Noname manuscript No.
(will be inserted by the editor)

Ontological Concepts for Information Sharing in Cloud Robotics

Edison Pignaton de Freitas · Joanna Isabelle Olszewska · Joel Luís Carbonera · Sandro R. Fiorini · Alaa Khamis · S. Veera Ragavan · Marcos E. Barreto · Edson Prestes · Maki K. Habib · Signe Redfield · Abdelghani Chibani · Paulo Goncalves · Julita Bermejo-Alonso · Ricardo Sanz · Elisa Tosello · Alberto Olivares-Alarcos · Andrea Aparecida Konzen · João Quintas · Howard Li

Received: date / Accepted: date

Abstract Recent research and developments in Cloud Robotics (CR) require appropriate knowledge representation to ensure interoperable data, information, and knowledge sharing within cloud infrastructures. As an important branch of the Internet of Things (IoT), these demands to advance it forward motivates academic and industrial sectors to invest on it. The IEEE 'Ontologies for Robotics and Automation' Working Group (ORA WG) has been developing standard ontologies for different robotic domains, including industrial and autonomous robots. The use of such robotic standards has the potential to benefit the Cloud Robotic Community (CRC) as well, supporting the provision of ubiquitous intelligent services by the CR-based systems. This paper explores this potential by developing an ontological approach for effective information sharing in cloud robotics scenarios. It presents an extension to the existing ontological standards to cater for the CR domain. The use of the new ontological elements is illustrated through its use in a couple of CR case studies. To the best of our knowledge, this is the first work ever that implements an ontology comprising concepts and axioms applicable to the CR domain.

Keywords Knowledge-Based Systems · System Interoperability · Automated Collaboration · Cloud Robotics · Multi-Agent Systems · Autonomous Robotics · Robotics and Automation

Edison Pignaton de Freitas
Joel Luís Carbonera
Edson Prestes
Andrea Aparecida Konzen
Federal University of Rio Grande do Sul, Brazil
E-mail: epfreitas@inf.ufrgs.br

Joanna Isabelle Olszewska
University of West of Scotland, UK

Alaa Khamis
University of Waterloo, Canada.

Veera Ragavan
Monash University, Malaysia.

Marcos E. Barreto
Federal University of Bahia, Brazil.

Maki K. Habib
The American University in Cairo, Egypt

Signe Redfield
US Naval Research Laboratory, USA

Abdelghani Chibani
Sandro R. Fiorini
Université Paris-Est Créteil, France

Paulo Goncalves
Instituto Politécnico de Castelo Branco, Portugal

Julita Bermejo-Alonso
Ricardo Sanz
Universidad Politécnica de Madrid, Spain

Elisa Tosello
University of Padova, Italy

Alberto Olivares-Alarcos
Universitat Politécnica de Catalunya, Spain

João Quintas
Instituto Pedro Nunes, Portugal

Howard Li
University of New Brunswick, Canada

1 Introduction

Cloud Robotics (CR) has emerged, in the last decade, as an important research area potentially enhancing the usability and application areas of autonomous robotic systems by integrating reconfigurable, computational and storage resources, applications and services into collaborative, shared cloud systems (Kehoe et al.,

2015a). Within the context of Internet of Things (IoT), it is able to provide scalable ubiquitous intelligence by means of networking robots beyond time and space constraints (Rahimi et al., 2017) combining concepts from robotics, service-oriented architecture (SOA), and cloud computing, leading to important applications in areas such as smart cities, semantic sensor networks (Bruckner et al., 2012), and cloud manufacturing (Lu and Xu, 2019).

In this context, the concept of autonomous robot is understood as a goal-oriented intelligent system that can function, decide, and interact autonomously within a structured or unstructured, static or dynamic and fully or partially observable environments without explicit human guidance (Bayat et al., 2016). As stated in (Fiorini et al., 2017), the robot is considered as a type of agent in the environment under concern. This understanding extends the idea of agents for cloud-based systems, as described in (Sim, 2012), which considers software agents only, and consistent with the abstract concept of agent presented in (Fiorini et al., 2017), thus making the proposed extension coherent for the CR domain.

CR eases the transition from networked robotics to more advanced uses of multi-agents systems (Hu et al., 2012) towards the provision of intelligent services in the IoT context. The rationale behind CR is the possibility to take advantage of the benefits of sharing data and processing resources provided by cloud-based systems in order to enhance and to enlarge the power and the usefulness of robotic systems, according to their specific needs (Quintas et al., 2011). When using the cloud, a robot could improve its capabilities of image understanding, language translation, speech recognition (Zinchenko et al., 2017), path planning, and/or 3D mapping (Guizzo, 2011).

Regarding the sharing of data and services (Xie et al., 2017), standardized concepts are crucial to allow data be easily interchanged among robots, and services be unambiguously accessed within the cloud infrastructure. Standardized concept representation can facilitate robot coordination and collaboration by providing the basis for the deployment of collective intelligence algorithms supported by the cloud. *DavinCi* (Arumugam et al., 2010) and *RoboEarth* (Waibel et al., 2011) are well-known efforts towards the provision of cloud-based services to allow data sharing and management among heterogeneous and autonomous multi-robot systems. World's largest tech giants Amazon, Google and IBM recently released AWS RoboMaker, Google Cloud Robotics Platform and IBM Watson Cloud Robot, respectively, as cloud robotics platforms. Other cloud robotics platforms emerged as spin-offs from research projects, like Rapyuta (a spin-off from RoboEarth and ETH Zurich), or from university projects, like RoboTurk (Stanford University) and RoboBrain (Cornell University). Robot Operating System (ROS) also provides an ecosystem to support cloud robotics. Hence, industrial applications can also benefit from these advances toward Cloud Robotic Systems (CRS), since an autonomous industrial robot can be viewed as a subclass of an autonomous robot (Bayat et al., 2016).

Due to the importance of CR and the inherent need for standards that support its growth, this work aims, on the one hand, to focus on the identification and formalization of some of CR notions that are important for this area. On the other hand, as the proposed set of concepts specifically developed for the CR domain extends the Core Ontology for Robotics and Automation (CORA)ontology (Prestes et al., 2013), it could be potentially included in the standard ontology for autonomous robots under development by the IEEE Autonomous Robotics Working Group (AuR)[1].

The main contributions of this work are: (1) the formalization of additional concepts to the existing standard which could be used for CR as well as support the development of further ones in this domain; and (2) support the effort being done by the IEEE ORA AuR in defining ontological concepts for robotics applications. Additionally, this paper brings a comprehensive overview of how ontologies can be used in the CR domain to enhance machine-to-machine communication, and a couple of useful and comprehensive case-study scenarios using the defined concepts illustrating how practitioners could use ontologies in this domain.

This paper is structured as follows: Section 2 presents the Cloud Robotics landscape and related work, discussing the need for ontologies in that domain. Section 3 brings the ontological basis for the CR. Section 4 presents two CR usage scenarios and Section 5 describes their implementation, based on scenarios that apply the proposed ontological approach to efficiently share information in the considered CR. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Cloud Robotics at a Glance

Cloud Robotics represents a branch of IoT that emerges from joining the Cloud Computing (Mell and Grance, 2011) and Networked Robotics (Wang et al., 2015) paradigms to provide an extended functionality to networked

[1] https://standards.ieee.org/develop/project/1872.2.html

robots and amplify the possibilities of cloud-based systems (Kehoe et al., 2015b).

On one hand, the large storage capacity provided by centralized clouds can help unifying the large volume of information about the environment that robots acquire. It can also provide an extensive library of skills and behaviors robots must comply with to perform specific tasks. This allows robots to learn based on the experience of other robots, leading to effective machine learning (Quintas et al., 2011; Kehoe et al., 2015b).

The understanding of CR includes the definition of its architecture, underlying communication, and extension of elastic benefits of cloud-based systems that it uses (Hu et al., 2012). Typical cloud robotics architectures (Miratabzadeh et al., 2016) are organized into two complementary levels: machine-to-machine (M2M) and machine-to-cloud (M2C). The first level represents a group of robots communicating wirelessly to form a collaborative robotic unit. It provides an increased computing capability and an interoperable exchange of information for collaborative decision making and tasks execution in robot-related applications. The second level centralizes the cloud infrastructure providing a pool of shared computation and storage resources that can be elastically allocated according to current demands.

Cloud Robotics benefits from the concept of elasticity in cloud computing (i.e. the degree to which a system is able to adapt to workload changes by provisioning and re-provisioning resources in an autonomic manner) (Herbst et al., 2013). This concept is realized by enabling the provision of cloud-based resources to networked robots, while extending the cloud by including the network of robots into the pool of shared resources. Exploring elasticity, all resources provided by the cloud and by the robots can be allocated according to current needs. Similarly, the concepts of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) represent elastic resources that robots may benefit from. Besides, the robots themselves seen as resources substantiates the additional concept of Robots as a Service (RaaS) (Kehoe et al., 2015b).

In the context of RaaS, as proposed by several researchers (Quintas et al., 2011, 2017; Bozcuoğlu et al., 2018; Yazdani et al., 2018), robots can be seen as service-providers, transparently sharing their resources with other robots. This resource sharing depends on a match between what is being demanded and what are the robots' capabilities to configure and accomplish a service that meets the demands.

2.2 Ontologies in the Robotics Domain

An ontology defines a set of ontological elements as representational primitives to model a domain of knowledge (Studer et al., 1998)). From this viewpoint, this conceptualization includes the types of entities that are supposed to exist in a given domain, shared by a community of people, i.e., an ontology captures a common understanding about a given domain. Therefore, ontologies can be used to promote *semantic interoperability* among stakeholders, being viewed as task-independent reusable and consensual knowledge.

There have been several projects focusing on the use of ontologies to express the vocabulary and knowledge acquired by robots in specific scenarios, such as bioinformatics (Soldatova et al., 2006), kitting (Balakirsky et al., 2012), rehabilitation (Dogmus et al., 2015), and real-time video tracking (Liu et al., 2014). Other example is an ontology representing the theory of obstacles is described in (Bermejo-Alonso et al., 2010), providing the basis to identify and perform reasoning about potential obstacles in the vehicle environment in order to support navigation. In (Dogmus et al., 2015), a formal ontology is introduced to represent information about rehabilitation robots and their properties. A manipulation framework where physics-based motion planning is enhanced with ontological knowledge representation and reasoning is proposed in (Diab et al., 2019). In (Quintas et al., 2018; Quintas, 2018), a human-machine interaction framework is described as using an ontological knowledge representation to formalize workflow and context information that is used in the decision processes of artificial social companions, including social robots. The work reported in (Okresa Duric et al., 2019) presents MAMbO5, an ontology to model and to manage intelligent virtual multi-agent systems environments. As the robots in CRS can be regarded as agents, this work presents a useful tool for this domain, despite not being specifically designed for networked robotics systems, as CRS are. The lack of specific concepts for CRS justify the works such the one presented in this current paper.

Recent works tackled the need to represent data in robotics, including manipulation and mobile robotics. For example, the definition of the *pose* concept is mandatory and is defined in the IEEE 1872:2015 standard (708, 2015), as well as related concepts like translation, rotation, region and heading in order to share unambiguously knowledge between machines, humans and human-machine. Data representation for robotics does not end with pose definition, and at the same time, concepts for Robot Map Data Representation for Navigation have been defined in the IEEE 1873:2015 standard

(730, 2015). This standard also defines specifications for representing 2D metric and topological maps to be used for exchanging map data among robots, computers, and other devices. The standard defined concepts for static maps, and can be enhanced to deal with moving objects and also with 3D representations of the real-world maps, as seen by robots. On the other hand, sensors and communication between them and robots play a decisive role in any robotic scenario. Several works were carried out to deal with them, using ontologies, in order to enhance the robotic perception. A review is presented in (Schlenoff et al., 2013), where SensorML (Sensor Markup Language[2]) and SSN (Semantic Sensor Network ontology[3]) were discussed as candidate knowledge sources to be included in a Cloud Robotics Ontology.

Previously works not only formalized the concepts of position, mapping, and sensors, but highlighted the importance of using an ontological framework in robotic applications (Olszewska et al., 2017).

Other efforts regarding cloud robotic interoperability exist, such as (Xie et al., 2017), suggesting the use of a semantic model instead of ontologies to represent an information resource service for cloud manufacturing systems. Despite its soundness, a deeper analysis of this approach reveals this model is not as general as the ontological approach proposed in this paper. In this sense, their approach is too bound to the application under concern, while the one proposed here is more general, thus suitable for different types of cloud robotics systems. Another approach named KnowRob (Tenorth and Beetz, 2013), instead, is the most similar solution to the one presented in this current paper, since it is also based on an ontology and MongoDB, as it will be further explained in Section 5. From an implementation perspective, the main difference is that the ontology of KnowRob is based on DOLCE (Masolo et al., 2003), while CORA is based on SUMO (Niles and Pease, 2001). On the other hand, from a conceptual perspective, KnowRob was designed to model robotic manipulation tasks only (Olivares-Alarcos et al., 2019). The ontology proposed in this current paper, instead, aims to be more generic and models an agent (robot) which is acting in a workspace through the set of actions that the agent can perform according to its capabilities.

2.3 The Need for Ontologies in Cloud Robotics

Considering RaaS, the gathered data and produced information by any robot unit can also be shared among others through specific cloud services. This cloud based sharing and services are crucial properties with the increasing demand from government agencies and the private sector alike to use autonomous systems, such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), and autonomous underwater vehicles (AUVs) for homeland security, reconnaissance, search and rescue, surveillance, and urban planning among other tasks (Bozcuoğlu et al., 2018; Yazdani et al., 2018).

In such RaaS scenarios, robots are expected to cooperate and manage different types of data circulating in the cloud (Quintas et al., 2017). Ontologies are important to formally specify the key concepts, properties, relationships, and axioms within this context (Fiorini et al., 2017). Particularly, as an IoT-based system providing services (RaaS), the establishment of a common base of understanding is a key feature for the system to work properly (Gyrard et al., 2018), allowing, for instance, an efficient service discovery (Aziez et al., 2017).

Some benefits of using ontologies include the definition of a standard set of concepts and their meanings, attributes and inter-relations, as well the possibility of knowledge capture and reuse, facilitating systems specification, design and integration (Soldatova et al., 2006). These benefits are also true to CRS. However, its importance is even more stressed in this context due to the heterogeneous and complex environment that CRS represent. Thus, the need for standardization is mandatory to allow these systems to work, and ontologies can make it happen. Once the ontology is defined, the concepts are understood in the same way by the different robots (members) of the CRS. This common base of knowledge enables the robots sharing data that is understandable for all robots, which makes possible not only sharing data about the environment, such an object that is classified as an obstacle, but also data about the capabilities of each robot and how they can help each other in performing tasks in the environment.

3 Ontological Concepts for Cloud Robotics

For robots to work coherently as an associated system, several ontological concepts should exist. Among the set of concepts present in the CR domain, this paper focuses on three: Action, Behavior, and Capability. Subsection 3.1 presents the top-level concepts used for modeling these three new concepts, further defined in subsections 3.2, 3.3 and 3.4.

[2] http://www.sensorml.com/
[3] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

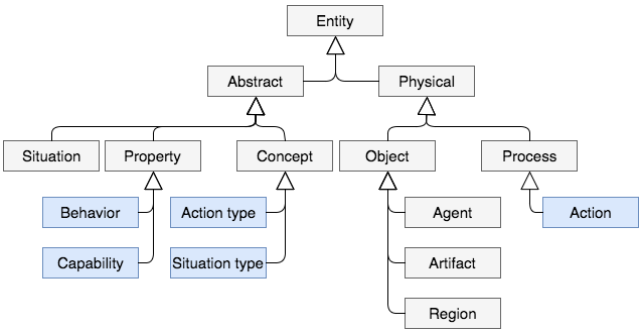

Figure 1: Main ontological concepts proposed (in blue) in relation to existing concepts (in gray).

3.1 Top-level Entities

In this work, a simplified version of SUMO's top-level categories[4] was adopted (Fig. 1), as previous IEEE standard ontology efforts (e.g. CORA) were based on it. The proposed taxonomy first differentiate between *Physical* and *Abstract* entities. Physical entities are those having some physical existence (i.e. some location in time and space), such as rocks, chairs and a soccer match. Abstract entities, on the other hand, do not have physical existence, such as ideas and numbers.

Physical entities are further divided into *Objects* and *Processes*. Objects are wholly present every time (i.e. they do not have temporal parts). Examples include ordinary objects, such as animals, rocks, planets, etc. Processes, on the other hand, happen in time, accumulating temporal parts. Every time a process is happening, only one of its temporal parts is present. Examples include wars, soccer matches, a presentation, an earthquake, etc. Among the objects, two main classes are distinguished: *Agent* and *Artifact*. An agent is an object performing actions by itself, such as animals, humans, robots, etc. Artifacts are objects created by some agent.

Three main abstract entities (*Properties*, *Concepts*, and *Situations*) are defined, loosely influenced by the Descriptions and Situations ontology (Gangemi, 2008). Properties are existentially dependent entities (i.e. they only exist if another entity exists) representing specific qualities of other entities, e.g. weight, length, color, etc. Concepts are abstract entities representing a class of individuals such as the concepts of animal, human and robot. Finally, situations are abstract entities representing the specific way an agent frames a specific fragment of the reality in which it is interested in. For example, the situation in which a robot is moving and collecting images for surveillance purpose to share them within the cloud. The main taxonomy of this proposed ontology[5] can be viewed in Fig. 1.

The following subsections present a formalization and a discussion of the notions *Action*, *Behavior*, and *Capability*. Definitions are expressed using First Order Logics (Barwise, 1977).

3.2 Action

The first ontological concept is Action, which is is intuitively a process initiated by an agent and capable of changing the world. It can be formalized as follows:

Definition 1 An *action* is a process performed by an agent.

$$\forall x\ Action(x) \rightarrow Process(x) \wedge (\exists a\ Agent(a) \wedge performs(a, x)) \quad (1)$$

Actions may be something on or over the environment in which the agent is immersed, such as moving from a location to another; or over another agent, such as sending an information to this other agent. According to Definition 1, each occurrence of an action is existentially dependent on some agent. Thus, in order to some action happen, it is necessary an agent that performs it (1).

In a CR, robots may perform different actions to reply to a given request, such as moving from one place to another, gripping a payload, climbing a wall, providing and requesting data, sending commands. Although there are controversies (Lowe, 2010) regarding if actions can be *intentional* or *non-intentional*, the presented formalization also considers the existence of non-intentional actions. This allows to consider that agents can perform *deliberate* actions (which are always intentional) as well as *reactive* actions (which are non-intentional responses to stimuli). Please note that the formalization of this distinction will be further investigated.

While actions are the very essence of agents, the notion of action is also important for defining the notion of Capability (Definition 2) and Behavior (Definition 3).

3.3 Capability

The second ontological concept is Capability. The notion of capability is related to the potential that an agent (i.e. an active element in the system, such as a robot) has to perform some type of action. It can be formalized as follows:

[4] `http://www.adampease.org/OP/`

[5] The OWL file with the complete ontology can be found at `https://drive.google.com/file/d/1Jx-KYa_1hbYitYmEo4Lu7tV29qOKXHmU/view?usp=sharing`

Definition 2 A *capability* is a property of an agent that allows it to perform or to participate in a certain type of action.

$$\begin{aligned}\forall c\ Capability(c) \rightarrow & Property(c) \wedge \\ & \exists a, t\ Agent(a) \wedge \\ & capabilityOf(c, a) \wedge \\ & ActionType(t) \wedge \\ & allowsPerforming(c, t),\end{aligned} \tag{2}$$

where

$$\begin{aligned}\forall a\ Action(a) \rightarrow & \exists t\ ActionType(t) \wedge \\ & isClassifiedBy(a, t)\end{aligned} \tag{3}$$

and

$$\forall t\ ActionType(t) \rightarrow Concept(t). \tag{4}$$

In (2), 'ActionType' is a *second order class*, whose instances are subclasses of 'Action' (3). 'ActionType' is also considered a subclass of 'Concept' (4), which generalizes other similar notions, such as 'Situation Type'.

Action types have to be considered in the ontology because a given specific capability allows the agent to perform a potentially infinite set of instances of a given action type. Due to this, specific instances of capability are related to instances of action types that, on the other hand, are specific subclasses of Action (and can have infinite instances).

The formalization of the second-order entity 'Action Type' as a class 'ActionType' enables both to reify the second-order entity 'Action Type' and to implement it using languages such as OWL, which are based on first-order logic.

Moreover, it is worth to notice that the relationship 'allowsPerforming' is held between an individual capability and a class of actions (an individual 'Action Type'). On the other hand, a capability is existentially dependent on an agent. Thus, every specific capability is a capability of a given individual agent. If a given agent ceases to exist, its particular capabilities cease to exist as well. Besides that, if an agent performs a given action (5), this means it has the capability to do so:

$$\begin{aligned}\forall a, x\ performs(a, x) \rightarrow & Agent(a) \wedge Action(x) \wedge \\ & (\exists c\ Capability(c) \wedge \\ & capabilityOf(c, a) \wedge \\ & allowsPerforming(c, t) \wedge \\ & ActionType(t) \wedge \\ & isClassifiedBy(x, t)).\end{aligned} \tag{5}$$

An example is the execution of an action to detect obstacles. In this case, the agent needs the capability of object detection.

Capability differs from Purpose. Indeed, a device is created with some explicit purpose and holds this purpose during its whole life cycle, whereas an agent can gain or lose some capability during its life cycle. It is also important to distinguish between 'a specific individual capability' and 'a specific type of capability'. A specific type of capability, like 'jumping', is a subclass of 'Capability', which has several different instances. On the other hand, a specific individual capability, like 'my capability of jumping', is an individual instance of the class 'jumping', which is a specific type (subclass) of capability.

3.4 Behavior

The third ontological concept is Behavior, which can be viewed as patterns of active responses (actions) that an agent exhibits to perceived situations. For example, a robot can have the behavior of 'Avoiding obstacles'. This behavior relates a class of situations 'Facing obstacle' to a class of actions 'Avoiding obstacle', which is triggered by the given situation.

In robotics and automation domain, behavior is directly connected to 'sense, reason and response/act' and 'interaction with the environment'. It can be formalized as follows:

Definition 3 A *behavior* is a property of an agent that makes it perform certain actions when it faces certain situations.

$$\begin{aligned}\forall b\ Behavior(b) \rightarrow & Property(b) \wedge \\ & (\exists a, p, s\ Agent(a) \wedge hasBehavior(a, b) \wedge \\ & ActionType(p) \wedge triggers(b, p) \wedge \\ & SituationType(s) \wedge activates(s, b)),\end{aligned} \tag{6}$$

where

$$\begin{aligned}\forall c\ Situation(c) \rightarrow & \exists t\ SituationType(t) \\ & \wedge isClassifiedBy(c, t),\end{aligned} \tag{7}$$

and

$$\forall t\ SituationType(t) \rightarrow Concept(t). \tag{8}$$

In (6), 'SituationType' is also a *second order class*, whose instances are specific subclasses of 'Situation' (7). A 'SituationType' is also a 'Concept' (8). The notion of 'SituationType' in the ontology follows the same reasoning as the one for 'ActionType' (2). To posses a behavior b, an agent a should have the capabilities of

performing the kinds of actions triggered by the behavior b:

$$\begin{aligned}\forall a, b\ hasBehavior(a, b) \rightarrow \\ (\forall t\ ActionType(t) \wedge triggers(b, t) \rightarrow \\ (\exists c\ Capability(c) \wedge hasCapability(a, c) \wedge \\ allowsPerforming(c, t))).\end{aligned} \tag{9}$$

Next, these ontological concepts presented in this section have been implemented and used in real-world CR system as explained in Section 4.

4 Cloud Robotics Scenario

4.1 Cloud Robotic Context

The proposed case studies focus on the resolution of Multi-Robot Task Allocation (MRTA) problems. These problems clearly emphasize how robotic systems can benefit from both a well-defined ontology (Bruckner et al., 2012) and a cloud-based task allocation framework to transfer the overall computation. Given a mission and a set of robots, MRTA aims to determine which robots should execute which tasks in order to achieve the overall system goal.

This problem can be seen as an optimal assignment problem, where the objective is to optimally assign a set of robots to a set of tasks while optimizing the overall system performance subject to a set of constraints (Khamis et al., 2015). This decision is complex as it usually comes to heterogeneous unreliable robots equipped with different capabilities that should be optimally matched with different tasks requirements. The complexity increases with addition of constraints restricting the space of feasible solutions. For example, there may exist proximity constraints specifying that a task must be performed within a range such as in less (or greater) than a certain distance.

In this context, robots can internally share data through some specific policy and technology, while forming a multi-agent system when considering the whole scenario. They can form a Robotic Cloud (Mell and Grance, 2011) to share and consume data about the environment, to use resources, to perform computationally intensive tasks, as well as to offer their resources to other members within the cloud. As different task scenarios can share the same Robotic Cloud, it is possible to state that this is a CR system. This being a heterogeneous system composed of different types of robots with different capabilities, sensors, and actuators, semantic coherent definitions of involved concepts should be provided (Olszewska, 2017). The semantic coherence of these definitions allows for sharing data representing a common knowledge of robotic experiences. In this way, required tasks can be assigned to any robot with ability to perform them, regardless of its type.

Besides data sharing, the cloud can solve the allocation problem by setting the tasks to be performed and deciding which robot will perform which task (Mell and Grance, 2011; Bruckner et al., 2012). The data sharing process starts by the definition of the ontology that will compose the common knowledge base. Once this knowledge base is set, it is disseminated among the robots in the CRS, which make use of its concepts to communicate with the other robots in the system. This communication is wirelessly done in one-hop or multi-hop. Updates in the knowledge base are shared among all robots in the system, for instance, the discovery of a new obstacle in the environment that is informed by one robot and informed to all other robots.

4.2 Example of Cloud Robotics Usage Scenarios

As concrete examples of this overall CR system architecture, two scenarios are proposed (Tosello et al., 2018a,b). The first scenario (Fig. 2) transports a payload consisting of a set of pieces from a starting position (top of a table) to a goal position (docking station far from the table). Two robots participate in the mission: one manipulator robot, equipped with a camera and a gripper, and one mobile robot, with laser scans. The former is fixed in front of the table and it can detect and manipulate objects, but it cannot navigate towards the goal region. The latter is not able to manipulate, but it can move within the environment while avoiding obstacles. Moreover, a container is located on its top, giving the mobile robot the capability of transporting objects from a start pose to a goal.

Mission accomplishment requires that sub-tasks be divided among the two robots according to their capabilities: the mobile robot should dock near the manipulator robot and the manipulator should pick the pieces and place them on the top of the mobile robot. Finally, the mobile robot should navigate to the goal station. After deciding on robot-task allocation, a task planner should compute the set of actions and the motion each robot has to perform. For example, the motions that the manipulator robot needs are: to approach the manipulation object, grasp it, and place it on the top of the mobile robot.

Computing and processing this data require computationally intensive algorithms such as Simultaneous Location and Mapping (SLAM), sampling-based planning and adaptive planning under uncertainty. Exploring cloud computing facilities would be much faster

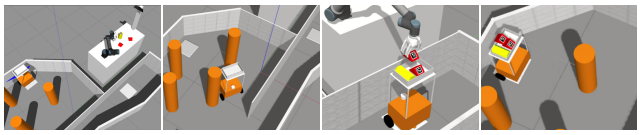

Figure 2: Scenario 1 - A mobile robot navigates towards a docking station next to a manipulator robot, which picks up some pieces from a table and places them on the mobile robot. The mobile robot comes back.

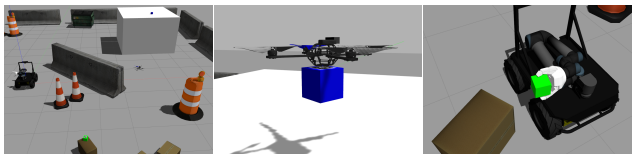

Figure 3: Scenario 2 - A UGV and a UAV picking up a green and a blue cube, respectively.

than relying on robots' on-board computers, as each robot could request the cloud to allocate tasks, recognize objects and compute the task and motion plan.

As mentioned in Section 4.1, besides taking into account robots capabilities, task allocators must also consider other constraints, such as the robot reachability. Moreover, multiple robots of different types can populate the scene and both the scene and the mission can change. The proposed CR system architecture should be able to address these challenges.

Thus, a second scenario is proposed (Fig. 3): two robots (UAV and UGV) populate the environment and the mission is to collect two cubes (green and blue). Both robots have the required capabilities: the UAV is able to fly, manipulate (it has a gripper attached), and perceive its surroundings, while the UGV is able to navigate, manipulate, perceive and scan. However, as seen from the figure, the reachability area of the UGV does not allow it to manipulate the blue cube, hence performs tasks involving green cube only.

Even in this situation, transferring the computation to the cloud could minimize the workload. Moreover, if all robots linked to the CR system share the same ontological concepts and vocabulary, the computed data and information can be stored in the cloud. Once a task is assigned, instead of computing a new allocation and a new plan, the information can be retrieved in the cloud and matched with the assignment. The overall computation is minimized and the disclosure of information speeds up.

5 Cloud Robotic Ontological Implementation

The proposed ontology can be used to represent and communicate behavioral triggers among agents (as described in Section 4.2). Basically, it is possible to represent which robot capabilities are required to allocate a task, as well which behavior (set of actions) should be activated when performing a task under a certain situation.

The implementation described here uses an ontology together with a set of databases (MongoDB) to support the allocator. The ontology stores data that should be easily modified and integrated, such as qualitative and quantitative information about robots, performed tasks, detected objects hypotheses, and explored environments. MongoDB stores data that needs integrity, such as objects' visual features, poses, robot description files and capabilities messages. MongoDB was chosen because it is a NoSQL database, thus providing better performance and flexibility. Among NoSQL databases, MongoDB is the most popular, the fastest-growing, and it has a rich document-oriented structure and deployed both as a fully managed cloud service and on self-managed infrastructure. Its flexible data model can be improved as the requirements change while data collections are made out of individual documents, which can be nested in complex hierarchies while remaining easy to query and index (Davoudian et al., 2018)..

Furthermore, a reasoner is connected to the ontology and both the set of databases and the ontology (collectively referred as 'Knowledge Base'[6] or 'KB' from now on) are located in the cloud. When a set of robots connects to the cloud and a mission is assigned, the reasoner solves the MRTA problem while exploring the information stored in the KB. Moreover, transferring the computation and the knowledge to the cloud means that robots are no longer involved in decision making, thus reducing the amount of local resources needed. If an agent fails, the others are still working on their own. If the failed agent sends a failure notice to the cloud, it can reassign the tasks and/or ask for cooperation among the remaining robots. In the same way, new agents can be added as soon as they are available for accomplishing the mission.

The implemented case study uses a private cloud to save both the knowledge base and the cloud engine. The former stores the information; the latter retrieves and generates data. The tests are performed in simulation: as the implementation is based on the Robot Operating System (ROS) (Quigley et al., 2015), Gazebo (gaz,

[6] The complete implementation of the knowledge base and the case studies can be found at `https://github.com/CloudRobotics-TAMP/RTASK-KB.git`

2014) is used as a simulator. Every simulated agent sends an HTTP request to the Cloud asking for the initialization of a WebSocket connection to the engine. In detail, the agent sends an identification message to the Cloud and waits for the acknowledgement that the connection is set up. After the connection has been initialized, the agent can start querying the knowledge base by sending JSON WebSocket messages containing the desired SPARQL queries. The engine starts retrieving the requested data and, if no data is available, it tries to generate data and stores it in the knowledge base. The server sends back the result of the query via a JSON message and, when the information exchange is finished, an HTTP request closes the connection.

Once a scenario is initialized, the configuration of the robots and their surroundings is sent to the cloud. This allows the reasoner to understand the situation in which robots are operating, and scene descriptions to be stored in the KB for reuse. Taking into consideration the second scenario, the scene can be formally represented by the instances:

```
UGV(ugv1)
OpenArea(scenario2)
MovingAction(ugv1NavigatingThroughScenario2)
Performs(ugv1, ugv1NavigatingThroughScenario2)

UAV(uav1)
OpenArea(scenario2)
MovingAction(uav1FlyingThroughScenario2)
Performs(ugv1, uav1FlyingThroughScenario2)

Situation(situation1)
IsSettingFor(situation1, ugv1)
IsSettingFor(situation1, scenario2)
IsSettingFor(situation1,
         ugv1NavigatingThroughScenario2)

Situation(situation2)
IsSettingFor(situation2, uav1)
IsSettingFor(situation2, scenario2)
IsSettingFor(situation2,
         uav1FlyingThroughScenario2)
```

The first two sets of instances detail two specific robots (*ugv1* and *uav1*), their operating area, actions, and relationship between the robots and the actions. The other two sets of instances collect specific relevant parts of reality such as the robots, their workspaces, and actions it performs. Every robot of the scene sends its description (name and its kinematic structure) to the cloud. So, the reasoner can identify which robots are available to perform the mission. Robot descriptions and capabilities can also be stored in the KB. For example, capabilities of the UAV of the second scenario are explicitly represented as follows:

```
UAV(uav1)
Capability(uav1FlightCapability)
Capability(uav1ManipulationCapability)
Capability(uav1VisionCapability)
ActionType(typeFlyAction)
ActionType(typeVacuumGraspAction)
ActionType(typeSeeAction)

AllowsPerforming(uav1FlightCapability,
         typeFlyAction)
AllowsPerforming(uav1ManipulationCapability,
        typeVacuumGraspAction)
AllowsPerfotming(uav1VisionCapability,
         typeSeeAction)
CapabilityOf(uav1FlightCapability, uav1)
CapabilityOf(uav1ManipulationCapability, uav1)
CapabilityOf(uav1VisionCapability, uav1)
```

Capability was defined as a property allowing an agent to perform an action. For the second scenario, the definition specifies that *uav1* is provided with a set of capabilities such as *fly, manipulate,* and visually *perceive* the environment.

Once information on the robots, their capabilities, and their surroundings are stored in the cloud, a user can assign a mission to the cloud. A *mission* M is a set of sub-tasks $\{T_1, T_2, ..., T_n\}$ where every sub-task T_i is characterized by a certain domain D_i. D_i defines the discrete actions the robot can take to solve T_i, including their preconditions and effects. For example, the action 'pick-up' may have the precondition that the object x is *ontable* and the effect that the robot's hand *holds* the object x. The actions, preconditions and the effects in the present case study are listed in Table 1. At the same time, a set of requirements $R_i = \{R_1, R_2, ..., R_m\}$ is associated to every T_i, where requirements are the capabilities that a robot must have to accomplish task T_i. For example, in order to perform a manipulation task with a pick-up action, the robot requires a gripper capable of manipulating.

Mission, domains and requirements are sent to the cloud and stored in the KB. If a manipulation of an object is required, features of the manipulation object are sent too. Features include the geometric and visual description of the object, its mass, its affordance (e.g. *a cup is pickable*), and its utility (e.g. *a cup is useful for drinking*). The reasoner matches the capabilities of the robots with the requirements of the sub-tasks. Once a first list of robots is assigned to every sub-task, a second check is performed against additional constraints such as the reachability area of the chosen robots. In the end, every sub-task is assigned to one of the robots connected to the cloud that gave its availability in accomplishing the mission.

Formally, when the robot starts executing a sub-task, a certain behavior is activated. This behavior depends on the robot's situation. In the case of the UAV of the second scenario, after flying over the blue cube, it can start the manipulation of that object:

```
Behavior(manipulationBehavior)
HasBehavior(uav1, manipulationBehavior)
```

Assuming that the manipulation behavior triggers a specific action type, this fact can be expressed as:

```
triggers(manipulationBehavior,
         typeVacuumGraspAction)
```

Action	Preconditions	Effects
PickUp(?x)	(and (clear ?x) (ontable ?x) (handempty))	(and (not(ontable ?x)) (not(clear ?x)) (not (handempty)) (holding ?x)))
Place(?x)	(holding ?x)	(and (not(holding ?x)) (clear ?x) (handempty) (ontable ?x)))
MoveArm(?from ?to)	(and (pose ?from) (pose ?to) (isAt ?from))	(and (isAt ?to) (not (isAt ?from)))
Navigate(?from ?to)	(and (pose ?from) (pose ?to) (isAt ?from))	(and (isAt ?to) (not (isAt ?from)))

Table 1: Set of primitive actions considered in the ontological implementation of Scenario-2.

It is worth noting that *typeVacuumGraspAction* is linked with the UAV (*uav1*) and its capability (*uav1ManipulationCapability*) to the previously defined properties for the *uav1* using 'ActionType'. During execution, the system checks whether the behavior is achievable by looking if the capability linked to the action type is active or not. The scene of the first scenario can be represented in a similar manner:

```
Manipulator(man1)
OpenArea(scenario1)
MovingAction(man1PickingInScenario1)
Performs(man1, man1PickingInScenario1)

MobileRobot(mob1)
OpenArea(scenario1)
MovingAction(mob1NavigatingInScenario1)
Performs(mob1, mob1NavigatingInScenario1)

Situation(sit3)
IsSettingFor(sit3, man1)
IsSettingFor(sit3, scenario1)
IsSettingFor(sit3, man1PickingInScenario1)

Situation(sit4)
IsSettingFor(sit4, mob1)
IsSettingFor(sit4, scenario1)
IsSettingFor(sit4, mob1NavigatingInScenario1)
```

In this scene, there is a manipulator robot able to perceive visual information from its workspace and to manipulate objects through a 3-finger gripper mounted on its end-effector:

```
Manipulation(man1)
Capability(man1ManipulationCapability)
Capability(man1VisionCapability)
ActionType(typeFingerGraspAction)
ActionType(typeSeeAction)

AllowsPerforming(man1ManipulationCapability,
        typeFingerGraspAction)
AllowsPerforming(man1VisionCapability,
         typeSeeAction)
CapabilityOf(man1ManipulationCapability, man1)
CapabilityOf(man1VisionCapability, man1)
```

Once the object to be picked has been detected, the manipulator robot can start the manipulation routine:

```
Behavior(manipulationBehavior)
HasBehavior(man1,
        manipulationBehavior)
```

This manipulation behavior triggers the grasp of the object through the 3-finger gripper:

```
triggers(manipulationBehavior,
         typeFingerGraspAction)
```

It is important to remember that all high-level information is stored in the cloud. This means that if a robot connects to the cloud and is characterized by the same kinematic properties of a robot seen in the past, its capabilities has already been stored in the KB and will not have to be processed again. The same is true for a task that has already been assigned and an object that has already been seen or manipulated. If the stored information is incomplete, it can be integrated with the new information while preserving ontological consistency.

6 Conclusions

The convergence of cloud computing and autonomous robot systems has brought to the fore an IoT emerging technology named Cloud Robotics. The key feature of CR is related to resource and data sharing among robots through the cloud, as well as the sharing of the robots themselves as resources within a CR system. Such systems present a clear need for well-defined standards to make possible a semantic, coherent, data sharing among the robots and service provision. Hence, according to the goals of the IEEE ORA Working Group, CR is one of the domain areas that need a standard set of terms and definitions for consistent and coherent knowledge sharing.

This paper contributes to the efforts of ORA-WG IEEE 1872.2 standard, by presenting the definition of the ontological concepts of Action, Behavior, and Capability pivotal to the CR domain. For the benefit of practitioners, using typical CR system scenarios, two case studies are provided which help to verify the concepts, relations, and vocabulary defined in this paper. The case studies highlight the main contributions of this work by prototyping the first ontology for cloud robotics which seeks to enhance the interoperability in these systems. As Cloud Robotics is a very new and complex domain, there are many directions for future works which demonstrate the potential of the proposed ontological approach. A possible future work directly related to what is presented in this paper is the definition of other concepts useful to the CR domain. Another

important direction of future investigations is the use of Machine Learning methods to handle new acquired skills that the robots may share knowledge of within the cloud. Interface with other IoT systems is also an important issue to be addressed.

Acknowledgment

This work was partly supported by Innovate UK; CNPq and CAPES in Brazil; FRGS/1/2015/TK08/MUSM/02/1 in Malaysia.

References

(2014) Gazebo tutorials. http://gazebosim.org/tutorials/, [Online; accessed 22-November-2019]

(2015) IEEE Standard for Robot Map Data Representation for Navigation. IEEE Std 1873-2015 pp 1–54

(2015) IEEE Standard Ontologies for Robotics and Automation. IEEE Std 1872-2015 pp 1–60

Arumugam R, Enti VR, Bingbing L, Xiaojun W, Baskaran K, Kong FF, Kumar AS, Meng KD, Kit GW (2010) Davinci: A cloud computing framework for service robots. In: IEEE International Conference on Robotics and Automation (ICRA), pp 3084–3089

Aziez M, Benharzallah S, Bennoui H (2017) An ontology based context model for the discovery of iot services in the internet of things. In: 2017 International Conference on Mathematics and Information Technology (ICMIT), pp 209–213, DOI 10.1109/MATHIT.2017.8259719

Balakirsky S, Kootbally Z, Schlenoff C, Kramer T, Gupta S (2012) An industrial robotic knowledge representation for kit building applications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1365–1370

Barwise J (1977) An introduction to first-order logic. Studies in Logic and the Foundations of Mathematics 90:5–46

Bayat B, Bermejo-Alonso J, Carbonera J, Facchinetti T, Fiorini S, Goncalves P, Jorge VA, Habib M, Khamis A, Melo K, Nguyen B, Olszewska JI, Paull L, Prestes E, Ragavan V, Saeedi S, Sanz R, Seto M, Spencer B, Vosughi A, Li H (2016) Requirements for building an ontology for autonomous robots. Industrial Robot: An International Journal 43(5):469–480

Bermejo-Alonso J, Sanz R, Rodriguez M, Hernandez C (2010) Ontology-based engineering of autonomous systems. In: 2010 Sixth International Conference on Autonomic and Autonomous Systems, pp 47–51

Bozcuoğlu AK, Kazhoyan G, Furuta Y, Stelter S, Beetz M, Okada K, Inaba M (2018) The exchange of knowledge using cloud robotics. IEEE Robotics and Automation Letters 3(2):1072–1079

Bruckner D, Picus C, Velik R, Herzner W, Zucker G (2012) Hierarchical semantic processing architecture for smart sensors in surveillance networks. IEEE Transactions on Industrial Informatics 8(2):291–301

Davoudian A, Chen L, Liu M (2018) A survey on nosql stores. ACM Comput Surv 51(2), DOI 10.1145/3158661, URL https://doi.org/10.1145/3158661

Diab M, Akbari A, Ud Din M, Rosell J (2019) Pmk—a knowledge processing framework for autonomous robotics perception and manipulation. Sensors 19(5):1166

Dogmus Z, Erdem E, Patoglu V (2015) Rehabroboonto: Design, development and maintenance of a rehabilitation robotics ontology on the cloud. Robotics and Computer-Integrated Manufacturing 33:100–109

Fiorini SR, Bermejo-Alonso J, Goncalves P, Pignaton de Freitas E, Olivares Alarcos A, Olszewska JI, Prestes E, Schlenoff C, Ragavan SV, Redfield S, Spencer B, Li H (2017) A suite of ontologies for robotics and automation. IEEE Robotics and Automation Magazine 24(1):8–11

Gangemi A (2008) Norms and plans as unification criteria for social collectives. Autonomous Agents and Multi-Agent Systems 17(1):70–112

Guizzo E (2011) Robots with their heads in the clouds. IEEE Spectrum 48(3):16–18

Gyrard A, Zimmermann A, Sheth A (2018) Building iot-based applications for smart cities: How can ontology catalogs help? IEEE Internet of Things Journal 5(5):3978–3990, DOI 10.1109/JIOT.2018.2854278

Herbst N, Kounev S, Reussner R (2013) Elasticity in cloud computing: What it is, and what it is not. In: International Conference on Autonomic Computing, pp 23–27

Hu G, Tay WP, Wen Y (2012) Cloud robotics: Architecture, Challenges and Applications. IEEE Network 26(3):21–28

Kehoe B, Patil S, Abbeel P, Goldberg K (2015a) A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering 12(2):398–409

Kehoe B, Patil S, Abbeel P, Goldberg K (2015b) A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering 12(2):398–409

Khamis A, Hussein A, Elmogy A (2015) Multi-robot task allocation: A review of the state of the art. In: Cooperative Robots and Sensor Networks, Springer, pp 31–51

Liu B, Chen Y, Blasch E, Pham K, Shen D, Chen G (2014) A Holistic Cloud-Enabled Robotics System

for Real-Time Video Tracking Application, Springer Berlin Heidelberg, pp 455–468

Lowe E (2010) Action theory and ontology. A Companion to the Philosophy of Action 662:662

Lu Y, Xu X (2019) Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. Robotics and Computer-Integrated Manufacturing 57:92–102

Masolo C, Borgo S, Gangemi A, Guarino N, Oltramari A (2003) Wonderweb deliverable d18: Ontology library. Technical report, Laboratory for Applied Ontology -ISTC-CNR

Mell P, Grance T (2011) The NIST definition of cloud computing. NIST Special Publication 800-145

Miratabzadeh SA, Gallardo N, Gamez N, Haradi K, Puthussery AR, Rad P, Jamshidi M (2016) Cloud robotics: A software architecture. In: IEEE World Automation Congress, pp 1–6

Niles I, Pease A (2001) Towards a standard upper ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001, Association for Computing Machinery, New York, NY, USA, FOIS '01, p 2–9, DOI 10.1145/505168.505170, URL https://doi.org/10.1145/505168.505170

Okresa Duric B, Rincon J, Carrascosa C, Schatten M, Julian V (2019) Mambo5: a new ontology approach for modelling and managing intelligent virtual environments based on multi-agent systems. Journal of Ambient Intelligence and Humanized Computing 10(9):3629–3641, DOI 10.1007/s12652-018-1089-4, URL https://doi.org/10.1007/s12652-018-1089-4

Olivares-Alarcos A, Bessler D, Khamis A, Goncalves P, Habib MK, Bermejo-Alonso J, Barreto M, Diab M, Rosell J, Quintas J, Olszewska JI, Nakawala H, Pignaton E, Gyrard A, Borgo S, Alenya G, Beetz M, Li H (2019) A review and comparison of ontology-based approaches to robot autonomy. Knowledge Engineering Review 34:1–38

Olszewska J (2017) Clock-model-assisted agent's spatial navigation. In: Proceedings of the International Conference on Agents and Artificial Intelligence, vol 2, pp 687–692

Olszewska JI, Barreto M, Bermejo-Alonso J, Carbonera J, Chibani A, Fiorini S, Goncalves P, Habib M, Khamis A, Olivares A, Freitas EP, Prestes E, Ragavan SV, Redfield S, Sanz R, Spencer B, Li H (2017) Ontology for autonomous robotics. In: IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp 189–194

Prestes E, Carbonera JL, Fiorini SR, Jorge VAM, Abel M, Madhavan R, Locoro A, Goncalves P, Barreto ME, Habib M, Chibani A, Gérard S, Amirat Y, Schlenoff C (2013) Towards a core ontology for robotics and automation. Robotics and Autonomous Systems 61(11):1193–1204

Quigley M, Gerkey B, Smart W (2015) Programming Robots with ROS: A Practical Introduction to the Robot Operating System. O'Reilly Media, URL https://books.google.com.br/books?id=Hnz5CgAAQBAJ

Quintas J, Menezes P, Dias J (2017) Interoperability in cloud robotics—developing and matching knowledge information models for heterogenous multi-robot systems. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), IEEE, pp 1291–1296

Quintas J, Martins GS, Santos L, Menezes P, Dias J (2018) Toward a context-aware human–robot interaction framework based on cognitive development. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49(1):227–237

Quintas JM, Menezes PJ, Dias JM (2011) Cloud Robotics: Toward Context Aware Robotic Networks. In: Biomechanics/752:Robotics, Actapress

Quintas JML (2018) Context-based human-machine interaction framework for arti ficial social companions. PhD thesis, 00500:: Universidade de Coimbra

Rahimi R, Shao C, Veeraraghavan M, Fumagalli A, Nicho J, Meyer J, Edwards S, Flannigan C, Evans P (2017) An industrial robotics application with cloud computing and high-speed networking. In: IEEE International Conference on Robotic Computing, pp 44–51

Schlenoff C, Hong T, Liu C, Eastman R, Foufou S (2013) A literature review of sensor ontologies for manufacturing applications. In: IEEE International Symposium on Robotic and Sensors Environments, pp 96–101

Sim KM (2012) Agent-based cloud computing. IEEE Transactions on Services Computing 5(4):564–577, DOI 10.1109/TSC.2011.52

Soldatova LN, Clare A, Sparkes A, King RD (2006) An ontology for a robot scientist. Bioinformatics 22(14):e464–e471

Studer R, Benjamins VR, Fensel D (1998) Knowledge engineering: principles and methods. Data and Knowledge Engineering 25(1-2):161–197

Tenorth M, Beetz M (2013) Knowrob: A knowledge processing infrastructure for cognition-enabled robots. The International Journal of Robotics Research 32(5):566–590, DOI 10.1177/0278364913481635, URL https://doi.org/10.1177/0278364913481635, https://doi.org/10.1177/0278364913481635

Tosello E, Castaman N, Michieletto S, Menegatti E (2018a) Teaching robot programming for industry 4.0. In: International Conference of Educational Robotics (EduRobotics)

Tosello E, Tagliapietra L, Fan Z, Gatto AC, Pagello E (2018b) Rtask: A cloud-based knwoledge engine for robot task and motion planning. In: Fields of Robotics, Journal of.

Waibel M, Beetz M, Civera J, D'Andrea R, Elfring J, Gálvez-López D, Häussermann K, Janssen R, Montiel JMM, Perzylo A, Schießle B, Tenorth M, Zweigle O, Molengraft RVD (2011) Roboearth. IEEE Robotics Automation Magazine 18(2):69–82

Wang S, Krishnamachari B, Ayanian N (2015) The optimism principle: A unified framework for optimal robotic network deployment in an unknown obstructed environment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 2578–2584

Xie C, Cai H, Xu L, Jiang L, Bu F (2017) Linked semantic model for information resource service towards cloud manufacturing. IEEE Transactions on Industrial Informatics 13(6):3338–3349

Yazdani F, Kazhoyan G, Bozcuoglu AK, Haidu A, Balint-Benczedi F, Beßler D, Pomarlan M, Beetz M (2018) Cognition-enabled framework for mixed human-robot rescue team. In: International Conference on Intelligent Robots and Systems (IROS), IEEE, Madrid, Spain

Zinchenko K, Wu CY, Song KT (2017) A study on speech recognition control for a surgical robot. IEEE Transactions on Industrial Informatics 13(2):607–615