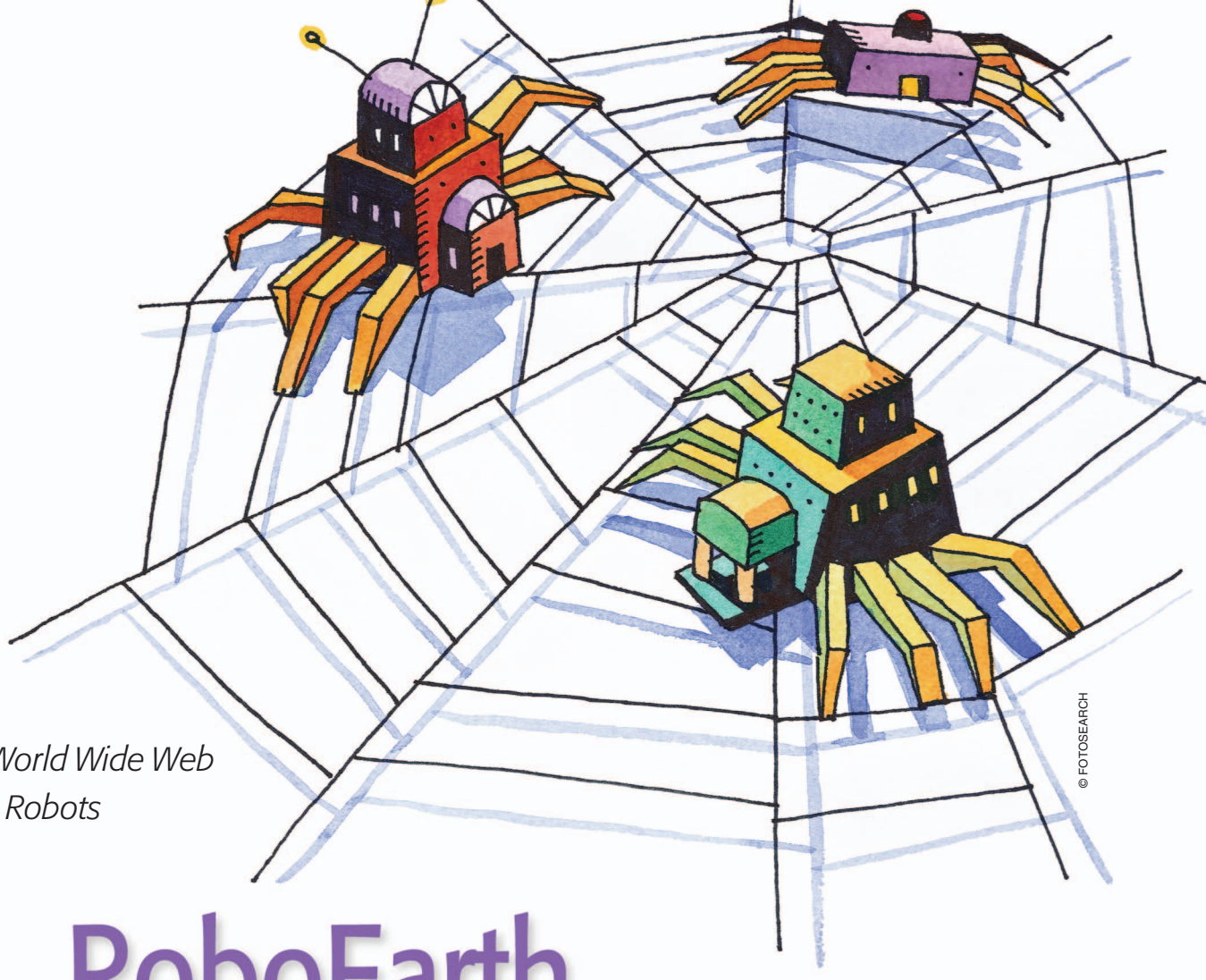


© FOTOSEARCH

A World Wide Web for Robots

RoboEarth

By Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, Rob Janssen, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Moritz Tenorth, Oliver Zweigle, and René van de Molengraft

Digital Object Identifier 10.1109/MRA.2011.941632
Date of publication: 14 June 2011

Humans can use the Internet to share knowledge and to help each other accomplish complex tasks. Until now, robots have not taken advantage of this opportunity. Sharing knowledge between robots requires methods to effectively encode, exchange, and reuse data. In this article, we present the design and first implementation of a system for sharing knowledge between robots.

In the manufacturing and logistics industries, robotic systems have brought significant sociological and economic benefits through improved human safety, increased equipment utilization, reduced maintenance costs, and increased production. In a world that is undergoing significant environmental and social change, there will be an increasing demand for these robots to leave the safety of their controlled environments and operate in the real world. Robots will be required to operate in homes and hospitals to service the health of a rapidly aging population, and they will be required to mine and farm in increasingly remote locations. In these environments, robots will need to reliably perform tasks beyond their explicitly preprogrammed behaviors and quickly adapt to the unstructured and variable nature of tasks.

Although there has been much progress in task performance with well-defined sets of objects in structured environments, scaling current algorithms to real-world problems has proven difficult. Today's robots can only

perform highly specialized tasks, and their operation is constrained to a narrow set of environments and objects. The majority of the world's 8 million service robots are toys or drive in preprogrammed patterns to clean floors or mow lawns, while most of the 1 million industrial robots repetitively perform preprogrammed behaviors to weld cars, spray paint parts, and pack cartons [1].

To date, the vast majority of academic and industrial efforts have tackled these challenges by focusing on increasing the performance and functionality of isolated robot systems. However, in a trend mirroring the developments of the personal computing (PC) industry [2], recent years have seen first successful examples of augmenting the computational power of individual robot systems with the shared memory of multiple robots. In an industrial context, Kiva Systems successfully uses systematic knowledge sharing among 1,000 individual robots to create a shared world model that allows autonomous navigation and rapid deployment in semistructured environments with high reliability despite economic constraints [3], [4]. Other examples for shared world models include research on multiagent systems, such as RoboCup [5], where sharing sensor information has been shown to increase the success rate of tracking dynamic objects [6], collective mapping of autonomous vehicles [7], [8], or distributed sensing using heterogeneous robots [9].

However, in most cases, robots rely on data collected once in a first, separate step. Such pooled data have allowed the development of efficient algorithms for robots, which can then be used offline without access to the original data. Today's most advanced personal assistant robots rely on such algorithms for object recognition and pose estimation [10], [11]. Similarly, large training data sets for images and object models have been crucial for algorithmic advances in object recognition [12]–[17].

In some cases, pooled data have also been used to offer additional robot services. Examples include localization services [18], scene recognition [19], and, more recently, robotic manipulation [20], [21]. This is especially true in mobile robotics, where computer vision algorithms have proven successful in detecting loops while traversing trajectories [22], labeling and classification of visited places [23], [24], and augmentation of maps with semantic concepts [25]. Other systems query large object databases to obtain information [26].

Another class of approaches uses pooled data to extract previously unknown correlations. Amazon's recommendation system [27], Apple iTunes' "Genius" feature [28], and Google Goggles [29] are well-known software examples. The Photo Tourism software [30] is a good example of how a large collection of image data can be used to obtain new information, such as three-dimensional (3-D) representations of public scenes. In the context of object recognition, priors extracted from pooled data have been shown to significantly speed up learning [31], [32]. Some work has applied this learning step on data gathered from the Internet [33] to obtain object models. Others have used correlations of diverse data to obtain richer models for image recognition [34]. Yet another method for exploiting data correlations in this context are bag of words [35], which summarize image features for creating appearance models for fast recognition.

Comparatively little research has addressed the sharing and reuse of knowledge. Some researchers have proposed sharing pooled data using an Internet search engine [36] or a cloud computing framework [37]. Others have suggested embedding knowledge directly into objects [38]. Attempts such as the planning domain definition language target the standardization of plan languages and planning domain specifications [39]. Another approach aims at creating abstract representations for high-level knowledge that can be shared across multiple platforms [40], [41].

Today, the vast majority of data for robots is dependent on specific hardware configurations, which limits reuse to identical robots. In addition, existing databases typically contain only one type of data in isolation. This continues to severely constrain the reuse of data.

RoboEarth collects, stores, and shares data independent of specific robot hardware. In addition, data in RoboEarth is linked [42]. For example, the computer-aided design (CAD) model of an object may be linked to semantic descriptors (e.g., the object's English name), the object's properties (e.g., object weight), its relation to other objects (e.g., belongs to the class "bottle"), or instructions for manipulating the object (e.g., grasp points).

RoboEarth

Example Scenario

Imagine the following scenario. A service robot (robot A) in a hospital room was programmed to serve a drink to a patient. This task includes locating a bottle that contains the drink, navigating to the bottle's position on a cupboard, grasping and picking up the bottle, locating the patient in the bed, navigating to the patient, and giving the bottle to the patient. Imagine that during task execution, robot A monitors and logs its progress and continuously updates and extends its rudimentary, preprogrammed world model with additional information. It updates and adds the position of detected objects, evaluates the correspondence of its map with actual perception, and logs successful and unsuccessful attempts during its task performance. If the robot is not able to fulfill a task, it asks a person for help and stores any newly learned knowledge. At the end of task performance, the robot shares the acquired knowledge by uploading it to a distributed database.

Sometime later, the same task is to be performed by a second robot (robot B) that has no prior knowledge on how to execute the task. This second robot queries the database for relevant information and downloads the knowledge previously collected by robot A. Although differences between the two robots (e.g., due to wear and tear or different robot hardware) and their environments (e.g., due to changed

object locations or a different hospital room) mean that the downloaded information may not be sufficient to allow robot B to reperform a previously successful task, this information can nevertheless provide a useful starting point—a prior. Recognized objects, such as the bed, can now provide rich occupancy information even for areas not directly observed by robot B. Detailed object models (e.g., of a bottle) can increase the speed and reliability of robot B's interactions. Task descriptions of previously successful actions (e.g., driving around the bed) can provide guidance on how robot B may be able to successfully perform its task.

This and other prior information (e.g., the previous location of the bottle, and the bed is a likely place to find the patient) can guide the second robot's search and execution strategy. In addition, as the two robots continue to perform their tasks and pool their data, the quality of prior information will improve and begin to reveal the underlying patterns and correlations about the robots and their environment.

Although many of the requirements for this scenario are challenging research questions, we believe that the availability of such prior information is a necessary condition for the robots to operate in more complex, unstructured environments. The benefits of storing, sharing, and reusing information are not restricted to tomorrow's mobile service robots. Today, thousands of robotic systems solve the same essential problems over and over again.

Ultimately, the nuanced and complicated nature of human environments cannot be summarized within a limited set of specifications but will require robots to systematically share data and build on each other's experience. We believe that a World Wide Web for robots will allow the robots to achieve successful performance in increasingly complex tasks and environments.

Proof of Concept

This article describes RoboEarth, a worldwide, open-source platform that allows any robot with a network connection to generate, share, and reuse data. The work described here is a first step by the RoboEarth Consortium [43] toward delivering a proof of concept that

- RoboEarth greatly speeds up robot learning and adaptation in complex tasks
- robots using RoboEarth can execute tasks that were not explicitly planned for at design time.

This proof of concept includes a distributed database that stores reusable data for objects, maps, and tasks. As part of its proof of concept, the RoboEarth Consortium will also implement a series of demonstrators centered around the hospital scenario outlined in the "RoboEarth: Example Scenario" section.

Design Principles

The architecture and implementation of RoboEarth is guided by a number of design principles, centered around the idea of allowing robots to reuse and expand each other's knowledge. To facilitate reuse of data, RoboEarth supports and leverages existing standards. The database is made available via standard Internet protocols and is based on an open-source cloud architecture to allow others to set up their own instance of RoboEarth, resulting in a truly distributed network. The code generated by the RoboEarth Consortium will be released under an open-source license, and will provide well-documented, standardized interfaces. Finally, RoboEarth stores semantic information encoded in the World Wide Web Consortium (W3C)-standardized Web Ontology Language (OWL [44]) using typed links and uniform resource identifiers (URIs) based on the principles of linked data [42].

Architecture

RoboEarth is implemented based on a three-layered architecture (Figure 1). The core of this architecture is a server layer that holds the RoboEarth database [Figure 1(a), the "Architecture: Database" section]. It stores a global world model, including reusable information on objects (e.g., images, point clouds, and models), environments (e.g.,

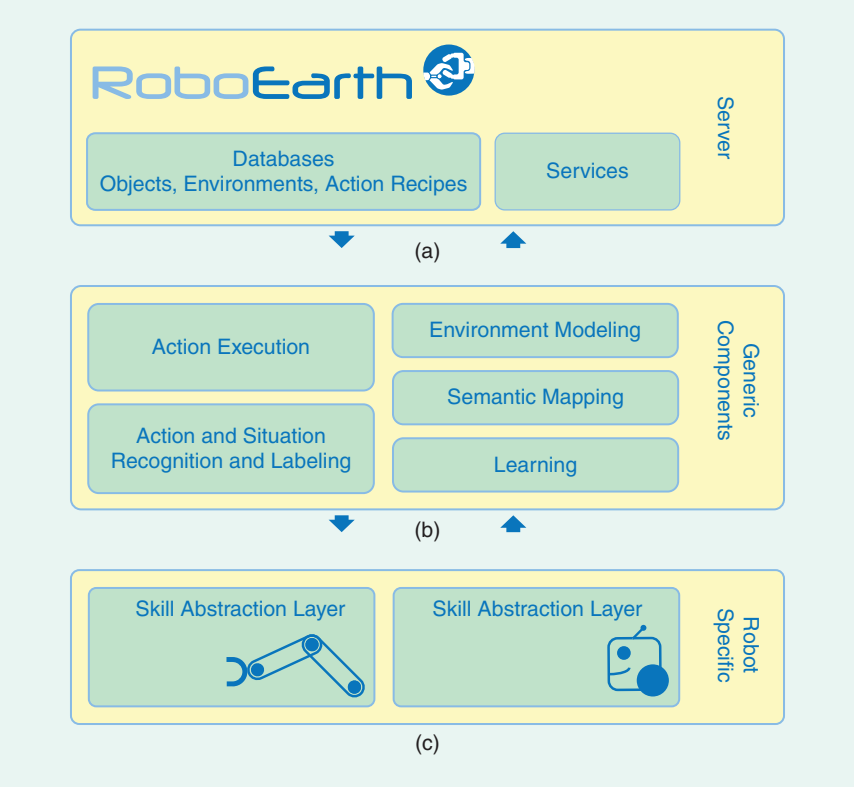


Figure 1. RoboEarth's three-layered architecture.

maps and object locations), and actions (e.g., action recipes and skills) linked to semantic information (e.g., properties and classes), and provides basic reasoning Web services. The database and database services are accessible via common Web interfaces (see the "Architecture: Interfaces" section for details).

As part of its proof of concept, the RoboEarth Consortium [43] is also implementing a generic, hardware-independent middle layer [Figure 1(b)] that provides various functionalities and communicates with robot-specific skills [Figure 1(c)]. The second layer implements generic components (see the "Architecture: Generic Components" section for details). These components are part of a robot's local control software. Their main purpose is to allow a robot to interpret RoboEarth's action recipes. Additional components enhance and extend the robot's sensing, reasoning, modeling, and learning capabilities and contribute to a full proof of concept that closes the loop from robot to the World Wide Web database to robot.

The third layer implements skills and provides a generic interface to a robot's specific, hardware-dependent functionalities via a skill abstraction layer (see the "Architecture: Skill Abstraction Layer and Robot-Specific Components" section for more details).

Database

RoboEarth stores CAD models, point clouds, and image data for objects. Maps are saved as compressed archives, containing map images and additional context information such as coordinate systems. Robot task descriptions are stored as human-readable action recipes using a high-level language to allow sharing and reuse across different hardware platforms. Such action recipes are composed of semantic representations of skills that describe the specific functionalities needed to execute them. For a particular robot to be able to use an action recipe, the contained skills need to have a hardware-specific implementation on the robot. To reduce redundancy, action recipes are arranged in a hierarchy, so that a task described by one recipe can be part of another more complex recipe. In addition, database services provide basic learning and reasoning capabilities, such as helping robots to map the high-level descriptions of action recipes to their skills or determine what data can be safely reused on what type of robot.

The RoboEarth database has three main components (Figure 2). First, a distributed database contains all data organized in hierarchical tables [Figure 2(a)]. Complex semantic relations between data are stored in a separate graph database [Figure 2(b)]. Incoming syntactic queries are directly passed to the distributed database for

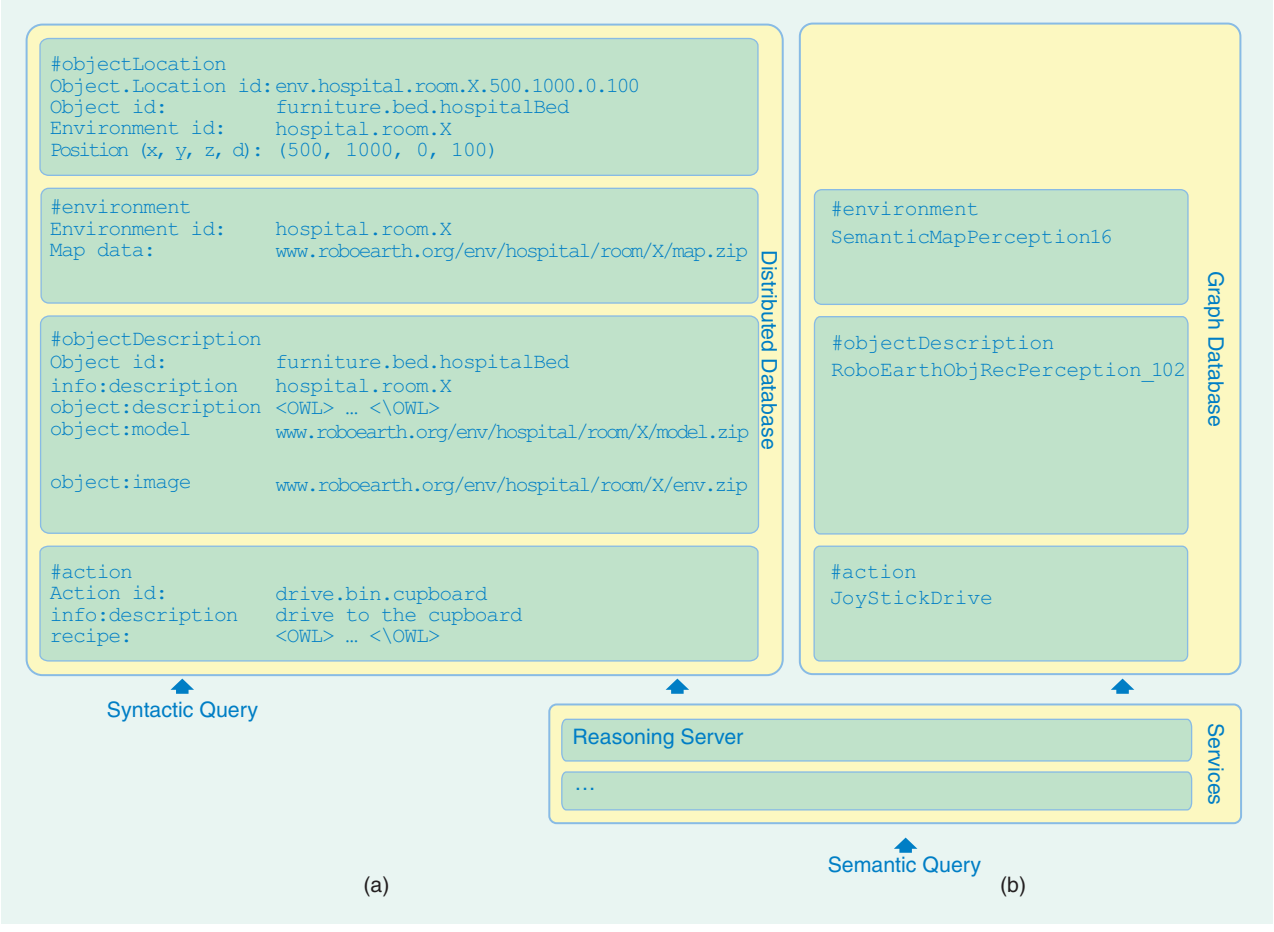


Figure 2. The three main components of the RoboEarth database (see details in the "Architecture: Database" section).

processing. Semantic queries are first processed by a reasoning server. Data are stored in a distributed database based on Apache Hadoop [45], which organizes data in hierarchical tables and allows efficient, scalable, and reliable handling of large amounts of data. Examples of the kind of data stored in this database include the recognition model of the bottle or the object model of the hospital bed mentioned in the "RoboEarth: Example Scenario" section.

Second, a centralized graph database holds semantic information encoded in the W3C-standardized OWL [44]. It stores the following data and their relations.

- *Objects:* The database stores information on object types, dimensions, states, and other properties as well as locations of specific objects a robot has detected and object models that can be used for recognition (Figure 3). Figure 3(a) describes a recognition model for a certain kind of object (defined by the property `providesModelFor`), giving additional information about the kind of model and the algorithm used. The actual model is linked as a binary file in the format preferred by the respective algorithm (defined by the property `linkToRecognitionModel`). Figure 3(b) describes the recognition of a specific object. An instance of a `RoboEarthObjRecPerception` is created, which describes that the object `Bottle2342` (linked through the property `objectActedOn`) was detected at a certain position (linked through the property `eventOccursAt`) at a given point in time using that recognition model (defined by the property `recognizedUsingModel`).
- *Environments:* The database stores maps for self-localization as well as poses of objects such as pieces of furniture (Figure 4). The semantic map combines a binary

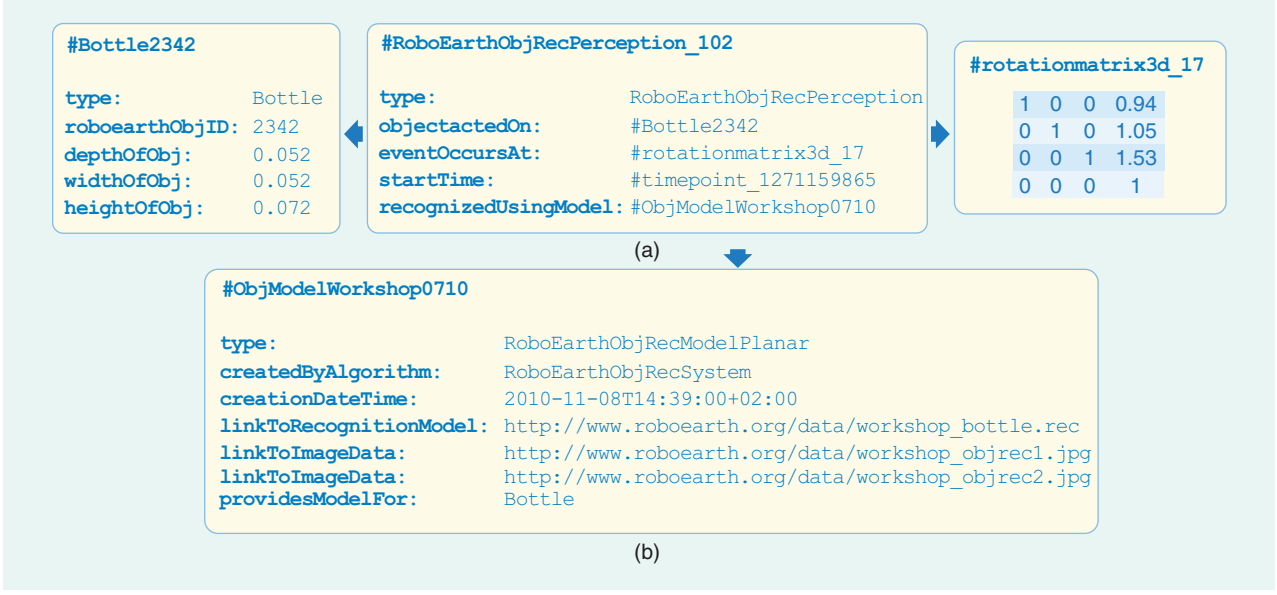


Figure 3. The object description, recognition model, and one particular perception instance of the bottle used in the second demonstrator (see details in the "Demonstrators: Second Demonstrator" section).

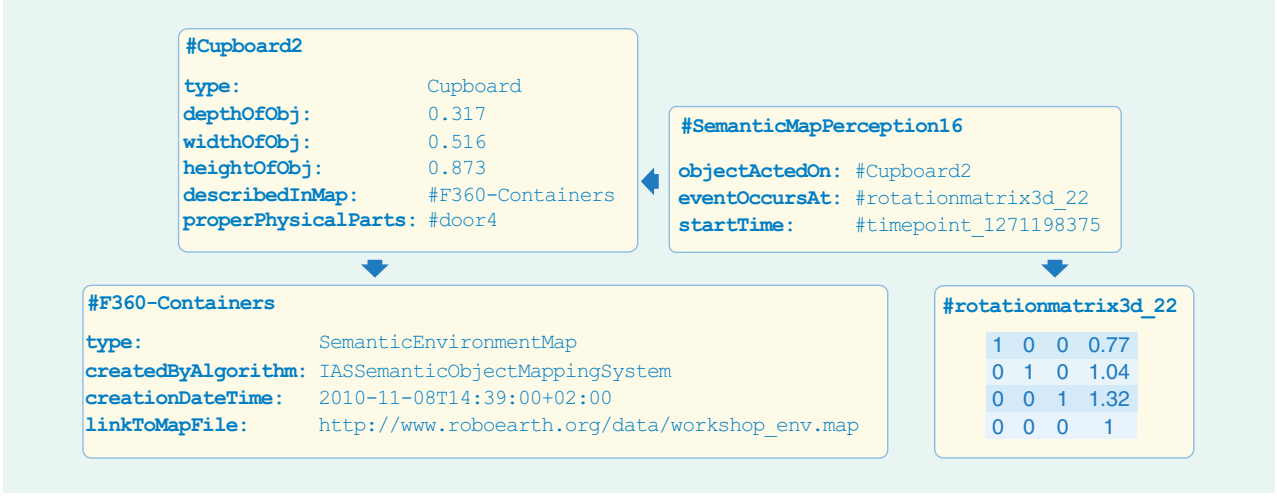


Figure 4. The environment map used in the second demonstrator (see details in the "Demonstrators: Second Demonstrator" section).

map that is linked using the `linkToMapFile` property with an object that was recognized in the respective environment. The representation of the object is identical to the one in Figure 3. This example shows that both binary (e.g., occupancy grids) and semantic maps consisting of a set of objects can be exchanged and even combined. The given perception instance not only defines the pose of the object but also gives a time stamp when the object was seen last. This can serve as a base for calculating the position uncertainty, which increases over time.

- *Action Recipes*: The stored information includes the list of subaction recipes, skills, and their ordering constraints required for executing an action recipe as well as action parameters, such as objects, locations, and grasp types (Figure 5). Action classes are visualized as blocks, properties of these classes are listed inside of the block, and ordering constraints are depicted by arrows between the blocks. The recipe is modeled as a sequence of actions, which can be action recipes by themselves, e.g., the `GraspBottle` recipe. Each recipe is a parameterized

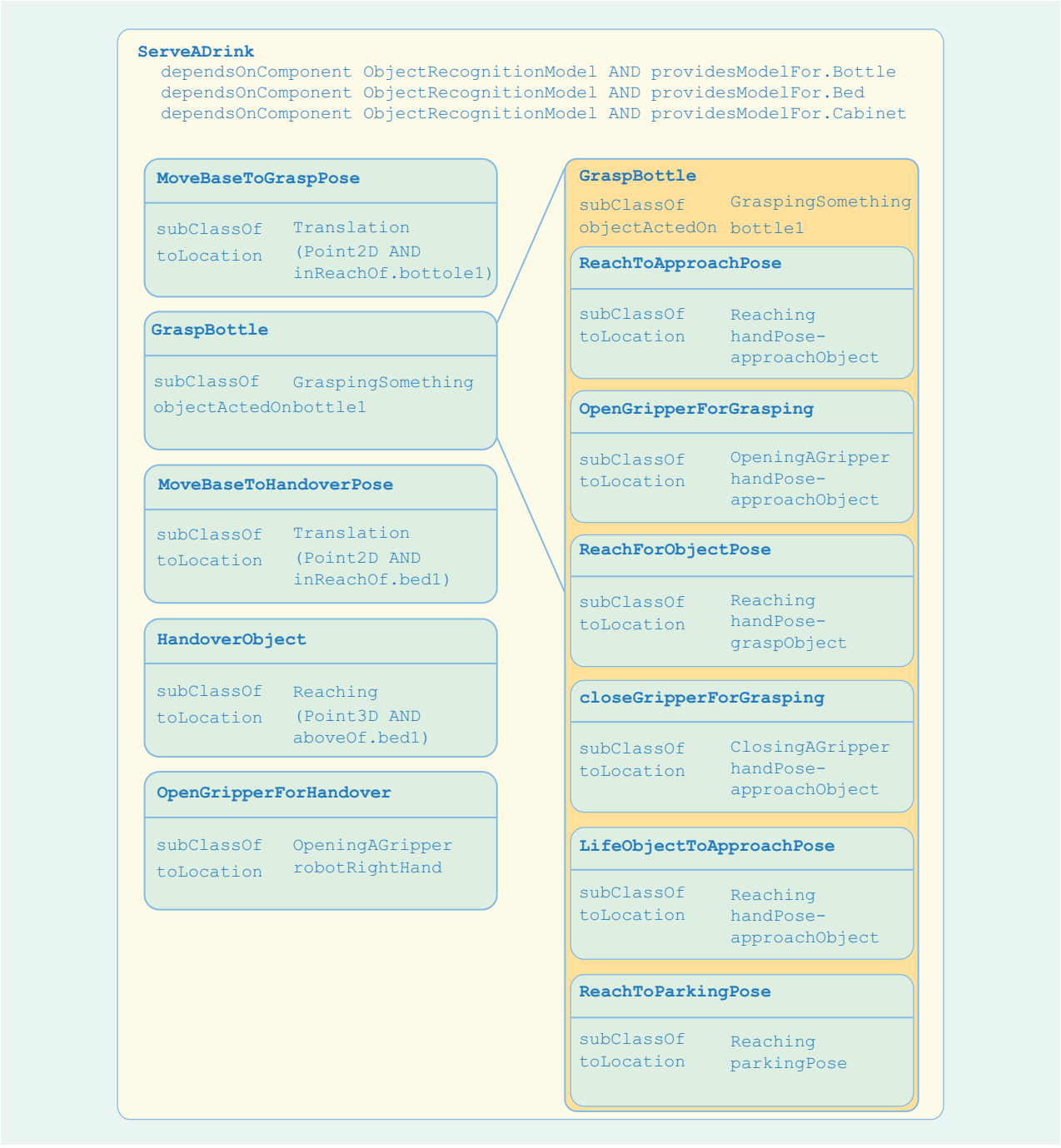


Figure 5. The action recipe used for the second demonstrator (see details in the "Demonstrators: Second Demonstrator" section).

type-specific subclass of an action such as `Translation`. Atomic actions, i.e., actions that are not composed from subactions, represent skills that translate these commands into motions (see the "Architecture: Generic Components—Action Execution" section for details).

Examples of the information stored in this second database include the action recipe used by robot A in the example detailed in the "RoboEarth: Example Scenario" section, the location of the bottle, and the number of times the patient was found in the bed.

Third, services that provide advanced learning and reasoning capabilities at the database level. A first type of service is illustrated by RoboEarth's reasoning server. It is based on KnowRob [40] and uses semantic information stored in the database to perform logical inference. This allows to determine if robot B in the example detailed in the "RoboEarth: Example Scenario" section meets the requirements needed to execute the action recipe used by robot A, or if it has the sensors needed to use the same object-recognition model.

Services may also solely operate on the database. RoboEarth's learning and reasoning service uses reasoning techniques [16], [40] to analyze the knowledge saved in the RoboEarth database and automatically generates new action recipes and updates prior information. For example, given multiple task executions, the database can compute probabilities for finding a bottle on top of the cupboard or on the patient's nightstand. Using the additional information that cups are likely to be found next to bottles, the service can automatically create a hypothesis for the probability of finding cups on top of the cupboard. Such cross correlations between objects can provide powerful priors for object recognition and help to guide a robot's actions. Additionally, if there are two action recipes that reach the same goal in different ways, the learning and reasoning service can detect this, fuse the recipes, and explicitly represent both alternatives. For example, if robot A was equipped with a dexterous manipulator but robot B only with a tray, the component could create a single action recipe "serve drink to patient" with two branches depending on the robot's abilities, which would have different requirements: the first branch would require a graspable bottle, whereas the second branch would require the availability of human or robotic help to place the bottle on the tray.

Interfaces

The RoboEarth database supports three types of interfaces. First, a Web interface that allows humans to exchange information with the database using hypertext mark-up language (HTML) forms. This interface may be used by a human to access the RoboEarth database and provide labels for robot A's perceptions of the bottle.

A second interface is based on the representational state transfer (REST) architecture [46]. As a stateless system, the REST interface transfers all necessary information as part of the request without the need to store any information on the server. This allows high scalability and fast response times, because it allows for a number of server-side optimizations (e.g., clusters with load balancing and fail over capabilities). The REST-style interface allows to encode data directly into JavaScript object notation (JSON) [47], which makes it suitable for interaction between robots and the database. Based on hypertext transfer protocol (HTTP) requests, this interface is easily implemented in all common programming languages and supported by most of today's robot platforms. Robot B's initial request for prior information about the hospital room as well as RoboEarth's response would both use REST.

Both the Web interface and the REST interface rely on the request–response message exchange pattern, which can result in unnecessary traffic because of periodic requests for updates. To avoid this kind of traffic, a third interface is based on the publish/subscribe message exchange pattern, which acts on changes to the database. This type of interface would allow robot B to subscribe to a notification for the patient leaving the hospital room to carry out a room-cleaning operation without constantly querying the database for the patient's location.

Generic Components

Action Execution

The execution of an action recipe (Figure 5) involves 1) determining and checking the availability of a set of skills required for the task, 2) finding an ordering among them that satisfies the ordering constraints, 3) linking a robot's perceptions to the abstract task description given by the action recipe, and 4) reliable action execution. The first and second steps can be resolved by RoboEarth's reasoning server (see the "Architecture: Database" section for more details). However, the third and fourth steps require a tight integration of RoboEarth's knowledge representation and the object recognition and action execution system used on the robot.

The action execution component ensures reliable action execution on the robot by coordinating communication with the RoboEarth database, monitoring the link between robot perceptions and actions, and providing failure-handling routines. In case of unresolvable problems, it asks for user input. For example, if robot B encounters an obstacle while trying to navigate to the bottle, the action execution component detects that the action recipe cannot be properly executed and asks for help. If the issue is solved, the component ensures a smooth continuation of the action recipe.

Environment Modeling

A world model component combines prior information available via the RoboEarth database with a robot's sensory inputs [6]. To guarantee fast and safe performance despite a potentially unreliable network connection to the RoboEarth database, it uses a two-level approach. A local world model supporting high update rates (typically tens of hertz) is executed on the robot. It merges and associates (possibly contradicting) sensor data from the robot's 3-D perception (e.g., the locations of labeled objects) with tracking data.

A second, global world model is maintained in the RoboEarth database. It updates the current local world model by factoring in suitable prior information stored in the RoboEarth database. This second level also provides simple reasoning capabilities to address issues such as object permanence. For example, if a bottle identified by a robot is occluded for 2 s, it is likely that the bottle on the cupboard has not changed. However, if the robot returns the next day, the probability of the appearance of the same bottle on that same cupboard has decreased significantly. Because of computational and communication constraints, these updates are typically provided at a slower rate (up to 1 Hz).

Semantic Mapping

The RoboEarth database provides generic action recipes defined in terms of the robot's skills, its environment, and the objects it is going to interact with. For example, the action recipe in the hospital scenario outlined in the "RoboEarth: Example Scenario" section is defined in terms of a cupboard, bed, bottle, and patient. RoboEarth implements a generic mapping component that uses monocular visual simultaneous localization and mapping (SLAM) to fuse partial observations of the environment with recognized objects downloaded from the database (Figure 6). This can be achieved by combining rich visual SLAM methods [48], [49], which produce geometrically accurate but typically semantically meaningless maps, with the semantically enriched models of recognized objects downloaded from RoboEarth. This allows the generation of semantic maps that can be used for localization by the robot's local world model in combination with basic skills such as "MoveTo" to navigate or execute a learned exploratory trajectory for mapping a given environment.

For example, in the hospital scenario mentioned in the "RoboEarth: Example Scenario" section, robot B may repeat part of robot A's movement trajectory to update the previous semantic map with potentially changed object locations, resulting in the rapid creation of a map with occupancy information even for areas not yet visited by robot B.

Learning

The repeated use of action recipes and skills on robot platforms with varying similarity offers many opportunities to improve the reliability and performance of action execution. A learning component allows the users to provide feedback on a robot's task performance. For example, if the robot uses an unsuitable grip position to grasp the bottle, the learning component allows a human to provide feedback and improve the robot's grasp performance.

In addition, action execution can be improved based on the experiences of other robots stored in the RoboEarth database as well as a robot's own experience. This allows the learning component to leverage the strategies and models used in skills and action recipes with data-based approaches by calculating feed-forward corrections to robot motions.

For example, the learning component allows a robot to access and integrate the experience of other similar robots grasping similar bottles into its corresponding grasp skill or to perform parts of a well-known trajectory in open-loop control to compensate for slow-sensor feedback.

Action and Situation Recognition and Labeling

The generation of useful, transferable action recipes and their successful reuse is a challenging task, because it must address all levels of a robot's actions, from the general skill abstraction layer to the high-level action execution layer. The recognition and labeling component is designed to simplify the generation of new action recipes by abstracting from the robot-specific, low-level layer and aggregating subsequent steps that belong to a complex task into an abstract task description. This is essential for learning new recipes by demonstration, either from observing the actions of another robot or those of a human controller.

The recognition and labeling component tackles this task from two sides. A low-level layer recognizes skills without any temporal or spatial relation using the signals given by the robot's sensors. A second, high-level layer then uses hidden Markov models (HMMs) to learn the (possibly nested) spatial and temporal relations of situations and actions using the already-recognized primitives of the lower level [50].

For example, when trying to enter the hospital room, robot A may find its path obstructed by a closed door. Following detection of the problem

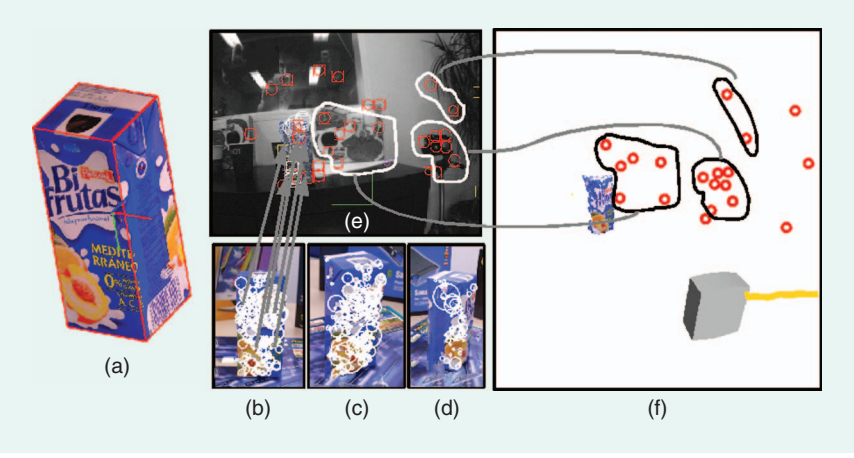


Figure 6. An example of semantic mapping as it was used for the second demonstrator (see the "Demonstrators: Second Demonstrator" section). (a) 3-D object model of a juice box bottle downloaded from RoboEarth. (b–d) Object images used for detection and the recognized sped up robust features (SURF) points. (e) Camera image and backprojection of the recognized object and the map points; notice the low semantic content for the map points. (f) General 3-D view showing some map points, the recognized object, and the camera's location (on the robot). (Photo courtesy of RoboEarth Consortium.)

and a request for user input, an operator uses a joystick to control the arm of the robot to open the door and creates an action recipe for "open a hospital room door." Sometime later, robot B may encounter the closed door of the hospital's kitchen area. The recognition and labeling component can now help this robot to detect that both situations are similar and, upon successfully opening the kitchen door, update the action recipe to reflect that knowledge.

Skill Abstraction Layer and Robot-Specific Components

The skill abstraction layer (Figure 1) provides a generic interface to a robot's specific, hardware-dependent functionalities. This is achieved by abstracting a robot's underlying hardware to offer a subset of standardized skills (e.g., MoveTo, MoveCloseToObject, Grasp, or Detect), sometimes also referred to as movement primitives, perception primitives, basis behaviors, or macro actions [51]. Skills accept input parameters, for example, to set the end goal of a movement or to define a grasp point. By providing a common interface to higher-level commands, skills play a key role in allowing the robots to successfully share and reuse action recipes. The specific subset of skills available in a specific robot is an important factor when selecting among different action recipes to perform a task.

For example, robot A may be equipped with an omnidirectional platform, whereas robot B may have a parallel drive. Using their respective MoveTo skills, both robots can nevertheless execute the same action recipe.

Demonstrators

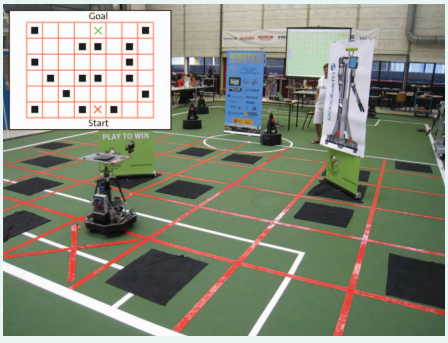
To provide a proof of concept of the benefits of sharing information via RoboEarth, we have implemented three prototypical demonstrators so far. The demonstrators used a knowledge base with reasoning engine based on KnowRob [40], the robot operating system (ROS) [52] for communication between generic components, and various robot-specific functionalities.

First Demonstrator

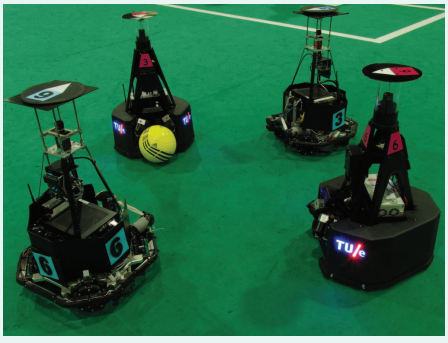
In a first demonstrator, two types of robot with different hardware and software configurations were tested in a maze exploration task (Figure 7). Both types of robot were preprogrammed with a basic set of skills using their respective RoboCup code [53], [54]. The skills were move 1 m forward, move 1 m back, move 1 m to the right, and move 1 m to the left. Both types of robot could autonomously detect maze fields blocked by black obstacles (a fifth skill) and used their respective RoboCup code for navigation and localization (i.e., the posters and red lines were not used). However, each robot was controlled by a generic action execution component (see the "Architecture: Generic Components—Action Execution" section for details), which coordinated the interaction of other robot-unspecific components, the execution of low-level, hardware-specific skills, and the exchange of information with the RoboEarth database. Each robot also used RoboEarth's generic components for environment modeling (more details can be seen in the "Architecture: Generic Components—Environment Modeling" section) and learning (see the "Architecture: Generic Components—Learning" section for more details) to improve its maze navigation.

Robots navigated the maze in turns. Each robot started on a predefined starting field and was allowed to continue moving through the maze until it reached a predefined target field.

To collaboratively learn the optimal path through the maze, the robots used Q-learning [57], with Q-learning states represented by the maze cells and Q-learning actions by the four-motion skills. At the start of the experiment, the first robot began moving through the maze using a random walk strategy. After each step of its navigation, the robot surveyed its four adjacent maze fields for obstacles and



(a)



(b)

Figure 7. (a) Setup for the first demonstrator. A 6 × 8 m maze consisted of 48 square fields outlined using red lines. Robots tried to find the shortest path from a predetermined starting field (indicated by the red cross) to a predetermined target field (indicated by the green cross) while avoiding blocked maze fields (indicated by black markers on the ground). (b) The two different robot platforms used [55], [56]. The University of Stuttgart's and the Technical University of Eindhoven's (USTUTT) RoboCup robots (first and third from left) use four omniwheels rather than the three omniwheels used by TU/e's robots. Both types of robots also use different robot operating software, preventing a direct transfer and reuse of knowledge. By defining a common set of hardware-specific skills, both types of robots could share a simple action recipe and information about their environment via RoboEarth. (Photo courtesy of RoboEarth Consortium.)

Figure 8. The second demonstrator. A robot serves a drink to a patient in a mock-up hospital room. By using prior knowledge stored in RoboEarth the robot could improve its navigation, object perception, and object manipulation capabilities and could efficiently serve a drink. (Photo courtesy of RoboEarth Consortium.)

stored them in the Q-matrix. If the robot did not reach the target field, it updated the value of its previous field with a small movement cost. Once it did reach the target field, the robot updated the entire Q-matrix [57] and uploaded it to the RoboEarth database (see the "Architecture: Database" section) hosted on a remote server. Subsequent robots exploring the maze first downloaded the latest version of the Q-matrix from the RoboEarth server and then used it to guide their maze exploration. Upon successful arrival at the target field, they similarly updated the Q-matrix with their knowledge and reuploaded it to RoboEarth.

Following multiple iterations of downloading the Q-matrix, navigating through the maze, and sharing of the improved Q-matrix, the values of the matrix converged. At the end of experiment, the converged matrix allowed all robots to move through the maze using the shortest path. (Videos of the demonstrator can be accessed at http://www.roboearth.org.)

Second Demonstrator

In a second demonstrator, a robot was asked to serve a drink to a patient in a mock-up hospital room (Figure 8). At the beginning of the experiment, the robot was preprogrammed with skills for movement, perception, and mapping (described in detail in the "Architecture: Generic Components—Semantic Mapping" section). However, it did not have any prior knowledge about its environment, relevant objects, or tasks.

At the start of the experiment, the written command `ServeADrink` was sent to the robot. The robot's action execution component (described in detail in the "Architecture: Generic Components—Action Execution" section) received and used the RoboEarth communication interface (see the "Architecture: Interfaces" section) to remotely access the RoboEarth database (see the "Architecture: Database" section). Similar to robot B mentioned in the example scenario in the "RoboEarth: Example Scenario" section, the robot then downloaded the relevant details, including the corresponding `ServeADrink` action recipe. Further relevant details were determined using logical reasoning based on a local knowledge base, a KnowRob installation [40]. As a result, the robot retrieved a map of the hospital room, approximate object positions gathered from previous trials, as well as perception and manipulation models for the cupboard, the bed, and the bottle (cf. Figure 6). The perception models were sent to the robot's semantic mapping component (see the "Architecture: Generic Components—Semantic Mapping" section), which started looking out for the given object. All perception instances were routed to the world modeling component (see the "Architecture: Generic Components—Environment Modeling" section), which was responsible for asserting the world state in the local knowledge base. The map was used by the robot's localization component to safely navigate the hospital room. The action execution component itself internally built a state machine from the action recipe and used it to trigger and monitor the execution of action primitives (see the "Architecture: Skill Abstraction Layer and Robot-Specific Components" section).

Despite differences between the downloaded information, such as changes in the map and variations in object locations, RoboEarth's world modeling component allowed the robot to use the downloaded information as a prior. For example, the last known location of the bottle was asserted in the local knowledge base and could be queried by the action execution framework to parameterize the `moveTo` action primitive as part of the `MoveBaseToGraspPose` subaction (Figure 5). This led to a bias of the robot's search strategy toward the most promising areas of the map. Similarly, downloading the known dimensions of the bed meant that detection of a single side of the bed was enough to provide occupancy information even for areas not covered by the prior map and not yet visited by the robot. The model of the bottle allowed the robot to reliably recognize and locate the bottle (Figure 6), use a suitable grasp point to pick it up, and apply a suitable grasp force. (Videos of the demonstrator can be accessed at http://www.roboearth.org.)

Third Demonstrator

In a third demonstrator, a compliant robot arm with accurate sensing capabilities was used to learn articulation models [58] for the drawers and doors of a cupboard (Figure 9). In a first step, a reference trajectory for the robot arm was generated based on an initial guess for the unknown articulation model. In subsequent steps, measurements of the compliant arm's deviation from this reference trajectory were used to continuously refine the model estimate in a closed-loop control structure. At the end of this learning stage, the learned articulation model of a specific door instance was attached to the door's object description and uploaded to the RoboEarth database.

In a second step, a service robot with less accurate sensing capabilities downloaded the stored articulation model and its parameters (e.g., door radius and axis of rotation) from the RoboEarth database. Using this knowledge, this

second robot was able to generate an open-loop trajectory and successfully open the cupboard's door. (Videos of the demonstrator can be accessed at http://www.roboearth.org.)

Open Issues

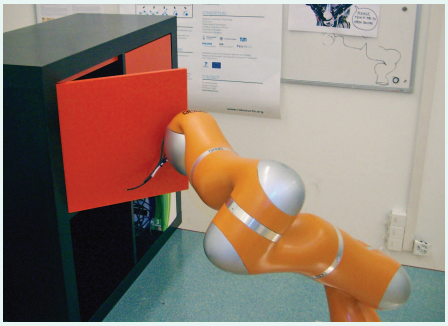
It is not only difficult to predict the benefits and opportunities but also the risks and challenges created by sharing data between robots. The global, open-source network outlined in this article will likely exacerbate current challenges concerning legal [59], [60], moral [61], [62], privacy [63], and safety [64], [65] aspects and create new and unexpected risks in the future. However, these risks are currently dwarfed by the technical challenges posed by creating a World Wide Web for robots.

Historically, robotics has been plagued by a lack of reusable components [66], [67]. Today, the code and algorithms used by robots are typically highly hardware dependent and difficult to reuse across platforms [68]. Not only the much-cited lack of a widely accepted robot operating system (ROS) [69], [70] but also the lack of standard interfaces to robot hardware components [71] continue to be great challenges in all areas of robotics. Such compatibility between robots is a necessary condition for many aspects of a truly World Wide Web for robots. It remains to be seen if the potential benefits of a World Wide Web-type database for robots can provide sufficient incentive to speed up the necessary process of interoperability and standardization, and to what extent a worldwide storage and an exchange of robot data can capitalize on the existing Internet standards.

Robots, unlike humans, excel at systematic and repetitive data collection. Although this provides unprecedented opportunities for obtaining consistent and comparable data as well as performing large-scale systematic analysis and data mining, the sheer amount of data collected by robotics systems far exceeds that processed by current information systems. Despite rapid progress [72], the technical feasibility of a system that connects robots worldwide remains unknown.

Another related issue is that of generating linked data [42]. Humans perceive objects in relation to their environment and other objects. However, allowing robots to learn the semantic (e.g., a bottle is a container to store liquid food), contextual (e.g., a bottle is more likely to be found on a cupboard than on the floor), and factual links (e.g., a bottle is not very heavy and can be picked up quite easily) between data remains a challenging research question.

Finally, despite a large number of potential benefits, it is difficult to predict to what extent sharing data between robots will be beneficial. Although data seem fundamental to robot learning and to extending robot performance beyond preprogrammed tasks, more information does not automatically translate into better robot performance [73]. Future experiments will help reveal currently unexpected drawbacks and the benefits of sharing data between robots.



(a)

(b)

Figure 9. The third demonstrator. (a) A compliant robot arm is used to estimate an articulation model of a cupboard's door by learning the right kinematic trajectory to open it. The robot then shares the model using RoboEarth. (b) A second, different robot arm, that has to interact with the same kind of cupboard downloads this model, reuses it to generate an open-loop trajectory, and successfully opens the cupboard. (Photo courtesy of RoboEarth Consortium.)

Conclusions

Connecting robots worldwide is a challenging task beyond the scope of any individual project. One main contribution of this article is conceptual, outlining the key requirements and potential benefits of the Internet for robots.

Another contribution of this article is a first demonstration of the feasibility of a World Wide Web for robots. In the first demonstrator, robots were able to successfully execute hardware-independent action recipes and could autonomously improve their task performance throughout multiple iterations of execution and knowledge exchange using a simple learning algorithm. This demonstrator illustrates how sharing data between multiple robots can lead to faster learning. In addition, it shows an example of how a robot can efficiently execute a task, such as navigating in a new environment, without explicit planning at design time. Once a good solution to navigate through the maze had been found, all robots could successfully navigate it even without a preprogrammed route-finding algorithm. The final result of this demonstrator was an improved, task-specific but platform-independent action recipe stored in the RoboEarth database.

The second demonstrator shows how prior knowledge can greatly increase the speed of performing complex tasks, such as serving a drink in the semistructured environment of

a hospital. It also shows how reasoning can allow a robot to execute a task that was not explicitly planned at design time.

The third demonstrator illustrates that robots with different hardware and control architectures can use a common database to share knowledge. It also shows how robots can create knowledge that is useful across different robot platforms and how robots benefit from the experience of other robots to interact with objects.

In addition to speeding up robot learning, sharing action recipes in an Internet database may also offer other benefits. By allowing component reuse across different systems and developers, human knowledge about the component usage, robustness, and efficiency is pooled. This allows incremental progress in performance and greatly reduces the time required to test new applications [74].

While RoboEarth focuses on the reuse of data, a future World Wide Web for robots will cover many more aspects of connecting robots. One area of particular interest is data collection and management. For example, a single pick-and-place task may automatically generate a series of image sequences under different conditions and angles, with and without partial occlusions. Logging of this partially labeled data may be useful to improve or aid identification of similar objects. Logging both successful and unsuccessful task performance may allow to determine bounds or rough estimates for object weight, center of gravity, elasticity, and surface friction, all of which may be useful to bias future object manipulations toward successful solutions. Logging context information, such as the location, time of day, or presence of other objects in the robot's vicinity, may be useful to generate priors to bias future object recognition or manipulation tasks for this or another type of robot that finds itself in a partially similar situation. On most existing robot platforms, such systematic data collection can be achieved during normal operation without significant overhead for robot performance. While the creation and maintenance of such data repositories is challenging, robot data sets may prove an invaluable resource for robotics.

RoboEarth's current architecture does not solve or address all challenges posed by creating a World Wide Web for robots. Rather, it is a first attempt at creating such a platform and will change and evolve with the wishes and requirements of the global robotics community. We strongly believe that a useful World Wide Web for robots can only be built in an open, iterative, and collaborative process.

Future work will focus on extending the capabilities of the RoboEarth database and on providing a series of demonstrators based on the hospital room scenario outlined in the "RoboEarth: Example Scenario" section. The RoboEarth database and supporting libraries for robot task execution will be released as open source at http://www.roboearth.org during mid-2011.

Acknowledgments

We thank the members of the RoboEarth Consortium [43] as well as Herman Bruyninckx, Koen Buys, Matei Ciocarlie, Elliot Duff, Caspar Garos, Brian Gerkey, and Thomas Henderson for discussions and comments on the manuscript. This research was funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no. 248942 RoboEarth.

References

[1] "World Robotics 2010," Int. Federation Robot. Statist. Dept., 2010.

[2] B. Gates, "A robot in every home," *Scientific Amer. Mag.*, vol. 296, no. 1, pp. 58–65, 2007.

[3] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, p. 9, 2008.

[4] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectr.*, vol. 45, no. 7, pp. 26–34, 2008.

[5] T. Buchheim, G. Kindermann, and R. Lafrenz, "A dynamic environment modelling framework for selective attention," in *Proc. IJCAI Workshop: Issues in Designing Physical Agents for Dynamic Real-Time Environments: World Modeling, Planning, Learning, and Communicating*, 2003, pp. 91–98.

[6] M. Roth, D. Vail, and M. Veloso, "A real-time world model for multi-robot teams with high-latency communication," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems 2003(IROS'03)*, vol. 3, pp. 2494–2499.

[7] J. Fenwick, P. Newman, and J. Leonard, "Cooperative concurrent mapping and localization," in *Proc. IEEE Int. Conf. Robotics and Automation 2002 (ICRA'02)*, vol. 2, pp. 1810–1817.

[8] H. Chang, C. Lee, Y. Hu, and Y. Lu, "Multi-robot SLAM with topological/metric maps," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems 2007 (IROS'07)*, pp. 1467–1472.

[9] L. Parker, K. Fregene, Y. Guo, and R. Madhavan, "Distributed heterogeneous sensing for outdoor multi-robot localization, mapping, and path planning," in *Proc. 2002 NRL Workshop Multi-Robot Systems: From Swarms to Intelligent Automata*, Washington, DC (*Multi-Robot Systems: From Swarms to Intelligent Automata,* A. C. Schultz and L. E. Parker, Eds.) Norwell, MA: Kluwer, Mar. 2002, p. 21.

[10] T. Grundmann, R. Eidenberger, M. Schneider, M. Fiegert, and G. v. Wichert, "Robust high precision 6D pose determination in complex environments for robotic manipulation," in *Proc. Workshop Best Practice in 3D Perception and Modeling for Mobile Manipulation at the Int. Conf. Robotics and Automation*, 2010, pp. 1–6.

[11] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2010, pp. 2155–2162.

[12] M. Everingham, A. Zisserman, C. K. I. Williams, L. Van Gool, et al., "The 2005 PASCAL visual object classes challenge," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment* (LNAI 3944), J. Quinonero-Candela, I. Dagan, B. Magnini, and F. d'Alche-Buc, eds. Berlin: Springer-Verlag, 2006, pp. 117–176.

[13] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[14] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: A database and web-based tool for image annotation," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 157–173, 2008.

[15] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, Tech. Rep. 7694, 2007.

[16] J. Geusebroek, G. Burghouts, and A. Smeulders, "The Amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.

[17] M. Merler, C. Galleguillos, and S. Belongie, "Recognizing groceries in situ using in vitro training data," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[18] T. Yeh, K. Tollmar, and T. Darrell, "Searching the web with mobile images for location recognition," in *Proc. IEEE Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. II-76–II-81.

[19] A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 30, no. 11, pp. 1958–1970, 2008.

[20] C. Goldfeder, M. Ciocarlie, H. Dang, and P. Allen, "The Columbia grasp database," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2009, pp. 1710–1716.

[21] M. Ciocarlie, G. Bradski, K. Hsiao, and P. Brook, "A dataset for grasping and manipulation using ROS," in *Proc. Workshop RoboEarth—Towards a World Wide Web for Robots at the Int. Conf. Intelligent Robots and Systems*, 2010.

[22] M. Cummins and P. Newman, "Highly scalable appearance-only SLAM-FAB-MAP 2.0," in *Proc. Robotics Science and Systems*, 2009.

[23] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *Int. J. Robot. Res.*, vol. 29, no. 2–3, pp. 298–320, 2010.

[24] I. Posner, D. Schroeter, and P. Newman, "Using scene similarity for place labelling," in *Experimental Robotics*, (ser. Springer Tracts in Advanced Robotics), O. Khatib, V. Kumar, and D. Rus, Eds. Berlin: Springer-Verlag, 2008, vol. 39, pp. 85–98.

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

[26] M. Martinez, A. Collet, and S. Srinivasa, "MOPED: A scalable and low latency object recognition and pose estimation system," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2043–2049.

[27] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, 2003.

[28] L. Barrington, R. Oda, and G. Lanckriet, "Smarter than genius? Human evaluation of music recommender systems" in *Proc. Int. Symp. Music Information Retrieval*, 2009, pp. 357–362.

[29] 2011"Google Goggles—Use pictures to search the web," [Online]. Available: http://www.google.com/mobile/goggles/

[30] S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, S. Seitz, and R. Szeliski, "Reconstructing Rome," *Computer*, vol. 43, no. 6, pp. 40–47, 2010.

[31] L. Fei-Fei, R. Fergus, and P. Perona, "A Bayesian approach to unsupervised one-shot learning of object categories," in *Proc. IEEE Int. Conf. Computer Vision*, 2003, vol. 2p. 1134.

[32] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Comput. Vis. Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[33] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from Google's image search," in *Proc. 10th IEEE Computer Society Int. Conf. Computer Vision (ICCV'05)*, 2005, pp. 1816–1823.

[34] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *Int. J. Comput. Vis.*, vol. 87, no. 3, pp. 284–303, 2010.

[35] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 2161–2168.

[36] X. Fan and T. Henderson, "RobotShare: A Google for robots," *Int. J. Humanoid Robot.*, vol. 5, no. 02, pp. 311–329, 2008.

[37] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. Kong, A. Kumar, K. Meng, and G. Kit, "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 3084–3089.

[38] S. Dalibard, A. Nakhaei, F. Lamiraux, and J. Laumond, "Manipulation of documented objects by a walking humanoid robot," in *Proc. 10th IEEE-RAS Int. Conf. Humanoid Robots*, Nashville, TN, 2010, pp. 518–523.

[39] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-the planning domain definition language," in *Proc. AIPS-98 Planning Competition Committee*, 1998.

[40] M. Tenorth and M. Beetz, "KnowRob—Knowledge processing for autonomous personal robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) 2009*, pp. 4261–4266.

[41] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, and M. Beetz, "ORO, a knowledge management module for cognitive architectures in robotics," in *Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010, pp. 3548–3553.

[42] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data—The story so far," *Int. J. Semantic Web Inform. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[43] R. v. d. Molengraft, G. Angelis, M. Beetz, R. D'Andrea, A. Knoll, R. Lafrenz, P. Levi, J. M. M. Montiel, H. Sandee, M. Steinbuch, J. D. Tardós Solano, M. Tenorth, M. Waibel, and O. Zweigle, (2010). The RoboEarth Consortium [Online]. Available: http://www.roboearth.org

[44] W3C OWL Working Group. (2009 Oct.). OWL 2 web ontology language document overview, W3C Recommendation [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/

[45] A. C. Murthy, C. Douglas, D. Cutting, D. Das, D. Borthakur, E. Collins, E. Soztutar, H. Kuang, J. Homan, M. Konar, N. Daley, O. O'Malley, P. Hunt, R. Angadi, S. Agarwal, K. Shvachko, M. Stack, T. W. (Nicholas) Sze, T. Lipcon, T. White, and Z. Shao, Apache Hadoop, a framework for reliable, scalable and distributed computing [Online]. Available: http://hadoop.apache.org

[46] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, 2000.

[47] D. Crockford. (2006, July). RFC4627—The application/json Media Type for JavaScript Object Notation (JSON) [Online]. Available: http://www.ietf.org/rfc/rfc4627.txt?number=4627

[48] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE and ACM Int. Symp. Mixed and Augmented Reality 2007 (ISMAR'07)*, pp. 225–234.

[49] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-Point RANSAC for EKF filtering. Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.

[50] T. Huenhagen, I. Dengler, A. Tamke, T. Dang, and G. Breuel, "Maneuver recognition using probabilistic finite-state machines and fuzzy logic," in *Proc. IEEE Intelligent Vehicles Symp. (IV) 2010*, pp. 65–70.

[51] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philos. Trans. R. Soc. London. B, Biol. Sci.*, vol. 358, no. 1431, p. 537, 2003.

[52] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Software*, 2009.
[53] (2010). TechUnited Eindhoven Robocup code repository [Online]. Available: http://robocup.wtb.tue.nl/svn/techunited/
[54] (2010). RFC Stuttgart Robocup code tarball [Online]. Available: http://robocup.informatik.uni-stuttgart.de/
[55] (2010). RFC Stuttgart [Online]. Available: http://robocup.informatik.uni-stuttgart.de/rfc/
[56] (2010). TechUnited [Online]. Available: http://www.techunited.nl/en
[57] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, 1998.
[58] J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard, "Learning kinematic models for articulated objects," in *Proc. Int. Conf. Artificial Intelligence (IJCAI)*, 2009, pp. 1851–1855.
[59] A. Mavroforou, E. Michalodimitrakis, C. Hatzitheo-Filou, and A. Giannoukas, "Legal and ethical issues in robotic surgery," *Int. J. Angiology*, vol. 29, no. 1, pp. 75–79, 2010.
[60] M. R. Calo. (2010, Nov. 9). Open robotics. *Maryland Law Rev.* [Online]. *70(3)*. Available: http://ssrn.com/abstract=1706293
[61] M. R. Calo. (2010, Apr. 2). Robots and privacy. *Robot Ethics: The Ethical and Social Implications of Robotics*, P. Lin, G. Bekey, and K. Abney, Eds. Cambridge, MIT Press [Online]. Available: http://ssrn.com/abstract=1599189
[62] R. Arkin, "Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture," in *Proc. 3rd ACM/IEEE Int. Conf. Human Robot Interaction*, Amsterdam, New York: ACM, 2008, pp. 121–128 [Online]. Available: http://hri2008.org
[63] D. S. Syrdal, M. L. Walters, N. Otero, K. L. Koay, and K. Dautenhahn, "He knows when you are sleeping—Privacy and the personal robot companion," in *Proc. Workshop Human Implications of Human-Robot Interaction, Association for the Advancement of Artificial Intelligence (AAAI'07)*, 2007, pp. 28–33.
[64] K. Ikuta, H. Ishii, and M. Nokata, "Safety evaluation method of design and control for humancare robots," *Int. J. Robot. Res.*, vol. 22, pt. 5, pp. 281–297, 2003.
[65] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Safety evaluation of physical human-robot interaction via crash-testing," in *Proc. Int. Conf. Robotics: Science and Systems*, 2007, pp. 217–224.
[66] Computing Community Consortium. (2008). A roadmap for US robotics: From Internet to robotics [Online]. Available: www.us-robotics.us/reports/CCC%20Report.pdf
[67] R. Bischoff, T. Guhl, A. Wendel, F. Khatami, H. Bruyninckx, B. Siciliano, G. Pegman, M. Hägele, E. Prassler, J. Ibarbia, C. Leroux, B. Tranchero, A. Knoll, and R. Lafrenz, "euRobotics—Shaping the future of European robotics," in *Proc. Int. Symp. Robotics/ROBOTIK*, 2010, pp. 728–735.
[68] E. Prassler, R. Bischoff, and H. Bruyninckx, "It's a long way to useful standards," *IEEE Robot. Automat. Mag.*, vol. 17, no. 3, pp. 18–19, 2010.
[69] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for robotics: A survey," in *Proc. Int. Conf. Robotics, Automation and Mechatronics*, 2008, pp. 736–742.
[70] W. D. Smart, "Is a common middleware for robotics possible?" in *Proc. Workshop Measures and Procedures for the Evaluation of Robot Architectures and Middleware at the Int. Conf. Intelligent Robots and Systems*, 2007.
[71] E. Prassler, H. Bruyninckx, K. Nilsson, and A. Shakhimardanov, "The use of reuse for designing and manufacturing robots," *White Paper*, 2009.
[72] D. Ferrucci, "Build Watson: An overview of DeepQA for the Jeopardy! challenge," in *Proc. 19th Int. Conf. Parallel Architectures and Compilation Techniques*, New York: ACM, 2010, pp. 1–2.
[73] A. Schöllig, J. Alonso-Mora, and R. D'Andrea, "Limited benefit of joint estimation in multi-agent iterative learning control," *Asian J. Control (Special Issue on Iterative Learning Control)*, vol. 15, no. 4, pp. 1–11, July 2013.
[74] D. Brugali and P. Scandurra, "Component-based robotic engineering, Part I: Reusable building blocks," *IEEE Robot. Automat. Mag.*, vol. 16, no. 4, pp. 84–96, 2009.

Markus Waibel, ETH Zurich, Switzerland. E-mail: mwaibel@ethz.ch.

Michael Beetz, Technische Universität München, Germany. E-mail: beetz@in.tum.de.

Javier Civera, Universidad de Zaragoza, Spain. E-mail: jcivera@unizar.es.

Raffaello D'Andrea, ETH Zurich, Switzerland. E-mail: rdandrea@ethz.ch.

Jos Elfring, Eindhoven University of Technology, The Netherlands. E-mail: J.elfring@tue.nl.

Dorian Gálvez-López, Universidad de Zaragoza, Spain. E-mail: dorian@unizar.es.

Kai Häussermann, University of Stuttgart, Germany. E-mail: kai.haeussermann@ipvs.uni-stuttgart.de.

J.M.M. Montiel, Universidad de Zaragoza, Spain. E-mail: josemari@unizar.es.

Alexander Perzylo, Technische Universität München, Germany. E-mail: perzylo@cs.tum.edu.

Björn Schießle, University of Stuttgart, Germany. E-mail: schiessle@ipvs.uni-stuttgart.de.

Moritz Tenorth, Technische Universität München, Germany. E-mail: tenorth@cs.tum.edu.

Oliver Zweigle, University of Stuttgart, Germany. E-mail: Oliver.Zweigle@ipvs.uni-stuttgart.de.

René van de Molengraft, Eindhoven University of Technology, The Netherlands. E-mail: m.j.g.v.d.molengraft@tue.nl.

Rob Janssen, Eindhoven University of Technology, The Netherlands. E-mail: r.j.m.janssen@tue.nl.