Accepted Manuscript

Query-based integration of heterogeneous knowledge bases for search and rescue tasks

Fereshta Yazdani, Sebastian Blumenthal, Nico Huebel, Asil Kaan Bozcuoğlu, Michael Beetz, Herman Bruyninckx

PII: S0921-8890(18)30348-8
DOI: <https://doi.org/10.1016/j.robot.2019.03.013>
Reference: ROBOT 3191

To appear in: *Robotics and Autonomous Systems*

Please cite this article as: F. Yazdani, S. Blumenthal, N. Huebel et al., Query-based integration of heterogeneous knowledge bases for search and rescue tasks, *Robotics and Autonomous Systems* (2019), <https://doi.org/10.1016/j.robot.2019.03.013>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Query-based Integration of Heterogeneous Knowledge Bases for Search and Rescue Tasks

Fereshta Yazdani^a, Sebastian Blumenthal^{b,c}, Nico Huebel^b, Asil Kaan Bozcuoğlu^a, Michael Beetz^a, Herman Bruyninckx^{b,d}

^a*Institute for Artificial Intelligence, University of Bremen,
Am Fallturm 1 (TAB E), 28359 Bremen, Germany*

^b*Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300,
B-3001 Heverlee (Leuven), Belgium*

^c*Locomotec GmbH, Bergiusstraße 15, 86109 Augsburg, Germany*

^d*Department of Mechanical Engineering, Eindhoven University of Technology,
5612 AZ Eindhoven, Netherlands*

Abstract

Many complex robotic scenarios require the integration of heterogeneous knowledge from various sources and their often domain-specific reasoning methodologies. This paper introduces an integrated knowledge processing methodology that uses query mechanisms and model-to-model transformations to process heterogeneous knowledge bases. These mechanisms are introduced for an outdoor scenario where knowledge and reasoning methods from Open Street Map (OSM), Digital Elevation Model (DEM), Robot Scene Graph (RSG), KNOWROB with its SHERPA ontology, and OPENEASE were integrated for.

Keywords: Querying, Big Data, Knowledge Sharing, Knowledge Management, Knowledge Maintenance

1. INTRODUCTION

Together with the advancements in robotics and related technologies, robotic systems become more and more important for many applications in in- and out-

Email addresses: yazdani@cs.uni-bremen.de (Fereshta Yazdani), blumenthal@locomotec.com (Sebastian Blumenthal), nico.huebel@kuleuven.be (Nico Huebel), asil@cs.uni-bremen.de (Asil Kaan Bozcuoğlu), beetz@cs.uni-bremen.de (Michael Beetz), herman.bruyninckx@kuleuven.be (Herman Bruyninckx)

Pre-proof submitted to Issue on Semantic Policy and Action Representations for Autonomous Robots September 23, 2018

Figure 1: A simplified example of a SHERPA architecture. A heterogeneous team collaborates in a search and rescue mission. The agents are heterogeneous with respect to their capabilities and, most important for this work, their software components which they are running or able to run.

door environments. One such application is the deployment of rescue robots in outdoor missions to find survivors after accidents. The SHERPA project (Marconi et al., 2012) is an example where a rescue team of ground and multiple aerial robots working alongside human rescuers are deployed in order to find victims of avalanches in the Alps. The SHERPA robots are equipped with different hard- and software modules to accomplish certain tasks in the context of the mission, e.g. a camera for taking pictures or a symbolic knowledge base for recording and sharing sensor data from and between robots. Using these modules, robots are expected to correctly reason and act within the course of rescue missions. To correctly perform tasks in these missions, robots need to combine various knowledge sources from sensor data, geospatial data, other robots and also human input. In this work, we propose an integrated methodology that enables robots to reason and act (i.e. perform tasks) autonomously by integrating multiple knowledge sources at query time. Figure 1 depicts a communication diagram of the agents and their available software components during a typical SHERPA mission. These components are revisited in more detail in Figure 3.

Starting from early childhood, people possess common-sense knowledge that allows to understand abstract tasks, interpret and recognize scenes, make decisions about their tasks and how to achieve. For successful collaborative missions, robots need to have a certain degree of common-sense knowledge to understand, interpret and convert abstract human instructions into low-level data for algorithms. They should be able to understand and ground humans' instructions into their own perception and knowledge. This behavior would facilitate the seamless coor-

dination in human-robot teams. For instance, in order to correctly understand and perform the instruction “Go to that house”, a robot needs to ground this instructional description into concrete action parameters, like metric points in space, that its navigation routines can use. It requires knowledge from different sources at different levels of abstraction, e.g. linguistic, semantics, geographics etc. In our example, these sources would be the input from the operator’s pointing device for inferring the direction and Open Street Map for inferring the closest house at that direction. Integrating knowledge from different sources and at different levels of abstraction, however, leads to several challenges:

- *No common information encoding.* Knowledge can be stored in different representations and formats, e.g. in Web Ontology Language (OWL)¹, in RDF triples (Klyne and Carroll, 2004), as JSON objects (Bray, 2014) or XML files (Bray et al., 2006). This enables the representation of different kinds of knowledge at different abstraction levels for various situations. Much of the knowledge, e.g., about algorithms or software configurations, is not formally encoded at all but only known by the human developers or accessible in human readable documents. However when working in heterogeneous teams, these different information encodings need to be dealable by all the team members in order to have a common ground to work with.
- *Incomplete and unreproducible data.* Knowledge bases are often coupled with inference mechanisms that affiliate with specific knowledge bases and generate data obtainable from the corresponding origin. However, integrating and merging knowledge bases with different world models into a consistent form, leads to a loss of the functionality of the mechanisms. The mechanisms would lose their capability of being resource-bounded in order to generate derivable and comprehensible results. Those results might be unobtainable and not exactly reproducible by the corresponding knowledge base anymore.
- *No explicit mapping theory.* There is no information about how various domain specific knowledge sources are connected or linked with each other within a domain or between domains. A traditional approach is mostly given by connecting and mapping facts and terms from one source or domain to another manually. This approach, however, is often time consuming and also combined with high costs.

¹<https://www.w3.org/TR/2004/REC-owl-features-20040210/>

- *Ambiguity in terminology of representations.* Different domains use different words and definitions that can lead to vagueness and collisions. In some domains the same terms can have different definitions while other different terms can mean the same. Combining and unifying these representations without losing their expressiveness of terms is challenging and also almost impossible.

There is a need for more than just combining various knowledge bases that have different structures, concepts and taxonomies. Each knowledge base provides distinctive reasoning capabilities. Therefore, integrating heterogeneous knowledge and their associated reasoning capabilities in different missions is also inevitable for many complex robotic tasks.

In this paper, we propose an integrated knowledge processing methodology for rescue robots within the SHERPA project. This methodology enables humans and robots to query, process, reason and make inferences about different knowledge bases on demand before, during or after rescue missions. It uses two approaches to integrate various knowledge bases, (1) query-based integration (used to integrate the Robot Scene Graph (RSG) (Blumenthal and Bruyninckx, 2014) and KNOWROB (Tenorth and Behr, 2013)) and (2) model-to-model transformations (used to integrate OSM into the RSG). Based on this methodology the contributions of this paper are as follows:

- Enabling human and robot teams to work with knowledge bases that cover different levels of abstraction, e.g., high-level knowledge as symbolic description as well as topological view of the mission together with low-level data compatible with robot control executives. This heterogeneous knowledge base enables grounding high-level knowledge in low-level information and vice versa.
- Distributing this heterogeneous knowledge base over different agents. Such a distributed knowledge base will extend the capabilities for robots with different reasoning and inference mechanisms to retrieve, process and update knowledge.
- A comprehensive analysis of requirements, design decisions and their trade-offs for combining heterogeneous knowledge bases.

2. RELATED WORK

Acquiring and reasoning about knowledge are two of the key capabilities towards intelligent systems. Thus, supplying knowledge to robot control programs is a well-studied topic in intelligent robotics. Stanford's Shakey (Nilsson, 1984) can be named as one of the pioneering works where the map of the block world that Shakey operates in is supplied as first-order logic sentences. There are also early studies by Bekey *et al.* (Bekey *et al.*, 1993) where a knowledge-driven grasp planner is presented and by Hoffmann *et al.* (Hoffmann and Bristler, 1997) where a fuzzy knowledge base for mobile robots is proposed.

As robots start to accomplish more and more complex tasks, more knowledge is necessary to perform these task executions in different circumstances. On one hand, supplying such knowledge from various sources is an obvious challenge for roboticists and engineers (Bateman *et al.*, 2017). On the other hand, yet a greater challenge is to implement control programs that are able to harmonize and make use of such heterogeneous knowledge bases (Nyga *et al.*, 2017).

The RoboEarth project (Waibel *et al.*, 2011) was one of the first projects in robotics and research that dealt with the overall challenge how to share information over the internet among different robots. OPENEASE (Beetz *et al.*, 2015) is a web-based knowledge service and provides an interface to different robotic systems to share several knowledge bases and reasoning components between each other. The STRANDS project deals with changes in a human environment over time (Krajník *et al.*, 2017). The TRADR project (Kruijff-Korbayová *et al.*, 2015) focuses on the persistence of environment models, multi-robot action models, and human-robot teaming for long-term search and rescue missions.

However, most projects propose their own way for representing and storing knowledge (Patil *et al.*, 1992). Work on integrating such existing knowledge bases is sparse. (Bagosi *et al.*, 2015) is suggesting a generic interface for knowledge representation but is focusing on the integration of an arbitrary but single knowledge base. (Dastani *et al.*, 2009) is proposing an integration approach similar to the one in this paper but with a focus on symbolic knowledge.

Beyond the scope of robotics, there are also studies that make use of heterogeneous knowledge-bases. Xu *et al.* (2016) propose a methodology to detect urban emergency events based on a knowledge base which is created by the posts in social media. A deep learning-based integration of knowledge sources with different semantics is presented in (Zhang *et al.*, 2016) for recommender systems. Li *et al.* (2017) propose a multi-source reasoning model that chooses a set of available reasoning rules for the given context in order to understand common

sense in natural-language texts.

Also other areas, such as bioinformatics, are concerned with the idea of integrating data from various sources. In (Davidson et al., 1993), for example, a solution was proposed to transform data to a federated database on demand. The basic steps are to transform bioinformatics data to a common data model, match of semantically related objects, schema integration, transform data to a federated database and match semantically equivalent data. This idea is already applied in the Bio2RDF project (Belleau et al., 2008) that is a semantic web approach where data is transformed to the RDF format. Another example is given by (Stein, 2003) where three essential approaches for data integrators are mentioned: link integration - e.g. by using the linking capability of the web, view integration - e.g. the creation of portals aggregating the information, and data warehousing - e.g. storing everything in a single unified database. Another approach Stein also proposed is the “knuckles and nodes” that is about building databases of links between data but not storing any of it. This strategy is very similar to the Bio2RDF project described above.

Our proposed methodology is distinguished from the aforementioned works in the following directions: First, we use heterogeneous knowledge bases that are distributed among different robotic agents. Second, our methodology integrates these knowledge bases on-demand using queries. Namely, when a robotic agent or the human makes a query to a knowledge base, this query is changed (both semantically and syntactically) and redirected to another in case that the knowledge is not available at that particular knowledge base. In (Blumenthal et al., 2016), we have presented a predecessor approach where different knowledge bases are integrated not with queries but by model composition, model-to-model transformations and an application dependent bridge component. Such as in other strategies, described above, we also consider knowledge in three different representations from structural, textual to visual. However, we do not represent a framework with capabilities that is able to learn the latent representations in collaborative filtering. Such as in (Liu et al., 2017), we use queries for a specific reasoning context, in our case in the SHERPA project. We show how our query-based methodology can be used by different agents to improve their own software and hardware capabilities.

3. REQUIREMENTS and DESIGN DECISIONS

3.1. Requirements

For the human-robot collaborative rescue missions, we have identified the following design requirements for the knowledge base:

1. The knowledge base must be designed as a **distributed system**. Software components and knowledge gathered by individual robots is spread among the team members. Further, since the communication is based on WiFi, the bandwidth is limited and can vary depending on how distant team members are from each other. The knowledge base must be **partition tolerant**, i.e., at least a robot's knowledge of its environment remains locally accessible in case of communication loss.
2. Different platforms have different **hardware** resource constraints. In our scenario a UAV will not be able to deploy all semantic reasoning components.
3. The overall knowledge has to be available for a human rescuer. Since the human rescuer will often deal with multiple aspects and different geographic areas at the same time, the system must provide a high degree of autonomy to robotic agents. Speech commands like "go to that tree" are required to be interpreted as correct task plans with appropriate parametrizations using advanced **semantic** reasoning capabilities.
4. The scenario is time critical. We cannot assume to first create a complete map and then perform the mission. Thus, **existing map formats** must be exploited as much as possible for outdoor missions.
 - (a) We need semantic tags like forest or road.
 - (b) We need height information for the path planning of UAVs.
5. People need to be able to inspect what has happened in the past, e.g., if an area has been visited by a UAV already. The world model needs to serve as an **episodic memory**.

To summarize the requirements (also see Fig 2): the knowledge can be differentiated along a partition dimension (with the most common cases being centralised systems depending on central agent or computer, networked systems with independently acting agents that are coordinated by a central unit, and fully distributed systems of independently acting agents) and along an Abstraction dimension with knowledge from different sources on different semantic levels ranging from, e.g. metric height map data to symbolic representations. Finally, it is also heterogeneous in its Update frequency ranging from frequently updated sensor data to static or rarely changing knowledge like maps and ontologies.

3.2. Decision Decisions

Based on the aforementioned requirements, we have applied the following design decisions to the presented methodology:

Figure 2: This figure shows the main dimensions for requirement and design decisions concerning data or knowledge. We will use the terms data, information, and knowledge somewhat interchangeably. For a discussion of the differences see for example (Rowley, 2007; Zins, 2007). Please note that these are not absolute properties but become design choices that can be different for each part of an application. The axis are also a continuum and the ticks along them indicate examples and not discrete points.

1. We interface with KnowRob for the semantic reasoning and use RSG as distributed world model within episodic memories because they provide well-established solutions for the requirements 1, 3 and 5.
2. We split the concerns of semantic reasoning and distributed episodic memory. This allows to realise different deployments if the hardware resources are too scarce (Requirement 2). Further, it allows to reuse the modules in other applications and they can be tested and developed individually. The coupling between those modules should be loose.
3. Since a single map format does not contain all relevant information, we make use of two map formats: Digital Elevation Models (DEM) for elevation information and semantically richer, topological OpenStreetMap (OSM) data (Requirement 4). Since the knowledge inside these maps is static, we use it as the basis for our distributed world model. Further, this step can be omitted if resource are scarce (Requirement 2).

4. Knowledge Bases

In Fig. 3 is shown the proposed architecture of a SHERPA mission. Knowledge about a map is loaded into the Robot Scene Graph (RSG) from OpenStreetMap (OSM) and Digital Elevation Models (DEM) while a-priori defined RSG models are loaded on start up as well. The RSG is a distributed world model, so it is connected to the other robots via a Communication Mediator which allows to

Figure 3: Architecture of the overall system denoted as Digital Cognitive Map (DCM), which integrates the heterogeneous knowledge sources in SHERPA. The core components KnowRob and Robot Scene Graph (RSG) are separated and loosely coupled via query interfaces. KnowRob extends the systems capabilities along the **abstraction** dimension by allowing semantic reasoning as well as grounding symbolic knowledge in data, which is often queried from the RSG. The RSG is keeping a history of the (robot specific) **frequent updates** as well as dealing with **partitioning**, i.e., the distributed team members. Note, the chosen modularity and loose coupling allows for different deployments. E.g., a UAV does not have a KnowRob component due to its limited hardware resources. Further details on the core components and their integration based on query interfaces can be found in Section 4.

exchange information between team members. Details about the Communication Mediator are beyond the scope of this paper. This combined information equated with knowledge can be queried using the RSGQuery interface by using JSON messages (see Section 5). In SHERPA the RSG serves as an episodic memory for KNOWROB (Fenrich and Beetz, 2013) that can answer semantic queries on a higher abstraction level by providing ontologies to the reasoning process. KNOWROB queries are formulated in Prolog. The commander can access KNOWROB via the OPENEASE web interface (Beetz et al., 2015) that provides a query interface and visualization tools for humans.

The properties of the different knowledge bases are summarized in the following. Details of the query interfaces are explained in Section 5.

4.1. OpenStreetMap (OSM)

OSM is a community driven effort to collect and publish geographic data and knowledge. In SHERPA this data serves as a static and semantic environment representation. OSM stores the static and topological information in a graph structure. The core elements are nodes, ways, and relations.

Nodes represent static positions in the map, typically derived from GPS measurements. Ways denote polygon features in order to represent paths, rivers or buildings. Relations are used to store general purposes relations between elements, e.g. roads that reference multiple way segments.

Each above listed element can have a list of associated key-value pairs called tags to further define the semantic meaning of the elements. The list of valid tags² is maintained by a collaborative community effort. In SHERPA only a subset which contains paths, forests, rivers and buildings are important. This list can be seen as a geographic knowledge base relevant for the mission. Though, as a knowledge base it does not have a deep class taxonomy as it has only one level of sub classing. OSM does not contain height information of the terrain.

4.2. Digital Elevation Models (DEMs)

DEMs represent the geometric surface of terrain with different possible representations. Here, the focus is on heightmaps. In this work heightmaps are obtained from GeoTIFF files accompanied with meta data in the file header to reconstruct the geo locations. In contrast to OSM, DEMs do not contain semantic information.

4.3. Robot Scene Graph (RSG)

The RSG (Blumenthal and Bruyninckx, 2014) is a distributed world model that is able to answer queries about which objects exist in the environment, where they are, and when they have been at a particular place. This capability for spatio-temporal reasoning is essential to enable reasoning about the world during a rescue mission. While most world models are static in terms of knowledge, the RSG is a dynamic component that can also run services on its existing knowledge and take decisions based on monitoring functions. The monitoring capability is also used to synchronize multiple world models by sending local changes to the other peers. While the OSM and DEM provide a static knowledge about the environment, the RSG includes the acquired knowledge about the environment, mission, and the team members.

The design approach for the RSG is to use a graph structure that is composable with domain specific models (Blumenthal and Bruyninckx, 2014). The nodes of this graph have the meaning of, e.g., objects or measurements, while edges represent relations between them. Here, composable means that domain specific

²http://wiki.openstreetmap.org/wiki/Map_Features

aspects like **geometry**, **topology** or **semantic** tags are composed with graph primitives, i.e., nodes and relations. A few examples are given below. Further details can be found in (Blumenthal et al., 2016; Blumenthal and Bruylants, 2014).

- A RSG Transform is a **geometric** relation storing a relative pose between two objects.
- **Topological** relations like paths or search areas are relations stored as RSG Connections which refer to an ordered set of nodes.
- **Semantic** tags are represented as Attributes by RSG. Multiple namespaces are distinguished by adding a prefix separated by a colon to the Attribute. For instance the `osm:` prefix identifies the knowledge coming from OSM.

The information about the world is accessible from the RSG through so called **RSGQueries** (see Figure 3 and Section 4.1). One kind of queries encode Create, Read, Update and Delete (CRUD) operations on the individual primitives of the graph structure, i.e., the nodes and relations. This is comparable to the interfaces of general purpose databases. However, more complex operations require many queries to achieve the desired result. Therefore, the function block query mechanism of the RSG was introduced. A function block can be seen as an active service implementing either aggregated CRUD queries or executing pre-compiled functions to deal with other knowledge sources (as in the DEM example in Section 6). This is comparable to a stored procedure, that is common for databases. They are dynamically loaded into the RSG. Different function blocks for handling different knowledge are developed: the `osmloader` to perform a model-to-model transformation of OSM to RSG, the `demloader` to handle DEM and the `jsongenerator` to store the RSG in JSON files to name just a few.

4.4. KNOWROB and OPENEASE

KNOWROB (Teuforth and Beetz, 2013) was mainly developed for service robots in household to serve as a knowledge representation and processing framework that provides necessary symbolic information for high-level plan executives, e.g. for reasoning about tasks, managing uncertainty, and fast inference. In KNOWROB, the ontologies about household, service tasks and capabilities are stored inside OWL ontologies using RDF triples (Carroll and Stickler, 2004). For querying, KNOWROB has built-in predicates which can be used in SWI-Prolog queries (Vielemeyer et al., 2012).

Figure 4: A human rescuer is querying the environmental information and robot data by using OPENEASE. A simplified query to highlight forests in OPENEASE is illustrated. Those forests are retrieved from OSM data through the RSG. The found forest is highlighted in green.

In this work, we have extended KNOWROB for SHERPA missions in various aspects. Together with the knowledge about the search-and-rescue domain and the geography, these extensions include the knowledge about robot actions, robot components, robot capabilities and, also, reasoning mechanisms for creating hypotheses and inferences based on available knowledge in SHERPA. For instance, we define new robot capabilities such as flying capability, charging ability for quadcopter bases and geographical concepts such as roads, rivers, roofings, and bridges, as well as additional region properties such as the shape or the traversability of paths for specific robots. These extensions allow robots and the human rescuer to infer valuable insights during missions such as which robot is capable of executing a certain task (Vazdarani et al., 2014). For instance, based on the acting robot and its capabilities, KNOWROB can answer the query of if the robot will be able to traverse a specific path.

OPENEASE is a cloud service, which allows humans to interface KNOWROB deployed on a centralized server. It mainly serves as an interface for the human rescuer to check robots' knowledge about their environment and their belief states during the mission. The rescuer can also use this mechanism for getting environmental insights (like seeing areas from a robot's camera), which may influence how he or she manages the on-going operation (see Figure 4).

5. Query-driven integration of Knowledge Bases

5.1. Integration between OpenStreetMap and Robot Scene Graph

As mentioned earlier, we use OSM as one of the source for the geographical knowledge. This knowledge is loaded into the RSG by a model-to-model transformation of the graph primitives. Since both representations are graph-based, OSM nodes can be mapped to RSG Nodes, OSM ways and relations to RSG Connections. Pose information is kept separately in P3G and dedicated connections, called *Transforms*, store the geolocated positions of all OSM objects in the map. This conforms to model to model transformations using so called triple graph grammars Kindler and Wagner (2007); Anjorin et al. (2015). Further details on the model-to-model transformation can be found in Blumenthal et al. (2016). While our implementation supports only the transformation from OSM to RSG, the modelled relations hold for both directions.

The process of converting OSM is encapsulated in a dedicated *RSG function block* called *osmloader*. Since it is a *function block*, it can be triggered by a query to load additional map whenever necessary.

Because OSM is converted to RSG primitives, *RSGQuery*s can be used to retrieve the knowledge in OSM. All queries to RSG are achieved using the query messages encoded with JSON. The RSG distinguishes message types by the *@worldmodeltype* field: *RSGQuery* is used for querying primitives (explained here), while *RSGFunctionBlock* triggers a function block (explained in Section 5.2). The *query* field specifies which query to perform. Queries asking for nodes having specific *Attributes* can be formulated in the *attributes* section based on (POSIX) regular expressions. An example query to retrieve nodes representing a forest (polygon) is shown in Listing 1.

Listing 1: Query to retrieve all forests.

```

1 {
2   "@worldmodeltype": "RSGQuery",
3   "query": "GET_CONNECTIONS",
4   "attributes": [
5     {"key": "osm:landuse", "value": "forest"},
6   ]
7 }
```

The corresponding reply is listed below (cf. Listing 2). All replies have a message type *@worldmodeltype* and a field *querySuccess* indicating if the

query was successful or not. In this example the query is successful and a list of **ids** (only one shown in example) is returned. The RSG uses universally unique identifiers³ (UUIDs) to reference entities in the graph. This property allows to perform subsequent queries by using the ids like primary keys in a database. Also every query has a unique identifier (UUID) which its result refers to. This allows to send queries and process results asynchronously by matching their UUIDs. For brevity UUIDs are not shown in the Listings.

Listing 2: Reply to retrieve all forests.

```

1 {
2   "querySuccess": true,
3   "@worldmodeltype": "RSGQueryResult",
4   "query": "GET_CONNECTIONS",
5   "ids": ["00000000-0000-0000-0000-000013aab0f1"]
6 }
```

5.2. Integration between Digital Elevation Models and Robot Scene Graph

Since OSM does not contain height data for all geographical nodes, DEMs are added as source for height data and are loaded by `demloader` which provides a set of queries that can be triggered with **RSGFunctionBlock** queries. In the RSG function blocks can be dynamically loaded at runtime and can be also subject to change during runtime. Thus, it can not be a priori known what exact input and output a `functionBlock` has. For this reason, the meta model of knowledge required in the respective query is explicitly specified in every message for a function block (see Listing 3). The meta model is a JSON Schema (cf. Galiegue et al. (2013)) that allows to dynamically validate every query before execution.

The `demloader` can handle different queries as specified in its schema file⁴. One of them returns the minimum and maximum values given a specific area. The input is a `Connection` forming a polygon. Since it can be referenced by its UUID it is enough to specify an `areaId` as shown in Listing 3.

The corresponding reply is listed below in Listing 4. The combination of queries to OSM and DEM sources and current locations of team members can be

³<https://www.ietf.org/rfc/rfc4122.txt>

⁴http://github.com/blumenthal/brics_3d_function_blocks/blob/01/models/fbx-demloader-output-schema.json

Listing 3: Query to retrieve elevation for a forest.

```

1 {
2   "@worldmodeltype": "RSGFunctionBlock",
3   "metamodel":       "rsg-functionBlock-schema.json",
4   "name":             "demloader",
5   "operation":        "EXECUTE",
6   "input": {
7     "metamodel": "fbx-demloader-input-schema.json",
8     "command":   "GET_MIN_MAX_ELEVATION",
9     "areaId":    "00000000-0000-0000-0000-000013aab0f1"
10  }
11 }

```

used by KNOWROB to reason, e.g., about traversability of different terrain areas as explained in Section 6. It is also possible to, e.g., query first OSM to get an ID for an area describing a mountain range and then using that to get the height information from DEM.

Listing 4: Reply to retrieve elevation for a forest.

```

1 {
2   "@worldmodeltype": "RSGFunctionBlockResult",
3   "metamodel":       "rsg-functionBlock-schema.json",
4   "operation":        "EXECUTE",
5   "operationSuccess": true,
6   "output": {
7     "metamodel": "fbx-demloader-output-schema.json",
8     "minElevation": 1567.7628173828125,
9     "maxElevation": 1576.4039306640625,
10    "result": "ELEVATION_VALUE_EXISTS"
11  }
12 }

```

5.3. Integration between Robot Scene Graph and KNOWROB

For the integration of KNOWROB and the RSG, **computables** are used, which are procedures of KNOWROB for accessing knowledge from other sources. Sim-

Figure 5: Excerpt of a RSG with OSM data and road segments added by KNOWROB. A road segment list is represented as a RSG Group node with attribute `kb:road = segment_list`. Road segments are modelled as RSG Connections connecting existing OSM nodes denoted by `osm:node_id`. All elements added by KNOWROB are tagged with `kb:iri`, the ids of the instantiated entities of the ontology. The positions of OSM nodes are represented by Transform relations with a `gis:origin` node as coordinate reference frame. Note, the ids are shortened to improve readability.

ilar to the RSG function blocks, KNOWROB computables are accessing knowledge on demand. By implementing computables, developers formally define how knowledge (possibly with different semantics) in other sources can be integrated into the rest of KNOWROB. When a query requires knowledge that can be accessed through the RSG, KNOWROB computables formulate the query into JSON using predefined symbolic rules. These symbolic rules describe how a Prolog query is translated into the JSON-based query schema of the RSG that was described in the previous subsection. Examples of such queries in English would be *getting the pose history of a single robot* or *retrieving all objects in a certain area*.

To make more concrete, a query for retrieving forests inside the Prolog environment is as follows:

Listing 5: Query to get information about the forests.

```
1 ?- findall(ForestPolygon,
2      owl_individual_of(ForestPolygon, knowrob:'Forest')),
   _ForestPolygon).
```

This query requires OSM-related knowledge from the RSG. Thus, KNOWROB translates this query to the JSON query already shown in Listing 1. After receiving the result, these forests can be highlighted in OPENEASE canvas for the human team leader (Figure 4).

In addition, KNOWROB can **add** new data to the RSG using the CRUD queries. One example is to store road segments (cf. Fig. 5) used for reasoning about traversability of terrain areas. This is illustrated in the Examples Section 6. Attributes added to the RSG by KNOWROB are indicated by a `know` prefix.

6. Examples

The agents and human commander can ask queries about agent capabilities (such as which agent can achieve a certain type of action), environmental features (such as how the slopes of a certain region are), and mission (such as which parts of the region have not been traversed yet by any robots). Other queries the system is able to handle and process can be read in (Blumenthal et al., 2016) and (Yazdani et al., 2018) where information about the traversability of regions or the capability of teammates to pass certain regions can be accessed. In order to illustrate how the proposed query-based integration works, this section contains examples that require querying both knowledge bases. For the sake of readability, we have provided examples with either visual or symbolic results. On the other hand, the knowledge base can also answer the queries require pure subsymbolic results (such as what the altitude of a certain tree is).

In the first example a mobile robot needs to reason about the traversability of a road. This requires knowledge about its own motion capabilities and the slopes of the road segments. The capabilities are stored in a KNOWROB ontology. The slopes have to be calculated from map. Then, the robot must check the traversability of the road by comparing its motion capabilities with the slope of each road segment. Figure 6 shows a visualization of the slope calculation.

The KNOWROB computable *path_traversable_by(Robot, Road)* validates whether a road is traversable for a given robot. Particularly, this computable checks whether the robot can pass every road segment by validating its motion capabilities (represented by a maximum slope it can traverse) against the environmental conditions (the calculated slope). The implementation of this computable is discussed in Blumenthal et al. (2016).

The knowledge about roads are coming from the polylines inside OSM. Each node of such a polyline corresponds to an OSM node with its location information. Using these locations, the altitude for each node can be retrieved from the DEM. In

Figure 6: Slope visualization of a road marked by arrows pointing at the slopes. The KNOWROB query that was called is *calculate_slopes(Road)*.

order to reason about the traversability, the polyline of a road needs to be split into smaller road segments represented by polygons. The splitting of OSM roads into segments is done manually using QGIS and the resulting subpolygons are stored in the OSM format. These subpolygons are loaded by the RSG (see Figure 5). After the road segmentation is done, the slope of those roads can be calculated by queries.

The query to retrieve the information about the whole road is written inside the KNOWROB system:

Listing 6: Query to calculate slopes.

```
1 ?- calculate_slopes(Road) :-
2   get_all_segments(Road, Segments),
3   forall(member(X, Segments),
4     get_min_max(X, Min, Max)).
```

The computable *calculate_slopes(Road)* contains two subqueries which require querying the RSG. Moreover, the computable *get_all_segments(Road, Segments)* retrieves all road segments of all roads stored in the RSG as a list by using the following RSG query:

Listing 7: Query to retrieve list of all road segment nodes

```

1 {
2   "@worldmodeltype": "RSGQuery",
3   "query": "GET_NODES",
4   "attributes": [
5     {"key": "kb:road", "value": "segment_list"},
6   ]
7 }

```

The computable *get_all_segments(Road01, RoadSegments)* to retrieve all segments of the of KNOWROB road instance *Road01* can be translated to an RSGQuery using the ID of the KnowRob instance *Road01* (Figure 5):

Listing 8: Query to retrieve node for *Road01*.

```

1 {
2   "@worldmodeltype": "RSGQuery",
3   "query": "GET_NODES",
4   "attributes": [
5     {"key": "kb:iri", "value": "305982"},
6   ]
7 }

```

All child nodes of the returned node are the RSG Connections that represent the road segments belonging to *Road01*. They can be retrieved with the query in Listing 9 by using the additional **subgraphId** field.

The subquery *forall(member(X, Segments), get_min_max(X, Min, Max))* requests the elevation data for each road segment. KNOWROB will first resolve the previously introduced query for all road segments. The resulting *polygonIds* are then used to retrieve the elevation data for each segment from the RSG:

The knowledge about elevation allows KNOWROB to calculate the slope for each road segment. Also, with the slopes it can reason about the traversability of the road for the robot as described above.

7. Discussion

This paper presents an integrated knowledge processing methodology that uses query mechanisms and model-to-model transformations to integrate hetero-

Listing 9: Query to retrieve all road segments of *Road01*

```

1 {
2   "@worldmodeltype": "RSGQuery",
3   "query": "GET_CONNECTIONS",
4   "subgraphId": "c866d82f-137d-4fb1-ba57-f1e344dc73ff"
5   "attributes": [
6     {"key": "kb:road", "value": "segment"},
7   ]
8 }

```

Listing 10: Query to retrieve elevation for a single road segment.

```

1 {
2   "@worldmodeltype": "RSGFunctionBlock",
3   "metamodel": "rsg-functionBlock-schema.json",
4   "name": "demloader",
5   "operation": "EXECUTE",
6   "input": {
7     "metamodel": "fbx-demloader-input-schema.json",
8     "command": "GET_MIN_MAX_ELEVATION",
9     "polygonId": "53edb74c-d277-45c5-9d4a-b4e921e226be"
10  }
11 }

```

geneous knowledge bases and shows its use in the search and rescue domain.

From our efforts, we have learned that there are different ways to integrate heterogeneous knowledge bases and for most complex application a combination of them is needed. In the following we suggest some design guidelines and will discuss them along the dimensions introduced in Figure 2 and along properties of the knowledge bases. Sometimes the guidelines are conflicting for a specific application and a trade-off has to be found by the application developer.

- One of the most important distinctions is between static and dynamic knowledge bases. A passive knowledge base only provides knowledge but does not offer any reasoning or advanced computation methods. An dynamic knowledge base does not just passively provide knowledge but also has functionalities outside of the scope of the integration that are required by the overall system. So if the purpose is to make knowledge from one knowledge

base available to another, but the specific reasoning methods and activities of one knowledge base are not required, a model-to-model transformation is recommended because model-to-model transformations are mostly more efficient and easier to achieve. Then the transformed knowledge can be used in the reasoning methods of the target knowledge base. This requires the analysis of both representations and the creation of a (bidirectional) model-to-model transformation. Such Model-to-model transformations using so called triple graph grammars Kindler and Wagner (2007); Anjorin et al. (2015).

- The integration of dynamic components that have their own specialized reasoning methods and functionalities is more complicated. One possibility is to duplicate all available knowledge by finding transforms and making it available to all components. However, that leads to largely redundant knowledge as well as an overhead to keep dynamically changing knowledge consistent. If the components only use few well defined functionalities of each other and memory is not a concern but communication between the components is, this is a viable option together with one or several bridges that transform information for all components. On the other end of the spectrum is the integration through flexible query interfaces where every operation is a query and no information is duplicated. This implies bidirectional communication and that the query itself has to comply to a metamodel. Unfortunately, most existing systems offer a static API, rather than such a dynamic interaction. This solution can also become inefficient depending on the overhead and the reply times of the queries. However, these two cases are only the extremes of a continuum, which provides opportunities for application dependant trade-offs.
- The update frequency is another important factor. If knowledge is static or rarely changing (as is the case for ontologies), querying is recommended. However, if knowledge is frequently changing (as is the case for the RSG, which is frequently updated with robot data), it is best to keep it in the format in which it is mostly used. That way only the initial knowledge has to be transformed once and the changes can be done in the format in which this information is used. There is also some dependency with the following point.
- The partitioning is another point that influences the choice how to integrate knowledge. For a fully distributed system, communication can become a

defining factor. Synchronising the state of multiple systems and negotiating actions typically require dialogues (mostly by using queries). So using that query infrastructure comes naturally. However, if communication is insecure, most robotic application require the availability of certain knowledge to keep functioning. In that case, the required knowledge needs to be replicated locally on each agent (which may or may not require a model-to-model transform). In a completely centralised system, communication is usually not an issue and the other guidelines are more important for the choice. However, depending on the level of detail one looks at modern systems, there are rarely centralised systems in use anymore, since even standard computation hardware comes with multiple cores, caches etc, but for the application level, this often does not matter.

- At a higher level of abstraction, knowledge bases typically come with reasoning methods to deal with the ontologies and symbolic knowledge of their domain. So the more abstract the content of a knowledge base is, the more likely is a query-based integration. In addition, symbolic/ontological knowledge typically changes less frequently, which also suggests query-based integration. In contrast, metric information is more likely to change frequently and also easier to transform. E.g., sensor data changes frequently but is easy to transform to different representations (like units or different ways of representing rotations).

In the context of the described search and rescue example, this lead to the following design decision. OSM and DEMs are passive knowledge sources in the sense that they do not provide reasoning methods and cannot perform computations. DEMs are treated as immutable and made available to the RSG through an internal query interface of a function block, which accesses the relevant data structures (partial model-to-model transformation). New OSM information is frequently added or modified. Therefore, a model-to-model transformation was chosen to integrate it into the RSG data structures because this allows direct querying and modification of the data with the RSG capabilities.

The RSG KNOWROB with its SHERPA extensions, and OPENEASE are active components, meaning they provide reasoning capabilities and the possibility to perform computations. They were developed as general purpose components. Therefore, they are integrated using a querying interface. A querying interface is more flexible w.r.t. a compiled API but less efficient. Adding new queries can be done at runtime, and unknown queries do not affect the stability of the system.

However, parsing and interpreting queries requires more time than interpreting the fixed data types of a compile time interface.

More generally, query interfaces provide models of what one is interested in while method calls typically provide a model of how something has to be formulated to compute what you are interested in. For method calls a model is interpreted at compile time and fixed into a binary format. This makes it computationally more efficient at run time because the received information does not have to be interpreted and such an interface is easy to use for the given use case. At the other end of the spectrum, query interfaces are more flexible because they exchange models. That brings robustness and flexibility at the cost of less computational efficiency and higher implementation complexity. The robustness comes from the fact that query interfaces can engage in a dialogue but requires the necessary application dependent processing. The flexibility comes from the exchange of models, which requires the means to interpret it, *i.e.*, a so called meta-model or domain specific language. That means that it requires more computations to interpret the model but is also more flexible to changes as long as they conform to the given meta-model. Therefore, such an interface poses less constraints for composing systems because the models and implementation for their interpretation can change and be extended without changing the structural part of the architecture while changes for method calls are limited to the compiled structure or will break the interface. These are again the two extreme cases for a continuum along which an application dependent design decision has to be taken. Flexible query interfaces make integration easier but require effort for implementing an application dependent query processing. However, that also means that the query processing is the only place that requires changing when the query models change.

In practice, most projects require the composition of existing (legacy) systems from various origins. The query-based methodology eases integration of such systems into one platform. DLMs, OSM, the RSG, and KNOWROB were all existing systems with existing interfaces. The already query-based interfaces of the RSG and KNOWROB considerably reduced the integration effort.

8. CONCLUSION

This work introduces an integrated knowledge processing methodology for heterogeneous knowledge bases. It introduces several system requirements common to robotics and discusses design guidelines that help with finding a design trade-off and integration method. In addition, the application of the integrated knowledge processing methodology is explained in detail for a search and rescue

scenario. In this scenario it is shown how mixed human-robotic teams can collaborate by using knowledge bases at different levels of abstraction ranging from symbolic descriptions over topological knowledge, to the grounding in metric data. It also provides an example for knowledge that is distributed over multiple agents and how this influences the integration and reasoning.

One of the key issues is the insufficient modelling of software and data in robotics. Often, there are no formal models at all and if there are models, they are meant for people and not for automated reasoning. There are few common meta-models or schemas that would help with model-to-model transformations, which would help translating the plethora of formats for knowledge and data representations and even for creating bridges to connect legacy software with fixed interfaces. In other communities some solutions are available or under development. A few interesting examples are the linked data community Lanthaler and Gütl (2013) looking for generalized (query) interfaces, the ontology and database communities with their work on database ontology alignment and merging Rahm (2011), or the internet of things community Yang et al. (2015) with their work on service discovery and sharing of heterogeneous data. However, it is difficult to find the relevant knowledge and then most of these solutions are also not interoperable. However, they can serve as inspiration for robotic solutions.

ACKNOWLEDGMENT

This work was supported by the University of Leuven IOF Kennisplatform Transition, and from the European Union's 7th Framework Programme projects SHERPA (FP7-600958), and Pic'nPack (FP7-311987).

BIBLIOGRAPHY

- Anthony Anjorin, Erhan Leblebici, Roland Kluge, Andy Schürr, and Perdita Stevens. A systematic approach and guidelines to developing a triple graph grammar. In *Beyond STAF*, pages 81–95, 2015.
- Timea Băgosi, Joachim de Greeff, Koen V. Hindriks, and Mark A. Neerincx. *Designing a Knowledge Representation Interface for Cognitive Agents*, pages 33–50. Springer International Publishing, Cham, 2015. doi: 10.1007/978-3-319-26184-3_3. URL https://doi.org/10.1007/978-3-319-26184-3_3.

- John Bateman, Michael Beetz, Daniel Beßler, Asil Kaan Bozcuglu, and Mihai Pomarlan. Heterogeneous ontologies and hybrid reasoning for service robotics: The ease framework. In Third Iberian Robotics Conference (ROBOT '17), Sevilla, Spain, 2017.
- Michael Beetz, Moritz Tenorth, and Jan Winkler. Open-FASE: a knowledge processing service for robots and robotics/ai researchers. In IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, USA, 2015. Finalist for the Best Cognitive Robotics Paper Award.
- George A Bekey, Huan Liu, Rajko Tomovic, and Walter J Karplus. Knowledge-based control of grasping in robot hands using heuristics from human motor skills. IEEE Transactions on Robotics and Automation, 9(6):709–722, 1993.
- François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. Journal of Biomedical Informatics, 41(5):706 – 716, 2008. ISSN 1532-0464. doi: <https://doi.org/10.1016/j.jbi.2008.03.004>. URL <http://www.sciencedirect.com/science/article/pii/S1532046408000415>. Semantic Mashup of Biomedical Data.
- Sebastian Blumenthal and Herman Bruyninckx. Towards a domain specific language for a scene graph based robotic world model. CoRR, abs/1408.0200, 2014. URL <http://arxiv.org/abs/1408.0200>.
- Sebastian Blumenthal, Benjamin Brieber, Nico Huebel, Fereshta Yazdani, Michael Beetz, and Herman Bruyninckx. A case study for integrating heterogeneous knowledge bases for outdoor environments. In International Conference on Intelligent Robots and Systems (IROS), Workshop on Integrating Multiple Knowledge Representation and Reasoning Techniques in Robotics (MRRR-16), Daejeon, South Korea, 2016.
- Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, March 2014. URL <https://rfc-editor.org/rfc/rfc7159.txt>.
- Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation, W3C - World Wide Web Consortium, September 2006. URL <http://www.w3.org/TR/2006/REC-xml11-20060816/>.

- Jeremy J Carroll and Patrick Stickler. Rdf triples in xml. In Proceedings of the 13th international World Wide Web conference on Alternative track papers & posters, pages 412–413. ACM, 2004.
- Mehdi M. Dastani, Koen V. Hindriks, Peter Novák, and Nick A. M. Tinneimeier. Combining Multiple Knowledge Representation Technologies into Agent Programming Languages, pages 60–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-93920-7. doi: 10.1007/978-3-540-93920-7_5. URL http://doi.org/10.1007/978-3-540-93920-7_5.
- S.B. Davidson, C. Overton, and P. Buneman. Challenges in integrating biological data sources. In Journal of Computational Biology, 1995. 2(4):557-72.
- Francis Galiegue, Kris Zyp, et al. Json schema: core definitions and terminology. <https://tools.ietf.org/html/draft-zyp-json-schema-04>, 2013. URL <https://tools.ietf.org/html/draft-zyp-json-schema-04>. Accessed: September 2017.
- Frank Hoffmann and Gerd Pfister. Evolutionary design of a fuzzy knowledge base for a mobile robot. International Journal of Approximate Reasoning, 17(4):447–469, 1997.
- Ekkart Kindler and Robert Wagner. Triple graph grammars: Concepts, extensions, implementations, and application scenarios. Technical report, Tech. Rep. TR-ri-07-284, Department of Computer Science, University of Paderborn, Germany, 2007.
- Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.
- Tomáš Krajník, Janne Pulido Fentanes, João Santos, and Tom Duckett. Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. IEEE Transactions on Robotics, 2017.
- Ivana Kruijff, Korbayová, Francis Colas, Mario Gianni, Fiora Pirri, Joachim de Greeff, Koen Hindriks, Mark Neerincx, Petter Ögren, Tomáš Svoboda, and Rainer Worst. Tradr project: Long-term human-robot teaming for robot assisted disaster response. KI - Künstliche Intelligenz, 29(2):193–201, Jun

2015. ISSN 1610-1987. doi: 10.1007/s13218-015-0352-5. URL <https://doi.org/10.1007/s13218-015-0352-5>.
- Markus Lanthaler and Christian Gütl. Model your application domain, not your json structures. In Proceedings of the 22nd International Conference on World Wide Web, pages 1415–1420. ACM, 2013.
- Hongyu Lin, Le Sun, and Xianpei Han. Reasoning with heterogeneous knowledge for commonsense machine comprehension. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2022–2033, 2017.
- Lorenzo Marconi, Claudio Melchiorri, Michael Beetz, Dejan Pangercic, Roland Siegwart, Stefan Leutenegger, Raffaella Carboni, Stefano Stramigioli, Herman Bruyninckx, Patrick Doherty, et al. The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In Safety, Security and Rescue Robotics (SSRR), 2012 IEEE International Symposium on, pages 1–4. IEEE, 2012.
- Nils J Nilsson. Shakey the robot. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1984.
- Daniel Nyga, Mareike Picklman, Sebastian Koralewski, and Michael Beetz. Instruction Completion through Instance-based Learning and Semantic Analogical Reasoning. In International Conference on Robotics and Automation (ICRA), Singapore, 2017.
- RS Patil, RE Fikes, PF Rietel Schneider, D McKay, T Finin, TR Gruber, and R Neches. The clasp knowledge sharing effort: progress report. principles of knowledge representation and reasoning: Proceedings of the third international conference, 1992.
- Erhard Rahm. Towards large-scale schema and ontology matching. In Schema matching and mapping, pages 3–27. Springer, 2011.
- Jennifer Rowley. The wisdom hierarchy: representations of the diKW hierarchy. Journal of Information Science, 33(2):163–180, 2007. doi: 10.1177/0165551506070706. URL <https://doi.org/10.1177/0165551506070706>.
- L. D. Stein. Integrating biological databases, 2003.

- Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *Int. J. Rob. Res.*, 32(3):566–590, April 2013. ISSN 0278-3649. doi: 10.1177/0278364913481635. URL <http://dx.doi.org/10.1177/0278364913481635>.
- Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, JMara Montiel, Alexander Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2): 69–82, 2011.
- Jan Wielemaker, Tom Schrijvers, Markus Triska and Torbjörn Lager. Swi-prolog. *Theory and Practice of Logic Programming*, 12(1):67–96, 2012.
- Zheng Xu, Hui Zhang, Chuanping Hu, Lin Mei, Jinyu Xuan, Kim-Kwang Raymond Choo, Vijayan Sugumaran, and Yiwei Zhu. Building knowledge base of urban emergency events based on cross-collecting of social media. *Concurrency and Computation: Practice and Experience*, 28(15):4038–4052, 2016.
- Fan Yang, Nelson Matthys, Rafael Bachmair, Sam Michiels, Wouter Joosen, and Danny Hughes. μ pn: plug and play peripherals for the internet of things. In *Proceedings of the tenth European conference on computer systems*, page 25. ACM, 2015.
- Fereshta Yazdani, Benjamin Brieber, and Michael Beetz. Cognition-enabled robot control for mixed human-robot rescue teams. In Emanuelle Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *In Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, volume 302, pages 1357–1369, Padova, Italy, July 15–19 2014. Springer. ISBN 978-3-319-08538-4.
- Fereshta Yazdani, Gayane Kazhoyan, Asil Kaan Bozcuoglu, Andrei Haidu, Ferenc Balint-Lencsedi, Daniel Beßler, Mihai Pomarlan, and Michael Beetz. Cognition-enabled framework for mixed human-robot rescue team. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2016. IEEE. accepted for publication.
- Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM, 2016.

Chaim Zins. Conceptual approaches for defining data, information, and knowledge. Journal of the American Society for Information Science and Technology, 58(4):479–493, 2007. ISSN 1532-2890. doi: 10.1002/asi.20508. URL <http://dx.doi.org/10.1002/asi.20508>.

Fereshta Yazdani has graduated from Computer Science Department in University of Bremen, Germany 2012. In 2012, she earned her Dipl.-Inf. in Computer Science in the working group Computer Architecture at University Bremen. Her diploma thesis was about Embedding reversible Systems in SystemC. From 2012 to until 2017, she has been working as a research assistant under the supervision of Prof. Michael Beetz in the Institute for Artificial Intelligence, University of Bremen. She was working in the EU project Sherpa which was about the development of search and rescue mechanisms for a mixed human robot team in alpine environments. Her primary research focus is on human-robot interaction and ensuring natural tasking in human-robot teams.

Sebastian Blumenthal received his B.,Sc. and M.,Sc. degrees in computer science from the University Bonn-Rhein-Sieg in 2007 and 2009. In 2013 he started his Ph.D. degree at the KU Leuven, focusing on the software engineering aspects of world modeling in large-scale robot control systems.

Nico Huebel received his Dipl.-Ing. (M.Sc.) in Engineering Cybernetics from the University of Stuttgart, Germany, in 2010. From 2010 to 2014 he was research assistant at the Institute for Dynamics and Control, ETH Zurich, Switzerland. Since 2014 he is a research Scientist in the Robotics Research Group of KU Leuven. He was a research scholar at Tokyo Institute of Technology and a member of R&D at KUKA Robotics. His research interests are in the area of autonomous robotics, learning, control systems theory, and software engineering for these areas.

Asil Kaan Bozcuoglu has graduated from Computer Engineering Department in Bilkent University, Ankara in 2010. In 2012, he earned his M.Sc. degree in [Computer Engineering](#) from [Middle East Technical University](#) (METU), where he was a member of KCVAN Research Laboratory. He wrote his master thesis on the cognitive development of grasp behavior for iCub, an open-source humanoid robotic platform. From 2012 to until now, he has been working as a research assistant under the supervision of Prof. Michael Beetz in the Institute for Artificial Intelligence, University of Bremen. His primary research focus is on knowledge representation and learning using cloud robotics.

Michael Beetz is a professor for Computer Science at the Faculty for Mathematics & Informatics of the University Bremen and head of the Institute for Artificial Intelligence (IAI). IAI investigates AI-based control methods for robotic agents, with a focus on human-scale everyday manipulation tasks. Michael Beetz received his diploma degree in Computer Science with distinction from the University of Kaiserslautern. His B.Sc., M.Phil., and PhD degrees were awarded by Yale University in 1993, 1994, and 1996 and his Venia Legendi from the University of Bonn in 2000. Michael Beetz was a member of the steering committee of the European network of excellence in AI planning (PLANET) and coordinating the research area "robot planning". He is associate editor of the AI Journal. His research interests include plan-based control of robotic agents, knowledge processing and representation for robots, integrated robot learning, and cognitive perception.

Herman Bruyninckx obtained the Masters degrees in Mathematics (Licentiate, 1984), Computer Science (Burgelijk ingenieur, 1987) and Mechatronics (1988), all from the KU Leuven, Belgium. In 1995 he obtained his Doctoral Degree in Engineering from the same university, with a thesis entitled "Kinematic Models for Robot Compliant Motion with Identification of Uncertainties." He is full-time Professor at the KU Leuven, and held visiting research positions at the Grasp Lab of the University of Pennsylvania, Philadelphia (1996), the Robotics Lab of Stanford University (1999), and the Kungl Tekniska Hogskolan, Stockholm (2002). Since 2014, he has a part-time affiliation with the Eindhoven University of Technology. His current research interests are on-line Bayesian estimation of model uncertainties in sensor-based robot tasks, kinematics and dynamics of robots and humans, and the software engineering of large-scale robot control systems. In 2001, he started the Free Software ("open source") project Orocos <http://www.orocos.org>, to

support his research interests, and to facilitate their industrial exploitation. He participated in about a dozen European research projects on robotics, with a focus in the recent years on the software engineering aspects. In October 2014, he received an honorary doctorate from the University of Southern Denmark, for his leading role in software development and research in robotics.

Fereshta Yazdani

Sebastian Blumenthal

Nico Huebel

Aslı Kaan Bozcuoglu

Michael Beebe

Herman Bruyninckx

Highlights of our paper

Providing human-robot teams heterogeneous knowledge bases, increases the shared awareness in teams to correctly perform tasks and understand each other by grounding high-level knowledge into low-level information and vice versa.

Distributing knowledge bases over different agents extend the capabilities and mission-awareness for robots with different reasoning and inference mechanisms in order to retrieve, process and update knowledge.

When designing a knowledge base, it must be modeled as a distributed system and partition tolerant, e.g. at least a robot's knowledge of its environment remains locally accessible in case of any communication loss.

In order to interpret semantic symbolic high-level instructions, e.g. "go to that tree", advanced semantic reasoning capabilities are required in order to be interpreted as correct task plans with appropriate parameterizations comprehensible by robotic systems.

All the knowledge and the world models need to serve as an episodic memory accessible and processable by different team members at different times.