of object types and object–room relations, for example from online image databases [22] or from search engine results [23]. Joho et al. [24] use co-occurrence information and other heuristics for efficiently searching for objects in structured environments, in particular supermarkets. Schuster et al. exploit similarity scores computed based on an ontology of object types for directing the search towards locations where semantically similar objects are known to be [25]. Kunze et al. propose a utility-based approach for object search that particularly focuses on the decision of which location to search first [26]. This work was extended in [27] to use geometric models of directional qualitative spatial relations with respect to landmark objects and to use 2D cones for approximating the sensor field of view. Wong et al. include manipulation actions into the object search, which allows the robot to reason about which objects have to be removed before being able to see the target object [28]. The approach by Aydemir et al. [29] is similar to ours in that they also use landmark objects to guide the search for smaller objects inside or on top of the landmarks. While they focus on the probabilistic formulation of the search procedure as a Markov Decision Process, we explore a knowledge-based strategy that exploits formal knowledge about object types, their (likely) spatial relations, and their shape and appearance.

III. System overview

Figure 1 shows the typical workflow of a robot using the system. We assume that the RoboEarth knowledge base (right block) contains the required task descriptions (called "action recipes") and object models. In this paper, we focus on two action recipes for (a) semantic mapping of an unknown environment and (b) active search for an object based on a partial semantic map. The locations of objects already detected in the room thereby serve as landmark objects.

Each piece of information is annotated with a description of the capabilities required for making use of it (depicted as colored puzzle pieces), that is matched against a formal model of the robot's capabilities described in the Semantic Robot Description Language. Based on the background knowledge about which objects are likely to be encountered in which kinds of rooms, RoboEarth infers a set of object models that can be recognized during the exploration.

After download, a robot plan is generated from the action recipe (Section IV) and the task is executed accordingly. The robot explores the environment using a frontier-based algorithm (*re_explore* component), recognizes objects using the *re_vision* module and inserts them into a map build by the *re_vslam* module. After the exploration has finished, the robot exports the map in the formal RoboEarth language and uploads it to the RoboEarth knowledge base.

IV. Action Recipes for Active Perception Tasks

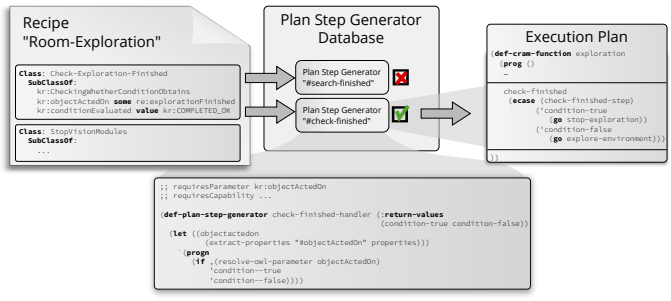

Fig. 2. Generation of the execution plan. The recipe (left) is an OWL document composed of parametrized subactions, described in terms of OWL classes. To generate the plan, the system looks in the database for code generating functions that are applicable on the specific instance and robots (bottom), and inserts the resulting function into the final execution plan (right).

Action recipes abstractly specify which actions need to be performed to accomplish a task in a (largely) robot- and environment-independent manner. RoboEarth aims at the exchange of recipes between heterogeneous robots in different environments, which therefore need to be reduced to a description of the task itself, eliminating all hardware- and environment-specific parts. While the resulting descriptions can easily be transferred to another robot, they are too abstract to be directly executable. The robot thus needs to interpret the instructions, fill in missing information, and select and parameterize suitable "skills" that provide the implementation for the respective action steps. The capability matching procedure described in Section V verifies that all skills needed for executing a recipe are available on a robot.

Action recipes are formulated in the RoboEarth language [6] that is based on the W3C-standardized Web Ontology Language OWL [30]. Actions in a recipe are described as classes whose properties described action parameters such as the *objectActedOn*. They can inherit properties from more generic classes in the knowledge base, which we often use for inheriting information about required capabilities. This way, the recipes can be kept short and concise, since 'common-sense' knowledge does not have to be communicated. Examples of action recipes for exploring an environment and searching for objects can be found in Figures 1 and 3, respectively. These

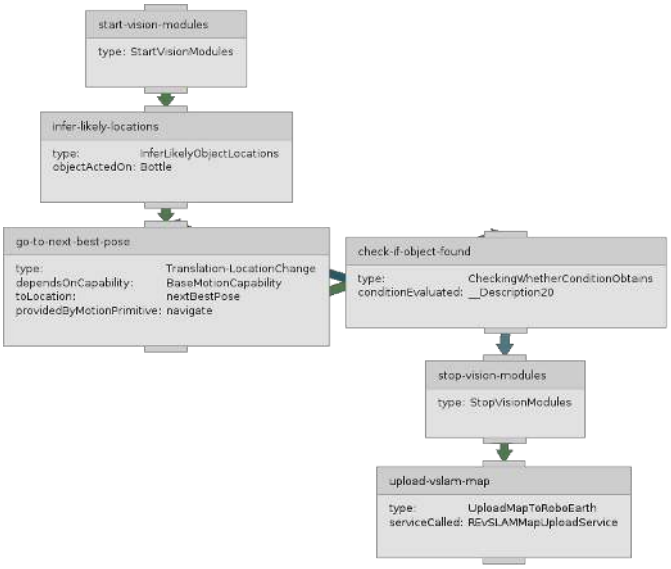

Fig. 3. ObjectSearch Action recipe task execution

recipes can easily be created using a graphical editor without knowledge of the OWL language[1].

As mentioned earlier, action recipes are not executable by themselves, but aggregate "skills" (that implement single action steps) into more complex task structures. Our system uses the CRAM executive [31] for controlling the robot, so the "skills" correspond to fragments of the robot plans. These fragments are not static, but are generated by Lisp macros that are parameterized with the OWL description of an action step (Figure 2). This allows to consider the action context as well as the robot model to generate tailored plans. For example, when generating the code for computing object visibility (Sec. VI), the pose of the camera relative to the robot base is read from the robot's SRDL description.

The code generation macros are also stored in the RoboEarth database and can therefore be shared among robots. For each action described in the recipe, the system searches for suitable macros considering the robot's capabilities. In case multiple results are found, the one with the minimal semantic distance (estimated via the Rada distance [32]) to the action at hand is selected. If the result is still ambiguous, a human operator is asked. The code generation macros then extract the required action parameters from the robot model and the semantic environment map.

As part of this work, we have created two action recipes to enable a simple robot to perform semantic mapping in the cloud using RoboEarth. The first action recipe (Fig. 1 left), sketched in Algorithm 1, enables a robot to build a semantic map for a novel environment, exploiting prior information about the room type. The second one (Fig. 3) illustrates how information from the semantic map can be exploited when searching for an specific object. Algorithm 2 sketches the steps of this recipe. The described recipes build upon a set of perception and navigation capabilities that are detailed in Section V.

[1]See http://knowrob.org/doc/action_recipe_editor.

A. SemanticMapping Action Recipe

The execution of the *SemanticMapping* action recipe results in an exploratory behavior of the robot. Before starting the exploration, the knowledge base infers a set of landmark objects that are typically found in the type of room to be explored. These models are loaded into a local subdatabase on the robot that allows real-time object recognition for map building. It further increases the recognition precision and recall because only objects that are likely to be in the room are searched for. After completing the room exploration, it produces a semantic map that is stored in the RoboEarth knowledge base.

The recipe commands the robot to explore the room while avoiding obstacles. Simultaneous to room exploration, the visual SLAM builds a map providing locations for selected geometrical features and landmark objects recognized in the scene. Once the exploration is finished, the object instances are linked with the RoboEarth ontology in order to upgrade the map of objects into a semantic one. The semantic map along with the occupancy grid and features map are uploaded as a RoboEarth environment.

B. ObjectSearch Action Recipe

The *ObjectSearch* recipe assumes that a (partial) semantic map valid for the room is already stored on a RoboEarth environment. Based on the locations of landmark objects in this map, the knowledge base infers potential locations from where the object might be detected. From the occupancy map, the free space for robot navigation is computed, and according to the robot's SRDL model, the sensors' ranges and locations within the robot are inferred. The features map stored on the RoboEarth environment allows the visual SLAM to provide a continuous robot localization when the map is reused. Using all this information, a list of robot locations is computed from where the object is likely to be detected.

Upon execution of the generated CRAM plan, the robot sequentially navigates towards the computed locations from where it searches for the object until it is found. The detected object is added to the initial semantic map, which is then finally uploaded back to the RoboEarth database.

V. Robot Capabilities for Active Perception

Since RoboEarth aims at knowledge exchange among heterogeneous robots, we cannot assume that every robot pos-

Algorithm 1 SemanticMapping(**in:** environType, environId)

```
subDataBase = load-typical-object-models(environType)
slamVisualMap = void
start-exploration-modules()
start-vision-modules(slamVisualMap, subDataBase)
repeat
  explore-environment(freeFrontiers)
until check-finished(freeFrontiers)
return-to-initial-pose()
environment = upgrade-to-semantic(slamVisualMap)
upload-environment-map(environment, environId)
```

Algorithm 2 ObjectSearch(**in:** environId, object)

```
environment = download-environment(environId)
semanticMap,slamVisualMap = extract(environment)
start-navigation()
start-vision-modules(slamVisualMap)
nextPoses = infer-likely-locations(semanticMap, object)
repeat
  go-to-next-best-pose(nextPoses)
  slamVisualMap = search(slamVisualMap,object)
until check-if-object-found(object) or last-location-reached
stop-vision-modules()
environment = upgrade-to-semantic(slamVisualMap)
upload-environment-map(environment,environId)
```

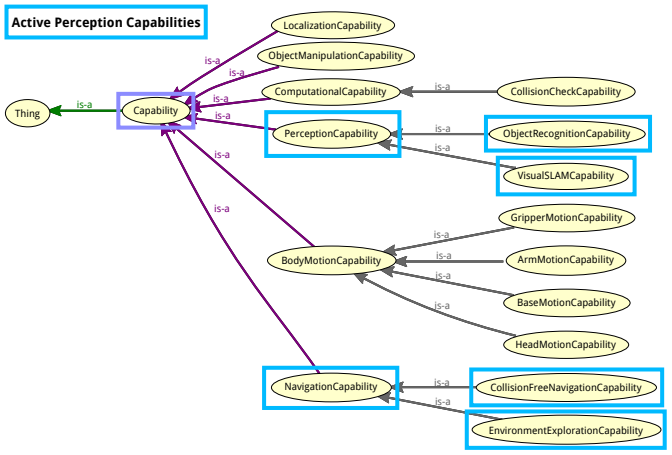

Fig. 4. A sub-branch of the SRDL ontology stating some of the robot capabilities. All the mandatory compatibilities for active perception are highlighted in blue.

sesses all required capabilities for executing a recipe. Therefore, both those capabilities that are available on a robot and those that are needed for a task are modeled and can automatically be matched using the Semantic Robot Description Language (SRDL). This procedure is described in detail in [6]. Capabilities are usually *provided* by software components (e.g. ROS nodes) on the robot, are *interfaced* from CRAM plan fragments, and are *described* in SRDL to allow reasoning about which tasks are feasible. Capabilities are often not binary, but may be available to a certain degree. It is however hard to measure this, since the criteria will be different for many different abilities. We therefore do not store a quantitative degree to which an ability is available, but distinguish different cases as specialized subclasses as can be seen e.g. for the different kinds of navigational abilities. In general, SRDL does support numerical attributes such as the range of a laser scanner or the resolution of a camera. Dependencies of actions on capabilities are usually not described in the recipe itself, but inherited from more generic action classes in the RoboEarth ontology (e.g. that all kinds of reaching motions need an arm component). Capabilities are also described as OWL classes and are declared in another branch in the RoboEarth ontology. The capabilities needed for the two recipes described

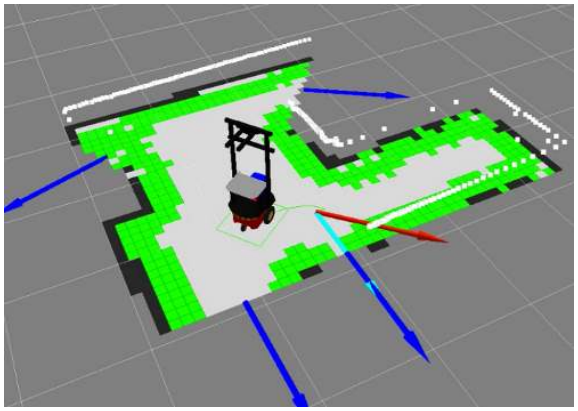

Fig. 5. Visualization of the frontier-based exploration algorithm. Black contours represent known obstacles and the green grid cells encode the inflation for safe navigation. The dark blue arrows represent unexplored frontiers. The next frontier to be explored is coded as a light blue arrow.

in this work that focus on active perception are highlighted in Figure 4 and will be presented in the remainder of this section.

a) CollisionFreeNavigationCapability: represents the ability to safely navigate to a goal. The initial global navigation plan to achieve the goal is locally modified by a reactive navigation module which is responsible for computing the motions finally commanded to the robot. The planning technique is based on a A*-type algorithm [33]. For reactive navigation, we have applied ORM [34] adapted for differential drive robots due to its performance in dense, complex and cluttered environments. A Rao-Blackwellized particle filter [35] is used to estimate the robot location and the 2D navigation map from 2D laser rangefinder readings.

b) EnvironmentExplorationCapability: declares the ability to actively build a 2D navigation map of an unknown environment. Based on the 2D laser readings and the odometry, the component guides the robot in building a 2D map of its environment. The main issue is to compute at run-time the next robot locations from where to perceive unexplored regions. The next point of view is computed according to the frontier-based approach [36], where the robot moves while avoiding obstacles and integrating the 2D laser readings into the map (*NavigationComponent*). The exploration ends when the map contains no more accessible frontiers. Figure 5 visualizes the method.

c) ObjectRecognitionCapability: declares the ability to recognize objects in single images and to provide an initial estimate of their 3D location with respect to the camera. The corresponding component implements an object recognition algorithm [12] in which each object is modeled as a collection of *faces*. Each face comprises an image that represents a point of view of the object, a set of SURF features [37] and their associated 3D coordinates in the local object frame, obtained by multi-view geometry [38]. These models are initially stored in the RoboEarth database. When a subset of them is required to fulfill a task, they are downloaded, creating a local subdatabase used by the recognition algorithm.

d) VisualSLAMCapability: declares the capability to estimate a visual SLAM map composed of point features and recognized objects, and a 3D occupancy grid map. This capability

is implemented by a distributed framework, C^{2}TAM [39], based on PTAM [40]. A lightweight process handles the camera tracking on the onboard robot computer, while the expensive map optimization is externalized as a service in the cloud (Amazon EC2 service [41]) using the RoboEarth Cloud Engine [42]. Thanks to this division, the hard real-time constraints mandatory in a robotic embedded system are met by the visual SLAM, despite the typical network delays in the link with the cloud server. The SLAM map not only includes visual point features but also objects that are recognized in the images by the *ObjectRecognitionComponent*. The recognition models come from the local subdatabase provided by RoboEarth. Once the map is computed, the result is incorporated into the RoboEarth environment data structure, providing the data for the following classes:

- *SemanticEnvironmentMap*s are described in OWL and consist of objects detected in the environment, described as instances of the respective object classes in the ontology. This allows the application of logical inference methods to the spatial configuration of objects. The object instances may further contain information about their extensions, 6D poses and possibly CAD models describing their geometry and appearance.
- *OctoMap*: a 3D occupancy grid map, coded as proposed in [43]. It is computed from RGB-D sensor readings in the visual SLAM keyframes. This map can be reused to generate 2D maps for navigation.
- *ReVslamMap*: the raw storage of visual maps for visual localization. They are built from the sole input of an RGB-D camera. Provides a continuous localization of the robot while the map is building. Furthermore, thanks to the capability of Roboearth system for sharing environments, this map can be downloaded and reused by other robots in order to localize, while navigate, in the same environment.

VI. Reasoning about Object Locations

In order to successfully find an object in the environment, a robot must answer the questions "Where is the object likely to be?" and "Where do I need to go in order to see it?".

1) Inferring likely object positions: We employ knowledge that has been extracted from the OMICS common-sense database [44] and converted into the representation used in the robot's knowledge base [45] to compute likely object positions. The OMICS database holds tuples of objects and their locations in the form $(object, location)$. The number of times a relation is contained in OMICS can be used to approximate the likelihood that an object O can be found at a location LOC:

$$P(O|LOC) = count(O, LOC)/count(LOC) \qquad (1)$$

where $count$ is the number of database entries. The value of $P(LOC|O)$ is calculated from the above model using Bayes' rule. To retrieve the location with the highest probability we simply apply the *argmax* operator

$$\underset{LOC \in Locations}{\operatorname{argmax}} \quad P(LOC|O) \qquad (2)$$

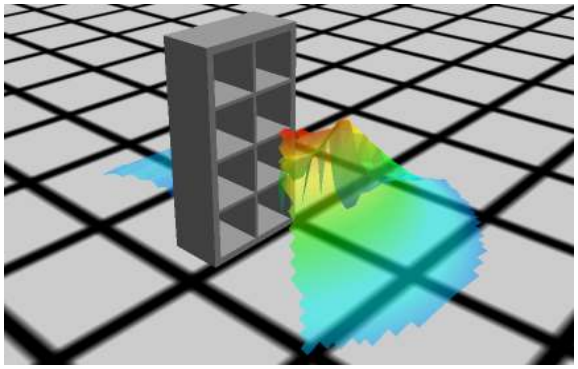

Fig. 6. Visibility costmap computed from the semantic environment map, the semantic robot model and geometric object models downloaded from RoboEarth. The colors indicate the amount of the object that is visible from a given camera of the robot considering its pose.

The resulting models allow queries for the locations of objects given by corresponding landmark objects. These object classes can be grounded in the robot's semantic environment map to determine their positions.

2) Computing robot poses using visibility reasoning: Based on the semantic map (that contains known object instances in the environment) and CAD models of these objects previously downloaded from RoboEarth, the system computes a visibility costmap describing from which robot poses the object is likely to be visible [46]. Especially for objects that are inside a cabinet or shelf, occlusions by the surrounding objects need to be taken into account when planning a pose for the robot. To compute the costmap, the system renders the scene from the viewpoint of the inferred object location and computes the amount of the object that is visible from each grid cell in the costmap (Fig. 6).

VII. Experiments

This section is devoted to showing how diverse robots benefit from the cloud-based RoboEarth semantic mapping system. The experiments include those carried out with a real Pioneer P3-DX robot and simulations. We demonstrate how a simple robot can reliably and efficiently build and exploit the semantic maps needed to perform quotidian tasks using the Roboearth cloud services [2]. The experiments are based on the two action recipes described in Section IV.

Assuming that RoboEarth contains a huge database of object models, one key advantage is the ability to serve a reduced subdatabase that only contains the relevant models for the current tasks. This reduces the local computation overhead and improves recognition precision and recall. In the case of the semantic map building for a novel environment, given the type of the environment, in this case a hospital room, RoboEarth is able to elaborate and serve to the robot a subdatabase containing only object models expected to be relevant and salient in this environment. In this case, the selected object categories are a bed and cabinet. For each object category all the relevant individual object models are included in the

[2] A video of the experiments can be found at http://robots.unizar.es/data/videos/roboearth/roboearthSemanticMapping.mp4

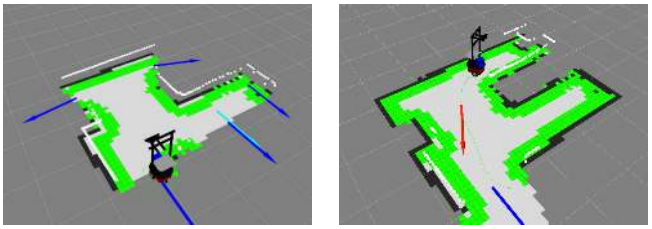

Fig. 7. Initial (left) and final (right) steps of the exploration algorithm. The dark blue arrows represent the currently unexplored frontiers.

Fig. 8. Object recognition events: bed (left) and cabinet (right).

subdatabase. In contrast in the active search recipe, the served subdabase would contain only the recognition model of the object searched for.

Given the SRDL model of a robot, RoboEarth can produce and serve a customized CRAM plan for this robot. In the simulation we consider two different robots operating in two different environments. It is shown how, from a single recipe, four different execution CRAM plans are generated, one per robot-environment combination.

To illustrate the benefits that a simple robot can gain from using RoboEarth, we focus on the increase in efficiency derived from the exploitation of the knowledge-based reasoning available in the semantic maps in the case of a search for a novel object. In contrast to an exhaustive search, RoboEarth exploits a map the environment acquired previously and performs a knowledge-based search strategy of small objects by landmark objects.

A. Real-world experiments

The following scenario has been investigated. A robot in a hospital room has to find a bottle to be served to a patient. Initially, the robot does not know the location of the bottle. The naïve and expensive solution would have been to exhaustively search the whole room. In contrast, to improve efficiency, we use both the semantic mapping and object search recipes (Alg.1, Alg.2) to embody a knowledge-based search strategy in the robot.

We used a Pioneer P3-DX in which the navigation is based on a Sick 2D laser scanner and odometry sensors. It has been implemented by means of the ROS stacks *GMapping*, and *move_base* that has been extended to include the ORM obstacle avoidance. The robot also incorporates a Kinect RGB-D that provides the raw data for visual mapping. The visual SLAM is implemented by means of the C^{2}TAM algorithm that externalizes heavy computations using a Platform as a Service, in our case the RoboEarth Cloud Engine. It is worth noting that during the experiments, C^{2}TAM has been able to fulfill all the mandatory real-time constraints of our robot-embedded computer, despite the delays and low bandwidth typical of any computer network. Regarding the inference methods, the ROS RoboEarth stack [47] and the KnowRob knowledge base [48] are used. The capability matching and the CRAM plan generation have been executed locally on our robot computer, but these can also be externalized to the cloud.

1) Semantic Mapping: Before performing the task, the knowledge base infers that the bed and the cabinet are likely landmark objects, and the corresponding object models are inserted in the model subdatabase that is served to the robot. A customized CRAM plan is generated for the robot based on the recipe. The robot executes the CRAM plan and starts to explore the unknown environment until it obtains a complete map of the room. Figure 7 shows the beginning and end of the exploration. At the beginning the map is incomplete, with several open frontiers that have yet to be explored. At the end the complete map is estimated.

While the robot is exploring the environment, the perception component builds the visual SLAM map and inserts the detected objects according to the models in the subdatabase. Figure 8 shows two examples of object recognition events. Once the exploration is finished the robot uploads the created semantic map to RoboEarth (as we can see in Fig. 1). This comprises the detected objects (Fig. 9), the map of visual features,and a 3D occupancy grid map, coded as an OctoMap (Fig. 10).

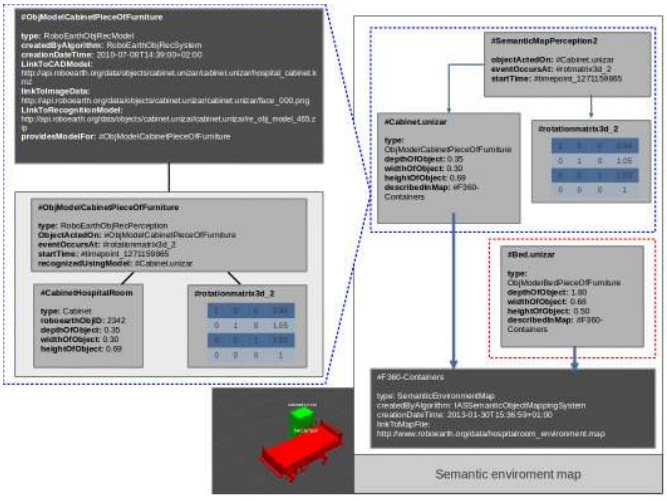

Fig. 9. Detailed storage format for a semantic map composed by two objects, a bed and a cabinet. Each object instance contains information about the type of object, dimensions, recognition model used, time detection and its location into the map.

2) Object Search: The second recipe execution presents the guided object search. This is based on the semantic map of the environment built and uploaded in the previous exploration (Sect. VII-A1). The object location inference determines the cabinet as the landmark object to guide the search for the bottle. Taking the scene layout and occupancy map into account, several reachable robot locations from where the bottle is likely be detected are computed (see Figure 11).

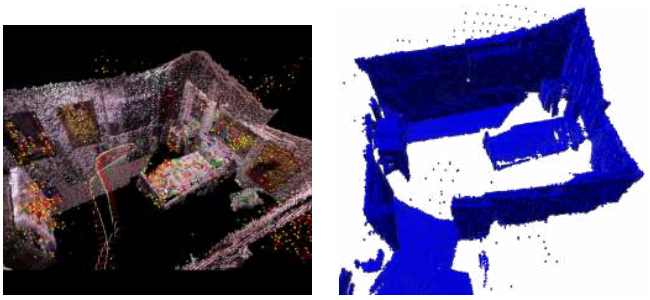

Fig. 10. Map of visual features (left), and 3D occupancy grid OctoMap (right).

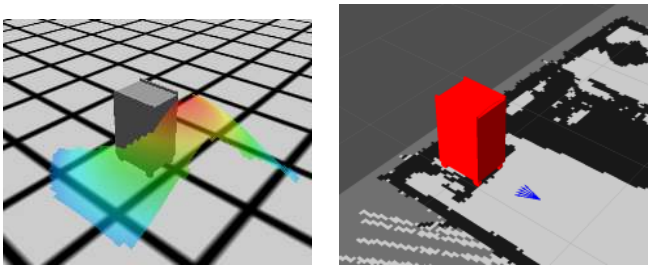

Fig. 11. Visibility costmap (left). Occupancy map and, in blue, the selected search robot locations (right)

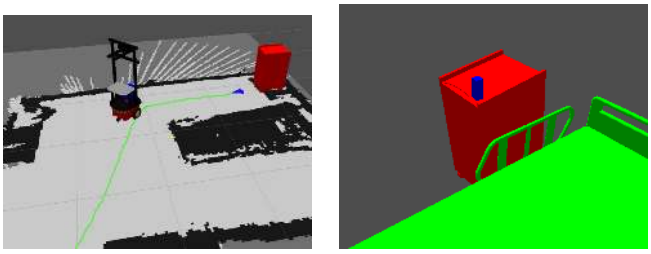

Fig. 12. Computed robot locations for detecting the object, in blue, and planned trajectory, in green (Left). Final semantic map including the bottle detected on top of the cabinet (Right).

Considering the SRDL description of the Pioneer P3-DX, the stored semantic map and the action recipe, RoboEarth provides: 1) a custom 2D map for navigation estimated from the OctoMap, 2) a customized CRAM plan, and 3) the set of recognition models for the bottle.

The provided CRAM plan iteratively drives the robot to a list of selected positions until the object is eventually found. Once it is located, its position is added to the map and the map is uploaded to RoboEarth. Figure 12 shows the robot trajectory and the semantic map including the objects known a priori (bed and cabinet) and the new one (the bottle).

B. Simulation Experiments

The goal of the simulation experiments is to demonstrate the interoperability of the system. For these experiments, we have used the open source robotics simulator Gazebo [49]. The same object search action recipe has been executed on two different robots in two different environments. The selected robots have been the holonomic service robot Amigo [50], and the previously described non-holonomic Pioneer P3-DX. Per each selected robot, the SRDL model describes its capabilities

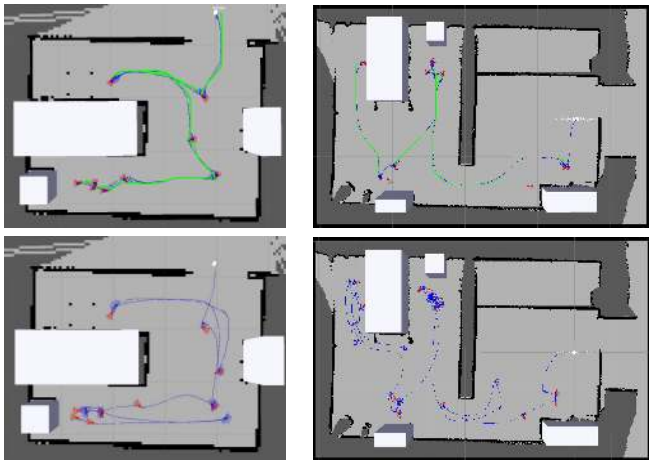

Fig. 13. Travelled path (blue) in simulated object search. Top row displays Amigo, and bottom row displays Pioneer. Left column for the room, right column for the suite.

and kinematics, enabling RoboEarth to produce a specific CRAM plan for a definite robot in a particular environment.

The searched object has been considered to be probably found on top of beds, cabinets or shelves. The first environment emulates the hospital room, assuming a semantic map where a bed, and a cabinet have been located. The second environment mimics a suite, composed of two rooms communicated through an open door. It is assumed to have a semantic map where a bed, a cabinet and two shelves have been detected. Figure 13 shows the paths resulting from the four different CRAM plans, one per each robot in each room. The paths mirror the difference in locomotion, the Amigo robot is able to maneuver more efficiently in the tight spaces than the non-holonomic Pioneer P3-DX. The two robots also have their camera in different locations; consequently the reasoning for the path generation has been influenced by the differences in visibility.

C. Performance Improvements

The purpose of this experiments is to highlight the benefits of using the proposed system. We focus the quantitative results on three aspects: (1) the externalization in the cloud of the most intensive computations; (2) the use of a subdatabase of objects which improves recognition; (3) the efficiency of the knowledge-based search strategy based on landmark objects implemented by the object search action recipe.

1) Computational Efficiency: The use of the cloud for externalizing the expensive computation processes provides an improvement in the response time, as we can see on figure 14. This figure presents two graphs that show the response time per frame of the tracking process of the visual SLAM system with respect to the size of the map during the execution of exploration action recipe. On the top graph, we can see the performance of the system when the expensive computation process of the visual SLAM system used is running on the cloud. The tracking response time remains constant (around 10 ms) independently of the map size. The bottom graph shows the tracking time when the complete C^{2}TAM system

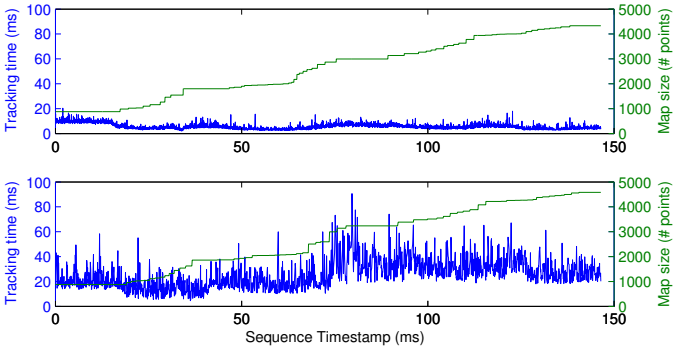

Fig. 14. Response time of the tracking process. Top graph C^{2}TAM running mapping in the cloud, botton graph all C^{2}TAM processes running onboard the robot.

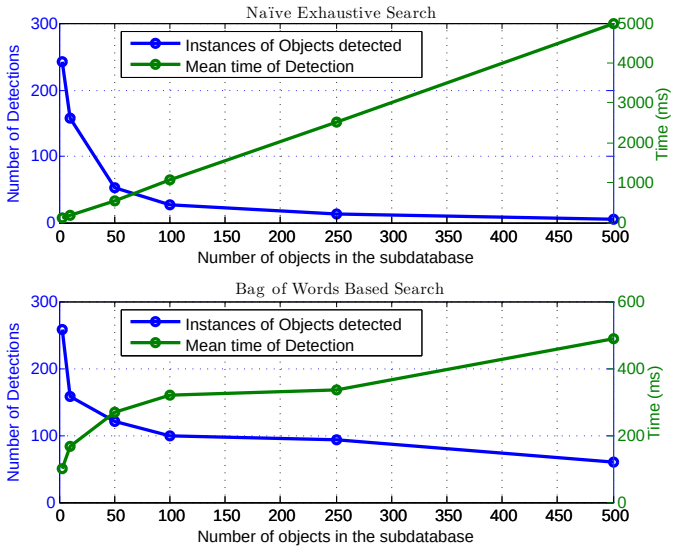

Fig. 15. Number of detections and time performance of the object detector as function of the subdatabse size. Top, naïve recognition. Bottom bag of words preselection

is running onboard the robot. We can see how the tracking response time increase when the size of the map grows and even it overtakes the video frame rate threshold (33 ms). We can conclude that the externalization of the expensive map optimization process of C^{2}TAM as a service in the cloud provides an improvement in the response time of the real-time critical processes because they can benefit of all the onboard resources once the mapping process is outsourced to the cloud.

2) Recognition using a Subdatabse: Two search strategies has been tested for a range of subdatabase sizes. The first strategy is a naïve detection that checks all the models in the subdatabase. The second is an advanced one that only checks the 10 most promising object models according to an appearance score obtained when their local features are converted into bags of words (BoW) [51]. In this experiment we focus on the semantic mapping of a hospital room. The subdatabase contains the RoboEarth provided relevant models in all the experiments. Additional object models, up to 500, are added to the subdatabase to analyze the effect of a big database containing objects not appearing in the actual scene. Figure 15 shows a quantitative performance analysis. The top graph

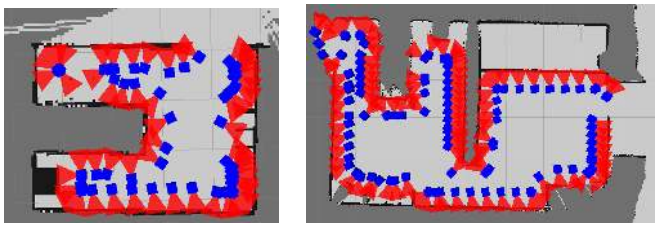

Fig. 16. Art Gallery exhaustive search. Blue squares code the search locations, and red sectors represent the camera field of view. Room: 40 locations (Left). Suite: 100 locations (Right)

shows the naïve detector, bottom graph shows the advance BoW recognition. As expected mean time of detection after the BoWs preselection scales better with the subdatabase size. Both of the methods produces more detections with reduced subdatabase, i.e. low false negative rate after processing the whole experiment sequence. Additionally in our experiments we did not detect any false positive, what is a good indicator of a remarkable recognition precision. In any case, in both algorithms, we can see how the increase number of objects in the subdatabase degrades the performance. We can conclude that a subdatabase which only contains the most relevant models for a specific task provides a better performance on the object detection in terms of number of detections and speed. RoboEarth is able to provide this subdabase of only relevant objects.

3) Knowledge-based Search Strategy: Finally, in terms of the proposed knowledge-based search strategy, we show how the priors provided by the semantic map are able to reduce the number of potential search locations, and hence significantly reduce the search time. We compare the guided exploration trajectories with those of an exhaustive search. The comparison is made in terms of the number of locations from where the object search is performed in the worst case. The selected scenarios are the room and the suite described in the previous section. For the search locations in the exhaustive case, we have selected the Art Gallery algorithm [52] because, for a given sensor visibility range, it provides the minimum number of positions which can cover a particular environment. Figure 16 shows the locations which achieve full coverage of the considered environments. The number of locations depends on the size of the environment – the bigger the environment, the higher the number. In our case, 40 locations were computed for the room and 100 for the suite. The benefit is evident if we compare this with the knowledge-based search (Fig. 13 bottom row), where the room needs only 9 locations and 15 were needed by the suite, leading to a corresponding reduction in the search time.

VIII. Conclusions

A robot operating in an environment for the first time can benefit from information previously stored by other robots operating in the same environment, thanks to the RoboEarth semantic mapping system. The proposed semantic mapping system combines a visual SLAM map of objects with an ontology representing the knowledge. Thanks to this combina-

tion, knowledge-based reasoning about map entities becomes possible.

We have demonstrated that the building and exploitation of this mapping system can be implemented as web and cloud services. The robot has to provide its SRDL description, and hence RoboEarth provides all the information needed to execute the task. The result of the execution is also stored in the database for reuse by the same or other robots. We have provided a pioneering experimental validation of a web-enabled cloud semantic mapping system exemplified in the case of map building and guided search for a novel object. We conclude that our system can (a) enable robots to perform novel tasks, (b) generate semantically meaningful environment maps, and (c) reason about these maps in conjunction with formally described background knowledge. Indeed, the strategy cannot address all cases in an open world at once (i.e. recognize all objects at all times), but this is in general not feasible at the moment. With limited on-board resources, the options are either to manually select a number of objects that can be recognized (as it is commonly done today), or to give the robot the ability to autonomously select a range of object models that are to be expected in the environment. This is obviously limited by the quality of the predictions, but still more flexible than a rigid selection of objects.

Evidence has also been provided about the possibility of externalizing in the cloud those processes which are demanding in terms of memory or CPU while at the same time meeting the hard real time robot constraints. We can hence conclude that the operation of simple robots with typical computing and networking facilities can be boosted by RoboEarth.

ACKNOWLEDGMENTS

The research was partly funded by the EU FP7 projects *RoboEarth* (grant number 248942) and *RoboHow* (grant number 288533), the Spanish Government grants DPI2012-32168 and DPI2012-32100, and the Aragón regional grant DGA-FSE (grupo T04).

REFERENCES

[1] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE T-ASE Special Issue on Cloud Robotics and Automation*, vol. 12, no. 2, 2015.

[2] M. Stenmark, J. Malec, K. Nilsson, and A. Robertsson, "On distributed knowledge bases for robotized small-batch assembly," *IEEE T-ASE Special Issue on Cloud Robotics and Automation*, vol. 12, no. 2, 2015.

[3] J. Salmerón-García, F. Díaz-del Río, I.-B. P., and C. D., "A trade-off analysis of a cloud-based robot navigation assistant using stereo image processing," *IEEE T-ASE Special Issue on Cloud Robotics and Automation*, vol. 12, no. 2, 2015.

[4] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg, "Cloud-based grasp planning for toleranced parts using parallelized monte carlo sampling," *IEEE T-ASE Special Issue on Cloud Robotics and Automation*, vol. 12, no. 2, 2015.

[5] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Haussermann, J. Montiel, A. Perzylo, B. Schiele, O. Zweigle, and R. van de Molengraft, "Roboearth - a world wide web for robots," *IEEE Robotics and Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[6] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "Representation and Exchange of Knowledge about Actions, Objects, and Environments in the RoboEarth Framework," *IEEE Transactions on Automation Science and Engineering (T-ASE)*, vol. 10, no. 3, pp. 643–651, 2013.

[7] L. Kunze, T. Roehm, and M. Beetz, "Towards semantic robot description languages," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 5589–5595.

[8] S. Harnad, "The symbol grounding problem," *Physica D*, vol. 42, pp. 335–346, 1990.

[9] M. Günther, T. Wiemann, S. Albrecht, and J. Hertzberg, "Building semantic object maps from sparse and noisy 3d data." in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2228–2233.

[10] J. Mason and B. Marthi, "An object-based semantic world model for long-term change detection and semantic querying." in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 3851–3858.

[11] R. Castle, G. Klein, and D. Murray, "Combining monoSLAM with object recognition for scene augmentation using a wearable camera," *Image and Vision Computing*, vol. 28, no. 11, pp. 1548–1556, 2010.

[12] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic slam using a monocular camera," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, sept. 2011, pp. 1277 –1284.

[13] R. Salas-Moreno, F. R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *IEEE Proc. Computer Vision and Pattern Recognition (CVPR)*, 2013.

[14] S. Vasudevan and R. Siegwart, "Bayesian space conceptualization and place classification for semantic maps in mobile robotics," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 522–537, 2008.

[15] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008.

[16] D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz, "Semantic object maps for robotic housework - representation, acquisition and use," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

[17] H. Zender, O. Martínez 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, Jun. 2008.

[18] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe, "Curious george: An attentive semantic robot," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, Jun. 2008.

[19] A. Oliva, A. Torralba, M. S. Castelhano, and J. M. Henderson, "Top-down control of visual attention in object detection," in *Int. Conf. on Image Processing (ICIP).*, vol. 1, 2003, pp. I–253.

[20] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*, 2005, pp. 84–89.

[21] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3d environment using a mobile robot," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 535–547, 2010.

[22] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 2168–2173.

[23] K. Zhou, M. Zillich, H. Zender, and M. Vincze, "Web mining driven object locality knowledge acquisition for efficient robot behavior," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 3962–3969.

[24] D. Joho, M. Senk, and W. Burgard, "Learning search heuristics for finding objects in structured environments," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 319–328, May 2011.

[25] M. Schuster, D. Jain, M. Tenorth, and M. Beetz, "Learning organizational principles in human environments," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.

[26] L. Kunze, M. Beetz, M. Saito, H. Azuma, K. Okada, and M. Inaba, "Searching objects in large-scale indoor environments: A decision-thereotic approach," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN, USA, 2012.

[27] L. Kunze, K. K. Doreswamy, and N. Hawes, "Indirect object search based on qualitative spatial relations," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, May 31 - June 7 2014, accepted for publication.

[28] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation-based active search for occluded objects," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6-10 2013, pp. 2814–2819.

[29] A. Aydemir, A. Pronobis, M. Gbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics." *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.

[30] W3C, *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. World Wide Web Consortium, 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027.

[31] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[32] E. Blanchard, M. Harzallah, H. Briand, and P. Kuntz, "A typology of ontology-based semantic measures." in *EMOI-INTEROP workshop at the 17th Conf. on Advanced Information Systems Engineering*, Porto, Portugal, 2005.

[33] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[34] J. Minguez, "The obstacle-restriction method (orm) for robot obstacle avoidance in difficult environments," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[35] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, February 2007.

[36] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. of the IEEE Int. Symposium on Computational Intelligence, Robotics and Automation*, 1997, pp. 146–151.

[37] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[38] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," in *ACM SIGGRAPH 2006 Papers*. ACM, 2006, pp. 835–846.

[39] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C2TAM: A Cloud framework for Cooperative Tracking And Mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, April 2014.

[40] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM Int. Symposium on*, 2007, pp. 225–234.

[41] Amazon Inc., "Amazon elastic compute cloud," 2012. [Online]. Available: http://aws.amazon.com/ec2/

[42] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The RoboEarth cloud engine," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), Karlsruhe, Germany*, 2013, pp. 438–444.

[43] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, 2010. [Online]. Available: http://octomap.sf.net/

[44] R. Gupta and M. J. Kochenderfer, "Common sense data acquisition for indoor mobile robots," in *Nineteenth National Conf. on Artificial Intelligence (AAAI-04)*, 2004, pp. 605–610.

[45] L. Kunze, M. Tenorth, and M. Beetz, "Putting People's Common Sense into Knowledge Bases of Household Robots," in *33rd Annual German Conf. on Artificial Intelligence (KI 2010)*. Karlsruhe, Germany: Springer, 2010, pp. 151–159.

[46] L. Mösenlechner and M. Beetz, "Parameterizing Actions to have the Appropriate Effects," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, Sept. 25-30 2011, pp. 4141–4147.

[47] "ROS packages for interfacing with RoboEarth." http://wiki.ros.org/roboearth_stack, accessed: 2014-01-25.

[48] M. Tenorth and M. Beetz, "KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System," *Int. Journal of Robotics Research*, vol. 32, no. 5, pp. 566 – 590, 2013.

[49] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, sept. 2004, p. 2149?2154.

[50] J. Lunenburg, S. van den Dries, J. Elfring, R. Janssen, J. Sandee, and M. van de Molengraft, "Tech united eindhoven team description 2012," in *RoboCup Team Description Papers 2012*, 2012.

[51] D. Gálvez-López and J. D. Tardós, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[52] T. Shermer, "Recent results in art galleries," *Proc. IEEE*, vol. 80, no. 9, pp. 1384 – 1399, 1992.