Cloud-Based Task Planning for Smart Robots

Elisa Tosello, Zhengjie Fan, Alejandro Gatto Castro and Enrico Pagello

Abstract This paper proposes an Open Semantic Framework for knowledge acquisition of cognitive robots performing manipulation tasks. It integrates a Cloud-based Engine, which extracts discriminative features from the objects and generates their manipulation actions, and an Ontology, where the Engine saves data for future accesses. The Engine offloads robots by transferring computation on the Cloud. The Ontology favors knowledge sharing among manipulator robots by defining a common manipulation vocabulary. It extends the work proposed by the IEEE RAS Ontology for Robotics and Automation Working Group by covering the manipulation task domain. During ontological data insertion, data duplication is avoided by providing a novel efficient interlinking algorithm. During their retrieval, visual data processing is optimized by using a cascade hashing algorithm that intelligently accesses data. No training is required for object recognition and manipulation because of the adoption of a human-robot cooperation. The framework is based on the open-source Robot Operating System.

Keywords Cognitive robotics · Task planning · Cloud-based object manipulation engine · Ontology · Core ontology · ROS

E. Tosello (✉) · A.G. Castro · E. Pagello (✉)
Intelligent Autonomous Systems Lab (IAS-Lab) Department of Information
Engineering (DEI), University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy
e-mail: toselloe@dei.unipd.it

A.G. Castro
e-mail: alejandro.gattocastro@studenti.unipd.it

E. Pagello
e-mail: epv@dei.unipd.it; zepv@dei.unipd.it

Z. Fan
Big Data Team Department of Big Data and IT Technology,
China Mobile Research Institute (CMRI), Innovation Building, No. 32, Xuanwumen West
Avenue, 100053 Beijing, Xicheng District, People's Republic of China
e-mail: fanzhengjie@chinamobile.com

W. Chen et al. (eds.), *Intelligent Autonomous Systems 14*,
Advances in Intelligent Systems and Computing 531,
DOI 10.1007/978-3-319-48036-7_21

1 Introduction

Robots of the future should be Smart: versatile and efficient Artificial Intelligence systems able to perform behaviors of growing complexity, adapt to changes, collaborate with humans and other robots, learn from the past and from action performed by other agents, and build on their capabilities based on that knowledge. Two are the reasons that limit robots intelligence: the limited capacity of on board data elaboration and the absence of a common medium to communicate and share knowledge. Tapping into the Cloud is the solution. A Cloud server allows to offload robots from CPU-heavy tasks and to perform intensive computation while meeting the hard real-time constraints of operations. A Web Ontology lets the definition of a common vocabulary that ensures a common understanding during the interaction as well as an efficient data transfer and integration.

In automation, a large amount of objects should be real-time manipulated. Failures can led to high costs. Moving data and computation to the Cloud favors data sharing [1], enables robots learning the stability of finger contacts from previous manipulations on the same object [2], and lets the application of strategies used on some objects on similar parts encountered later [3].

This paper proposes an Open Semantic Framework for knowledge acquisition of cognitive robots performing manipulation tasks. An Ontology and a Cloud-based Engine have been implemented. The former stores data about objects and actions necessary to manipulate them. The latter detects objects in the scene and retrieves their manipulation actions from the Ontology. If no ontological data exists, the Engine generates and stores it on the Ontology.

The rest of the paper is organized as follows. Section 2 details researches done in this context and compares existing works with the one proposed by this paper. Section 3 describes the Ontology design and its implementation details. Section 4 shows the Cloud-based Engine focusing on the intelligent ontological data insertion and retrieval. It also highlights how the elimination of the training phase for object recognition and manipulation and its substitution with a human-robot cooperation makes gradual the robots knowledge growth. Section 5 proves the good performances of the proposed system by discussing the experiments done. Section 6 presents conclusions and future work.

2 Related Work

Different Cloud Platforms exist. Their inadequacy for robotics scenario is mainly due to the difference in Web applications and robotics applications. Many Web applications are stateless, single processes that use a request-response model to talk to the client. Robotic applications are state-full, multi-processed, and require a bidirectional communication with the client. An example of efficient and widespread Cloud Platform which, however, is not suited for robotics applications, is the Google App

Engine.[1] It exposes only a limited subset of program APIs tailored specifically for Web applications, allows only a single process, and does not expose sockets, which are indispensable for robotic middlewares such as the open-source Robot Operating System (ROS) [4]. In order to overcome these limitations, some Cloud Robotics Platforms have been implemented. An example is Rapyuta, the RoboEarth Cloud Engine [5], a platform designed for robots to share data and action experiences with each other. With respect to Rapyuta, the proposed Engine focuses only on the robotics sharing of manipulation data and actions, but guarantees an efficient Cloud data access by adopting a cascade hashing algorithm [6]. Moreover, it avoids data duplication during the insertion of new data by using a novel powerful interlinking algorithm [7]. The algorithm finds the interlinking pattern of two data sets by applying two machine learning methods: the K-medoids [8] and the Version Space [9].

Focusing on the Knowledge Base to which the Engine accesses, many existing works are available online. Examples are the Columbia Grasp dataset [10], the KIT object dataset [11], and the Willow Garage Household Objects Database [12]. KnowRob [13], the knowledge base of RoboEarth [14], is the most widespread. They stores information about objects in the environment and their grasp poses. The Household object database is a *simple* SQL database: the SQL format does not favor robotics knowledge scalability. The others are well-defined by an Ontology. RobotEarth models objects as 3D colored Point Clouds [15], the others store objects as triangular meshes. Stored items are of high quality but each object model consists of several recordings from different point of views; thus requiring either a lot of manual work or expensive scanning equipments.

With respect to the existing Knowledge Bases, the one proposed, named RTASK, is scalable because of the adoption of an Ontology that defines data. It guarantees an intelligent data storage and access because of the type of data saved. Every object is characterized by multiple visual features (2D Images, B-Splines, and Point Clouds). When detecting an object, the recognition process starts the comparison of the smallest features (e.g. the ones representing the 2D Images), and eventually expands to the others (B-Splines and Point Clouds, in increasing order). No onerous manual work is required to store objects from different view points: an object is stored even if there exists only a single registration of one its views. A human teacher helps robots in recognizing objects when viewed from other orientations. The teacher exploits the connection between the new view point and other object properties, e.g., name and function. These new features will be stored in the Ontology gradually incrementing robots knowledge about the object itself. Moreover, the proposed Ontology observes the IEEE standards proposed by the IEEE Robotics and Automation Society (RAS)'s Ontology for Robotics and Automation (ORA) Working Group (WG) by extending the Knowledge Base it proposed (see Fig. 1) [16–18]. Respecting the standard, Balakirsky et al. [19] proposed a kitting ontology. RTASK generalizes the manipulation concept by introducing the notions of manipulation *task* and *action*. This means that any manipulation task can be represented, e.g., grasps and pushes.

[1] Google, Inc. "Google App Engine". Online: https://cloud.google.com/appengine/ (2014).

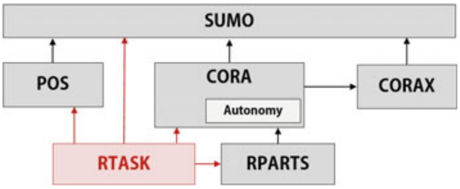


Fig. 1 The RTASK extension to the IEEE ontology for robotics and automation

3 The Ontology

RTASK formulates a common vocabulary for robotics manipulation. As proposed in [20], it separates the concepts of *tasks* and *tasks executions*. Tasks are *abstract* entities that describe goals to be reached; while tasks executions are *events* composed of *actions* that are performed by robots in order to reach goals.

3.1 Design

Figure 2 depicts the Ontology design: a **Task** is assigned to an **Agent**, e.g., a **Robot**. It should be executed within a certain **time** interval and requires the fulfillment of a certain **Motion** in order to be performed. **Manipulation** is a sub-class of **Task**. Several types of manipulations exist, e.g., grasps and pushes. They involve the handling of an **Object** located at a certain **Pose** (**Position** and **Orientation**) through the exectution of a **Manipulating** action. If the **Task** is assigned to a **Robot**, then the **Motion** will be represented by a **Robot Action**. In detail, the **Robot Manipulation Action** involves the activation of the robot **End Effector**. Studies have demonstrated that: (i) placing the arm in front of the object before acting improves actions; (ii) humans typically simplify the manipulation tasks by selecting one of only a few different prehensile postures based on the object geometry [21]. On this view, the **End Effector** is first placed at a **Pose** *p'* at a distance *d* from the **Object**, with its actuable **Joints** at a certain **Pre** manipulation **Posture**. Then, the **End Effector** is placed at a **Pose** *p* close to the **Object** and the effective manipulation **Posture** is assigned to its **Joints**.

In order to retrieve the manipulation data of an object in the scene, the object should be recognized as an instance previously stored in the Ontology. For this purpose, every **Object** is characterized by an *id*, *name*, *function*, and the visual features obtained by the **Sensors**. For every **Object**, RTASK stores multiple types of visual features: **2D Images**, **B-Splines**, and **Point Clouds**.

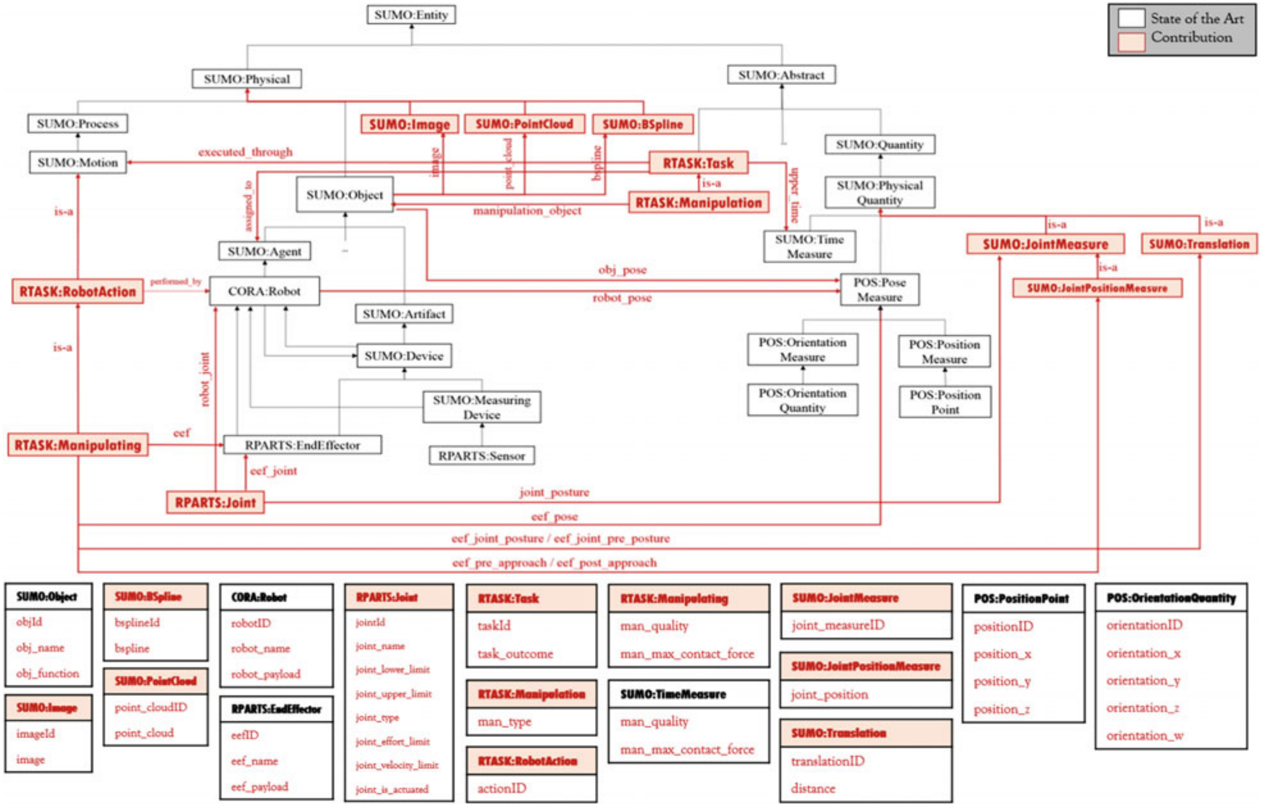


Fig. 2 RTASK: The ontology

3.2 Implementation Details

RTASK is represented through the union of the Resource Description Format (RDF)[2] and the Web Ontology Language (OWL),[3] namely OWL Full. RDF is used to define the structure of the data, OWL adds semantic to the schema and allows the user to specify relationships among data. OWL Full allows an ontology to augment the meaning of the RDF vocabulary guaranteeing the maximum expressiveness of OWL and the syntactic freedom of RDF. Indeed, OWL is adopted by the World Wide Web Consortium (W3C)[4] and it is the representation language used by the RAS ORA WG. Protégé is used as ontology editor.[5]

Queries allow robots to investigate the knowledge base and retrieve existing data. A robot able to query the database has the capability of efficiently and intelligently perform tasks. In our case, a C++ interface lets ROS users query RTASK using SPARQL.[6] Apache Jena Fuseki is used as SPARQL server.[7]

4 The Cloud-Based Engine

The current implementation of the Cloud-based Engine is based on a Cloud-based Object Recognition Engine for robotics (CORE) [22]. The robot has an internal *ROS node* that receives the segmented objects (objects are segmented using the functions offered by the Point Cloud Library [23]) and sends them to the Cloud-based Engine. The Engine is composed on another *ROS node* capable of reading the content of received messages. The communication is based on the **ros_bridge** interface, which provides a web socket channel between nodes.

4.1 Data Retrieval

4.1.1 Objects Manipulation Data Request

The robot asks the Cloud Server for the retrieval of the manipulation data of an object. It sends a message containing the type of its gripper and the compressed Point Cloud of the manipulable object. The compressed Point Cloud representation saves space and connection time. The compressed Point Cloud is encoded in order to

[2] Resource Description Format (RDF). Online: http://www.w3.org/RDF.

[3] Web Ontology Language (OWL). Online: http://www.w3g/TR/owl-features.

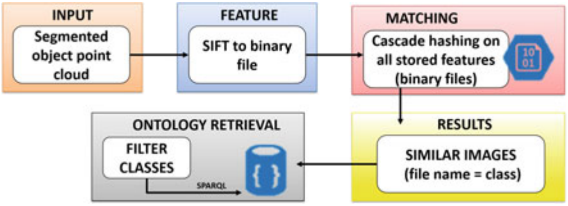
[4] World Wide Web Consortium (W3C). Online: http://www.w3c.com.

[5] Protégé. Online: http://protege.stanford.edu/.

[6] Simple Protocol and RDF Query Language (SPARQL). Online: http://www.w3.org/TR/sparql11-query.

[7] Apache Jena Fuseki. Online: https://jena.apache.org/documentation/fuseki2/index.html.

Fig. 3 Image processing pipeline of the matching phase


INPUT
Segmented object point cloud
FEATURE
SIFT to binary file
MATCHING
Cascade hashing on all stored features (binary files)
10
01
RESULTS
SIMILAR IMAGES (file name = class)
ONTOLOGY RETRIEVAL
FILTER CLASSES
SPARQL
{ }

be transmitted over the Web Socket channel. After the encoding, the whole message is represented as a JavaScript Object Notation (JSON) object. The ROS message being encoded on the Web Socket request follows.

```
# Task name (pick, place for example)
string task
# Gripper name
string gripper_id
# Compressed Point Cloud (object representation)
string data
```

The Server receives the Client request and performs a super-fast search of the object inside RTASK. The search starts from the comparison of the object 2D Images features (SIFT, Scale Invariant Feature Transform) [24] and, in case of mismatch, ends with the comparison of its Point Clouds features (HOG, Histogram of Oriented Gradients) [25]. Steps of the super-fast search of SIFT features follow (see Fig. 3).

1. Decode the message and decompress the Point Cloud;
2. Convert the Point Cloud to color image using the Open-source Computer Vision (OpenCV) [26] (OpenCV) functions;
3. Extract the image SIFT features[8] and store them on a binary file of a Server folder;
4. Match the features with the ones stored in the Server;
5. SPARQL query RTASK;
6. The Server returns a *moveit_msgs::Grasp* ROS message containing the relative manipulation data.

The search is fast because of the novel and super-fast Cascade hashing algorithm adopted during the matching phase (Step 4) [6] and because of the way in which features are stored. The algorithm allows constructing a dataset without a learning phase: there is not need to train the hashing function as in other Approximate Nearest Neighbor (ANN) methods. Given the input image, the function returns the names of its most similar images features (according to the SIFT parameters), the names are in the form of integer numbers. The same names define the object classes of RTASK. Moreover, features of stored objects are precalculated and stored on a Server folder: they are not calculated at every data set access. Thanks to the combination of these

[8] Chris Sweeney, Theia Multiview Geometry Library: Tutorial & Reference. Online: http://theia-sfm.org.

Algorithm 1 Interlinking Instances across Data Sets

```
    Input: Two Data Sets
    Output: Links accross Data Sets
 1: The data set $D$, $D'$; /*two data sets to be interlinked*/
 2: Similarity threshold $T$
 3: for Each property/relation in the data set $D$ do
 4:     for Each property/relation in the data set $D'$ do
 5:         Match properties/relations that are corresponding to each other and store as the alignment $A$
 6:     end for
 7: end for
 8: for Each instance in the data set $D$ do
 9:     for Each instance in the data set $D'$ do
10:         Compare instances' property values according to the correspondences of the alignment $A$;
11:         Aggregate all similarities between property values as a similarity value $v$
12:         if $v >= T$ then
13:             The two compared instances are interlinked with owl:sameAs.
14:         end if
15:     end for
16: end for
```

two characteristics only one query at the end of the matching process is needed in order to retrieve the similar objects stored on the Ontology.

4.2 Data Insertion

An example of Client message aiming to create a new Object instance follows. It is encoded as JSON string

```
{"op": "send_new_object",
"service": "insert",
"new_object":
 [
 {"new_point_cloud": "zc5p81H1cO8P+Ksmfdf...",
 "X": "0.5", "Y": "2.5", "Z": "1.5", "rad": "0.314"},
 ]
}
```

When the Server receives the message, it calculates the relative SIFT features and stores them on the features folder and on RTASK.

During the insertion of elements in RTASK, duplication avoidance is desirable. To this end, Algorithm 1 is applied to automate the interlinking process. It was proposed in [7] and finds out the interlink pattern of two data sets by applying two machine learning methods: the K-medoids and the Version Space. Although interlinking algorithms require interactions with users for the sake of the interlinking precision, computations of comparing instances are largely reduced than manually interlinking. As the work-flow of Algorithm 1 shows, when interlinking two instances across two data sets D and D', the algorithm first computes property/relation correspondences across two data sets (line 5). Then, instances property values are compared by referring to the correspondences (line 10). A similarity value v is generated upon all similarities of property values (line 11). If such a similarity is equal to or larger than a predefined threshold *T*, the two compared instances can be used to build a link with the relation *owl:sameAs* (line 12–14).

4.3 Human-Robot Interaction

Usually, an onerous a-priori human manual work is required to store objects visual features on a Robotics Knowledge Base: every object is represented by a large amount of registrations from different points of view. Objects representation is accurate but the a priori work is onerous. Our approach eliminates this prerequisite by introducing a human teacher that supports robots during their learning phase. First registrations will not be accurate, but knowledge will gradually increase until becoming absolute and giving robots autonomy.

Figure 4 highlights the reasoning at the bottom of this cooperation approach. Robots require human intervention to confirm the identity of a recognized object or assign one to a new object. Moreover, humans help robots in connecting visual features of objects already stored in the Knowledge Base but seen from other points of view. To perform the connection, other objects properties are exploited, e.g., name and function.

Many advantages are introduced by this approach. For example, it eliminates the a priori work currently done by humans by introducing a cognitive and social robot able to interact with other agents, e.g., human operators. A key outcome follows: A robot capable of learning is a flexible, adaptable, and scalable cyber-physical system.

The message used for human-robot interaction follows.

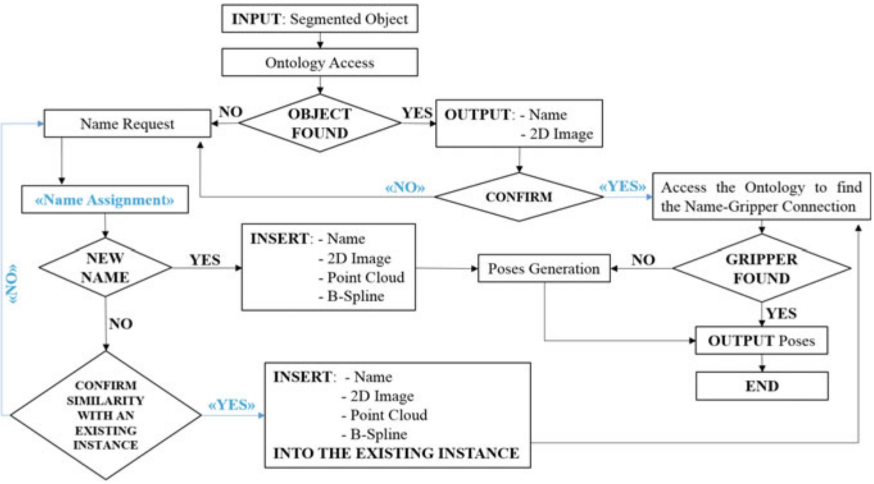


Fig. 4 The reasoning pipeline. The *blue* parts represent human interventions

```
# Operation type
string operation
# Object class name
string class_description
# Compressed Point Cloud (object representation)
string data
```

The human operator shows an object to the robot and gives it a description (e.g., coke, can, pen). The Server seeks for a similar object by filtering the Ontology through the object description. If matches exist, the Server returns a message containing the classes of the similar objects found:

```
{ "op": "search_object_response",
  "service": "query",
 "similar_objects":
 [
  {"class":"1", "X":"0.5","Y":"2.5","Z":"1.5","rad":"0.314"},
  {"class":"2", "X":"0.1","Y":"24.5","Z":"1.3","rad":"0.24"},
  {"class":"3", "X":"0.2","Y":"3.5","Z":"0.5","rad":"2.14"}
 ]
 }
```

a human feedback confirms the object class and visual features are added to it. Otherwise, a new instance is inserted in RTASK:

```
{ "op": "send_new_object",
"service": "insert",
"new_object":
  [
  {"new_point_cloud": "zc5p81H1cO8P+Ksmfdf...",
  "X": "0.5", "Y": "2.5", "Z": "1.5", "rad": "0.314"},
  ]
}
```

4.4 Manipulation Data Generation

Given new object, the Cloud-based manipulation planner generates a list of possible manipulations, each consisting of a gripper pose relative to the object itself. Manipulations consist of both grasps and pushes. The current version of the generator aligns the hand with the object principal axes, starting from either the top or the side of the object, and tries to manipulate it around its Center Of Mass through a trial-and-error Reinforcement Learning technique. As for GraspIt! [27] and the MoveIt! Simple Grasps tool developed by Dave T. Coleman,[9] given the safety distance *d* at which the gripper must be positioned before making the manipulation, the generator returns:

[9] MoveIt! Simple Grasps tool. Online: https://github.com/davetcoleman/moveit_simple_grasps.

- a pre-manipulation configuration: the gripper pose and joints configuration at the safety distance;
- a manipulation configuration: the gripper pose and joints configuration to be maintained during the manipulation.

Experiments proposed by Dave T. Coleman demonstrate that the grasping tool he developed does not fail because, in the scene

- the block to be grasped is known a priori;
- the configuration that the gripper has to maintain during the grasp is manually predetermined according to the block's dimensions;
- collision checking is not performed to verify the feasibility of grasps.

If we reason about arbitrary shapes, collisions or contact losses can be induced by pre-selecting the manipulation configurations. To overcome the problem, the generator used by the Engine exploits the Reinforcement Learning benefits and generates grasps and pushes according to the input object.

5 Experiments

Experiments aim to provide truthful temporal results on Cloud access and ontological data retrieval. For this purpose, RTASK and the Engine have been integrated inside the Cloud environment of CORE, which is available on the Wisconsin CloudLab cluster[10] under the project "core-robotics". It consists of one x86 node running Ubuntu 14.04 with ROS Indigo installed. Moreover, RTASK has been integrated with the Object Segmentation Database (OSD)[11]: a data set of 726 MB currently containing 111 different objects, all characterized by a 2D color Image and a Point Cloud.

In simulation, an Husky mobile robot equipped with a Universal Robot UR5 manipulator and a Robotiq 2 finger gripper has to solve a tabletop object manipulation problem: it has to grasp the nearest object located on the table in front of it. A Microsoft Kinect acquires the scene. Gazebo [28] is used as simulator. A ROS Sense-Model-Act framework has been implemented in order to give the robot the ability to detect objects on the table, access the Cloud server, and retrieve manipulation data (see Fig. 5). Implementation details follow.

Sense The Kinect acquires the RGB-D images of the environment. From the collected data, a compressed segmented 3D Point Cloud of each individual manipulable object is computed using PCL.

Model On the Cloud, from the compressed Point Cloud, the relative 2D images is computed together with its B-Spline [29] representation. The relative visual features

[10] CloudLab. Online: http://www.cloudlab.us.

[11] Object Segmentation Database. Online: http://www.acin.tuwien.ac.at/forschung/v4r/software-tools/osd/.

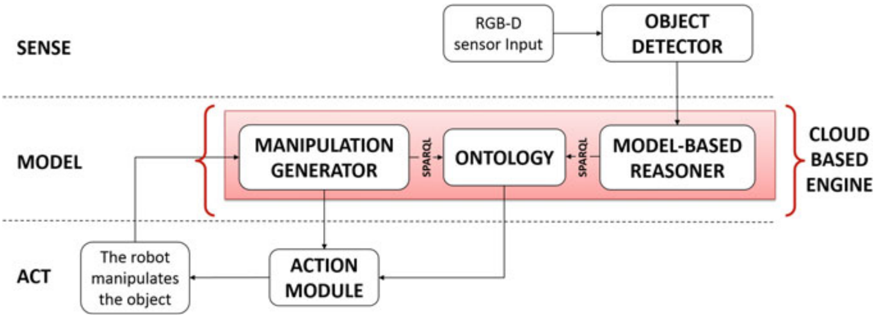


Fig. 5 The sense-model-act framework

are computed, e.g., SIFTs and HOGs. The cascade hashing algorithm starts searching for a match between the features of the segmented object and that saved in the data set. The comparison starts from SIFTs and eventually expands to B-Splines (by computing the squared distance among points) and HOGs. The reasoner accesses RTASK in order to retrieve the manipulation actions relative to the detected features. Again, if a match exists, together with the information relative to the assigned manipulation action (e.g., *push* or *grasp*) and to the relative gripper joints configuration, then the information will be outputted. Otherwise, the Manipulation Data Generator computes the necessary poses.

Act The module lets the robot move by activating its simulated engines. MoveIt![12] generates the kinematic information required for the system to pass from the current to the goal configuration. During the motion, the information acquired by the sensors is used to compare the system final state with the expected one. In case of mismatch, a trial and error routine starts correcting the joints configuration. The configuration that allows the task achievement is saved in RTASK.

5.1 Results

Figure 6 shows the system in operation: the robot detects the surrounding environment, segments the objects in front of it, and grasps the nearest objects through the manipulation configuration retrieved from the Cloud.

Tables 1 and 2 reports the most significant time data. Table 1 depicts the time taken for the extraction of the descriptors of the 2D images stored in the data set. The table shows two types of extraction: the extraction of descriptors of all the 111 images contained in the data set and that of descriptors of a single image. For each type of extraction, tests were done on 1, 2, 3, and 4 threads respectively. Authors point out the 0.387422 s used to extract descriptors of a single image through the

[12] Ioan A. Sucan and Sachin Chitta, “MoveIt!”, Online: http://moveit.ros.org.

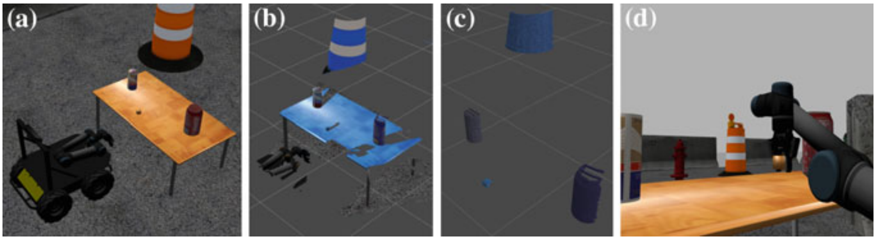


Fig. 6 **a** The robot in the scene; **b** The detected scene; **c** The segmented objects; **d** The grasp of the nearest object

Table 1 Elaboration time taken for features extraction

Features extraction				
	1 Thread (s)	2 Thread	3 Thread	4 Thread (s)
111 images	45.1554	21.8381 s	15.331 s	012.7545
1 image	0.379014	0.408295	0.396394	0.387422

Table 2 Elaboration time taken for features matching

Features matching		
	Time (s)	Matches on 6105 possible image pairs
15 inliers	4.82	1592
300 inliers	4.48500	18

employment of 4 threads. Table 2 focuses on times taken for features matching. By using 300 inliers, the match is accurate and the computational time does not affect the real-time constraints of a robotics manipulation.

The reported computational times consider SIFT features. Times gradually increments if B-Splines or HOGs are considered. Moreover, the computational effort depends on the richness of the features: the more complex the Point Cloud is, the greater the extraction time will be. The segmentation helps maintaining computational times low.

During the experiments, MoveIt! took on average 3.765 s to find a feasible inverse kinematics solution (best case: 0.162757 s; worst case: 7.367439 s; number of trials: 100) on a Dell Intel Core i7-4470 CPU @ 3.40 GHz x 8, 15.6 GiB Memory, 970 GB Disk. In the best case, it completed the planning after 0.8486 s. Reported data proves that the intelligent and efficient structure of the proposed Open Semantic Framework does not adversely affect the time required to complete a manipulation: executing the features extraction and matching on 4 threads off-loads robots and increases system performances.

6 Conclusion and Future Work

This paper presented an Open Semantic Framework able to increase robots knowledge and capabilities on objects manipulation. The Framework is composed of an OWL Ontology and a Cloud-based Engine. From the study of human actions when handling objects, the Ontology formulates a common vocabulary that encodes the robotics manipulation domain. The Engine, instead, was developed in order to transfer the computation on the Cloud: it off-loads robot CPUs and speeds up the robots learning phase. Given an object in the scene, the Engine retrieves its visual features and accesses the Ontology in order to extract the corresponding manipulation action. If no information is stored, a Reinforcement Learning technique is used to generate the gripper manipulation poses that will be stored on the Ontology. The Ontology respects the IEEE Standard by extending the existing CORA. The Engine minimizes visual data processing through an intelligent ontological data access and retrieval. During ontological data retrieval, a cascade hashing algorithm is adopted in order to optimize the comparison between saved and new visual features. Instead, in order to avoid data duplication during the insertion of new instances in the Ontology, a novel efficient interlinking algorithm has been adopted. Furthermore, the training for objects recognition and manipulation is replaced by a human-robot interaction.

We proved the efficiency and effectiveness of the proposed approach by building a ROS Sense-Model-Act framework able to associate manipulation actions to the features of the objects in the scene. Tests were performed in simulation.

As future work, we aim to extend the proposed Ontology by defining other tasks and actions, e.g., we would like to explore the Navigation domain. We aim to assign robots the new tasks and execute the relative new actions in order to increase the capabilities of the Cloud-based Engine. Moreover, we are developing a new Reinforcement Learning technique for the generation of the manipulation configurations.

References

1. Goldfeder, C., Allen, P.K.: Data-driven grasping. Auton. Robot. **31**(1), 1–20 (2011)
2. Dang, H., Allen, P.K.: Learning grasp stability. In: Proceedings of International Conference on Robotics and Automation (ICRA), pp. 2392–2397. Saint Paul, MN (2012)
3. Glover, J., Rus, D., Roy, N.: Probabilistic models of object geometry for grasp planning. In: Proceedings of Robotics Science and Systems (RSS), Zurich, Switzerland (2008)
4. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
5. Mohanarajah, G., Hunziker, D., D'Andrea, R., Waibel, M.: Rapyuta: a cloud robotics platform. IEEE Trans. Autom. Sci. Eng. **12**(2), 481–493 (2015)
6. Cheng, J., Leng, C., Wu, J., Cui, H., Lu, H.: Fast and accurate image matching with cascade hashing for 3d reconstruction. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
7. Fan, Z.: Concise Pattern Learning for RDF Data Sets Interlinking. PhD thesis, University of Grenoble (2014)

8. Kaufman, L., Rousseeuw, P.J.: Statistical Data Analysis Based on the L1-Norm and Related Methods, chapter Clustering by means of Medoids. North-Holland, pp. 405–416 (1987)
9. Dubois, V., Quafafou, M.: Concept learning with approximation: rough version spaces. In: Rough Sets and Current Trends in Computing: Proceedings of the Third International Conference (RSCTC), Malvern, PA, USA, pp. 239–246 (2002)
10. Goldfeder, C., Ciocarlie, M., Allen, P.: The Columbia grasp dataset. In: Proceedings of International Conference on Robotics Automation (ICRA), Kobe, Japan, pp. 1710–1716 (2009)
11. Kasper, A., Xue, Z., Dillman, R.: The KIT object models database: an object model database for object recognition, localization and manipulation in service robotics. Int. J. Robot. Res. (IJRR) **31**(8), 927–934 (2012)
12. Ciocarlie, M., Hsiao, K., Jones, E., Chitta, S., Rusu, R., Sucan, I.: Towards reliable grasping and manipulation in household environements. In: Proceedings of International Symposium on Experimental Robotics, Delhi, India, pp. 1–12 (2010)
13. Tenorth, M., Beetz, M.: KnowRob: a knowledge processing infrastructure for cognition-enabled robots. Int. J. Robot. Res. (IJRR) **23**(5), 566–590 (2013)
14. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: RoboEarth—a world wide web for robots. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011)
15. Di Marco, D., Koch, A., Zweigle, O., Haussermann, K., Schiessle, B., Levi, P., Galvez-Lopez, D., Riazuelo, L., Civera, J., Montiel, J.M.M., Tenorth,M., Perzylo, A., Waibel, M., Van de Molengraft, R.: Creating and using RoboEarth object models. IEEE Int. Conf. Robot. Autom. (ICRA) 3549–3550 (2012)
16. Prestes, E., Fiorini, S.R., Carbonera, J.: Core Ontology for Robotics and Automation. Standardized Knowledge Representation and Ontologies for Robotics and Automation. Workshop on the 18th, pp. 7–9. Illinois, USA, Chicago (2014)
17. Schlenoff, C., Prestes, E., Madhavan, R., Goncalves, P., Li, H., Balakirsky, S., Kramer, T., Miguelanez, E.: An IEEE standard ontology for robotics and automation. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1337–1342. Vilamoura, Algarve, Portugal (2012)
18. Schlenoff, C.I.: An overview of the IEEE ontology for robotics and automation (ORA) standardization effort. In: Workshop on the 18th, Standardized Knowledge Representation and Ontologies for Robotics and Automation, pp. 1–2. Illinois, USA, Chicago (2014)
19. Balakirsky, S., Kootbally, Z., T. RKramer, A. Pietromartire, C. Schlenoff, Gupta, S.: Knowledge driven robotics for kitting applications. Robot. Auton. Syst. **61**(11), 1205–1214 (2013)
20. Fiorini, S.R., Carbonera, J.L., Gonçalves, P., Jorge, V.A., Rey, V.F., Haidegger, T., Abel, M., Redfield, S.A., Balakirsky, S., Ragavan, V., Li, H., Schlenoff, C., Prestes, E.: Extensions to the core ontology for robotics and automation. Robot. Comput.-Integr. Manuf. 33(C), 3–11 (2015)
21. Cutkosky, M.R.: On grasp choice, grasp models, and the design of hands for manufacturing tasks. IEEE Trans. Robot. Automa. 5(3), 269–279 (1989)
22. Beksi, W.J., Spruth, J., Papanikolopoulos, N.: Core: a cloud-based object recognition engine for robotics. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4512–4517 (2015)
23. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9–13 (2011)
24. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision-Volume 2–Volume 2, ICCV '99, pp. 1150–1157 (1999)
25. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: 11th European Conference on Computer Vision (ECCV), Hersonissos, Greece, September 5–11 (2010)
26. Bradski, G.: Dr. Dobb's Journal of Software Tools
27. Miller, A.T., Allen, P.K.: Graspit! a versatile simulator for robotic grasping. IEEE Robot. Autom. Mag. **11**(4), 110–122 (2004). Dec

28. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2004), pp. 2149–2154 (2004)
29. Mörwald, T., Vincze, M.: Object Modelling for Cognitive Robotics. PhD Thesis, Vienna University of Technology (2013)