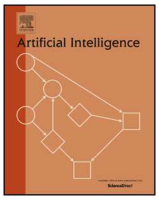
Contents lists available at ScienceDirect
Artificial Intelligence
www.elsevier.com/locate/artint



Transferring skills to humanoid robots by extracting semantic representations from observations of human activities

Karinne Ramirez-Amaro [a,*], Michael Beetz [b], Gordon Cheng [a]

[a] Faculty of Electrical Engineering, Institute for Cognitive Systems, Technical University of Munich, Germany
[b] Institute for Artificial Intelligence, University of Bremen, Germany

A R T I C L E I N F O

Article history:
Received in revised form 22 July 2015
Accepted 19 August 2015
Available online xxxx

Keywords:
Activity recognition
Human understanding
Knowledge-based
Semantic representation
Skill transfer

A B S T R A C T

In this study, we present a framework that infers human activities from observations using semantic representations. The proposed framework can be utilized to address the difficult and challenging problem of transferring tasks and skills to humanoid robots. We propose a method that allows robots to obtain and determine a *higher-level* understanding of a demonstrator's behavior via semantic representations. This abstraction from observations captures the "essence" of the activity, thereby indicating which aspect of the demonstrator's actions should be executed in order to accomplish the required activity. Thus, a *meaningful semantic* description is obtained in terms of human motions and object properties. In addition, we validated the semantic rules obtained in different conditions, i.e., three different and complex kitchen activities: 1) making a pancake; 2) making a sandwich; and 3) setting the table. We present quantitative and qualitative results, which demonstrate that without any further training, our system can deal with time restrictions, different execution styles of the same task by several participants, and different labeling strategies. This means, the rules obtained from one scenario are still valid even for new situations, which demonstrates that the inferred representations do not depend on the task performed. The results show that our system correctly recognized human behaviors in *real-time* in around 87.44% of cases, which was even better than a random participant recognizing the behaviors of another human (about 76.68%). In particular, the semantic rules acquired can be used to effectively improve the dynamic growth of the ontology-base knowledge representation. Hence, this method can be used flexibly across different demonstrations and constraints to infer and achieve a similar goal to that observed. Furthermore, the inference capability introduced in this study was integrated into a joint space control loop for a humanoid robot, an iCub, for achieving similar goals to the human demonstrator *online*.

1. Introduction

Transferring skills to humanoid robots based on observations of human activities is widely considered to be one of the most effective ways of increasing the capabilities of such systems [1,2]. It is expected that semantic representations of human activities will play a key role in advancing these sophisticated systems beyond their current capabilities, which will enable these robots to obtain and determine *high level* understanding of human behavior. The ability to automatically

* Corresponding author at: Technische Universität München, Institute for Cognitive Systems, Arcisstrasse 21, 80290, München, Germany. Tel: +49 89 289 26791.
E-mail addresses: karinne.ramirez@tum.de (K. Ramirez-Amaro), beetz@cs.uni-bremen.de (M. Beetz), gordon@tum.de (G. Cheng).

http://dx.doi.org/10.1016/j.artint.2015.08.009
0004-3702/© 2015 Elsevier B.V. All rights reserved.

recognize human behavior and react to it by generating the next probable motion or action according to human expectations will substantially enrich humanoid robots.

The main steps involved in inferring and reproducing the goal of the demonstrator's activity are [3]: 1) extracting the relevant aspects of the task; 2) processing the perceived information to infer the goal of the demonstrator; and 3) reproducing the best motion to achieve the inferred goal. Thus, in order to infer the desired goal, we first need a perception module that can determine which aspects of the activity are *important*. The most challenging aspect of this process is extracting the (visual) information from different sources, e.g., body movements, changes in the environment, and the gaze of the demonstrator, to obtain meaningful information about the system.

However, not all aspects of a task are observable but they may need to be inferred, such as the *goal* of the demonstrator, which in most systems is given by the researchers (as noted by [2]). Therefore, it is necessary to design a method that can process the perceived goal-relevant information to make inferences about the goal of the demonstrator. A powerful tool for accomplishing this aim is semantic representation, since the meaning of human behaviors can be extracted and expressed using the relationships between human motions and objects. In this study, we use the latter point as a definition for the *semantics of human behavior*.

Finally, the robot needs to decide how to execute the inferred goal, which means the robot determines which aspects of the observed (inferred) task should be performed in order to achieve a similar task that of a human. This mapping between the robot and the demonstrator is a very difficult problem, which is known as the "correspondence problem" because the robot and the human sometimes have different embodiments [4]. It has been shown that it is more useful to extract the *goal* from the demonstrator's activity, rather than simply copying the human's actions [2]. This allows the robot to evaluate and then decide the best way to achieve the goal, while considering its own constraints, i.e., whether to use the demonstrator's method or its own method to achieve the same outcome.

Thus, the robot requires some knowledge about the objects and actions, as well as their relationships. For instance, the system could have an ontology-based knowledge representation, which provides the ability to deal with partial information from the perceived environment. This is possible as the ontology representation has a hierarchical structure in the knowledge base, where the inferred instances inherit the properties and relations of the parental class. In other words, it is possible to infer new properties between instances that are not provided manually. This is a major advantage due to the fact the system can be more adaptable and flexible than classic approaches [5].

1.1. System overview

In this study, we present a framework that comprises three main modules: 1) observation of *human motions* and object properties; 2) hierarchical interpretation of *human behaviors*; and 3) activity imitation by a robot. A description of our framework and the connections between these modules is depicted in Fig. 1.

The first module (see Fig. 1.1) allows robots to perceive different types of visual information from different sources, such as video recordings. Thus, the visual features obtained from the environment (raw video) are analyzed and preprocessed, and three general motions are segmented from the videos, i.e., *move*, *not move*, and *tool use*. We refer to these motions as *human motions*. In addition, two main types of information are acquired from the objects and their properties present in the scene: *ObjectActedOn* and *ObjectInHand*.

The second module (see Fig. 1.2) interprets the visual data obtained from the first module and processes this information to infer the observed human intentions (the goal). This module represents the core of our framework because it is responsible for identifying and extracting the meaning of human motions by generating semantic rules that define and explain these human motions, i.e., it infers *human behaviors* such as reach, take, pour, and cut. At this point, the system evaluates whether the input information provided can be used to infer the human activity. It should be noted that this is the step where most systems fail [6]. For example, if we only have objects such as bread, knife, and cucumber during the training stage, then the system will be limited to only accepting these values in order to infer human activities correctly. However, we enhance our system by the inclusion of a knowledge base, which means that if we have new objects such as pancake and spatula during the testing stage, then the system can look for their corresponding class and infer the human activity associated with that class. Thus, if the evaluation fails, the system will use its knowledge base to obtain an alternative and equivalent state for the system that contains instances of the desired class. This represents a very important advantage of our system, as with the knowledge base means that we do not need to recompute the semantic rules every time we meet a new situation, i.e., the generated rules will even be preserved in different scenarios (see Section 4).

The input of the third module is the inferred activity from Module 2. In this module, the robot executes the motion primitive that may yield a similar activity to that observed. This means that given an activity, the robot needs to execute a skill plan to command the desired primitives from a library, until the inferred activity is successfully achieved by the robot.

1.2. Experimental setup

To validate our framework, we first recorded two real and challenging tasks[1]: *pancake*, and *sandwich making*. The experimental setup was similar for both tasks (see Fig. 2), where it comprised three cameras located in different positions and a

[1] The new recorded datasets are publicly available via the following link http://web.ics.ei.tum.de/~karinne/DataSet/dataSet.html.

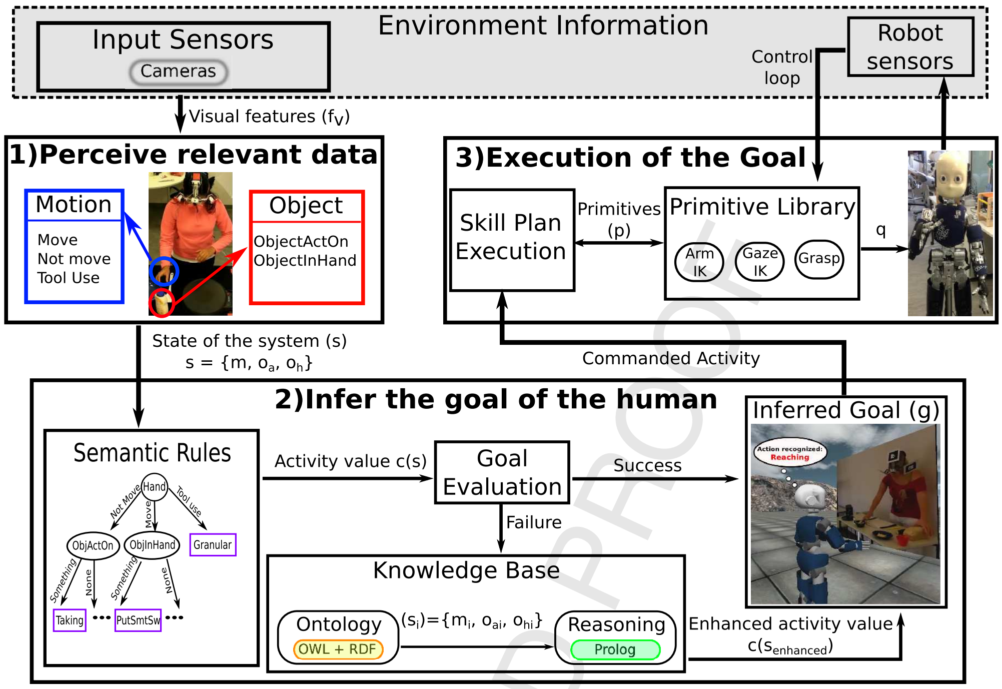


Fig. 1. The framework comprises three main modules: 1) perception of relevant information, 2) inference of the observed goal, and 3) execution of the goal by the robot. The core of our framework is defined in the second module, which defines the semantic rules and the enhancement of the system by the knowledge base.

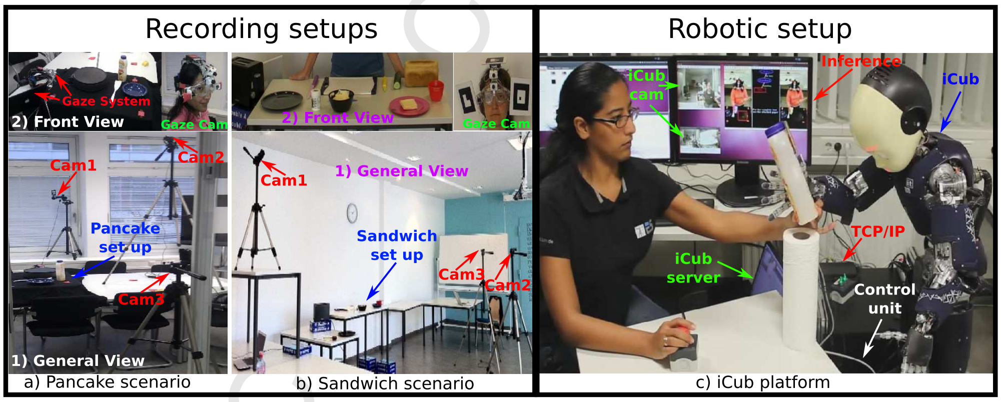


Fig. 2. Setup used for dataset acquisition and the robotic experimental setup. a) The location of the cameras during the *pancake making* scenario. b) The recording setup in the sandwich-making scenario. c) Robot setup used for the experimental validation in this study.

gaze camera with attached markers. In this study, we only used video information of the second camera in both scenarios (see Section 6), i.e., information from the gaze camera and the markers was not considered.

In addition, we use a robotic setup to successfully validate the proposed framework (see Section 7.3). The Robot Control System comprises the humanoid robot iCub, a control unit, and a workstation running GNU/Linux OS. Data communication between the PC and the control unit was via a local network, based on TCP/IP.

The following sections provide detailed descriptions of our proposed framework. Section 2 describes the state-of-the-art in this research area. Section 3 briefly explains the procedure implemented to segment the observed human motions and objects. Section 4 introduces the method used to obtain the semantic rules and the ontology-based knowledge representation. After, Section 5 presents the procedure employed to transfer the observed activities to the humanoid robot. Section 6 describes the datasets used and the results obtained are analyzed in Section 7. Finally, Section 8 gives our conclusions.

2. State-of-the-art

Automatically segmenting, recognizing, and understanding human activities based on observations has attracted the attention of researchers from different disciplines, such as computer vision [7–9], artificial intelligence [10,11,6], cognitive science [12–15], robotics [16–18], and neurology [19] each of whom has focused on solving a subset of the complex problem of interpreting human activities for different purposes, e.g., surveillance systems [7,20], monitoring patients [10,21], and anticipation of human behaviors thereby allowing robots to assist humans in an efficient manner [22–24]. The goal of human activity recognition was described by Aggarwal and Ryoo [10] as to: "automatically analyze ongoing activities from an unknown video". These authors also identified various types of human activities, which they categorized on different levels, depending on their complexity, such as gestures, actions, interactions, and group activities. For example, gestures refer to general human movements such as *raising a leg* or *stretching an arm*. Therefore, the approach employed depends on the complexity of the activity and several challenges must be addressed, including automatically segmenting the observed human motions, identifying the important features of motions, determining the importance of the object(s) for the task, and defining different levels of abstraction. One of the main issues in these problem domains is, that it is not easy to translate methods from one area to solve a similar problem in another area. Therefore, human activity recognition is still far from being an off-the-shelf technology [10]. Thus, how can we transfer the findings and advantages of one technique to solve a similar problem in a different domain?

As for the robotics domain, the problem of *activity recognition* involves solving the question: *what do we want the robot to learn?* [2], either in terms of a similar motion or extracting the meaning of the motion. The first problem is mostly investigated by analyzing the trajectories produced by human motions, i.e., using trajectory level methods. The second problem involves determining different levels of abstraction to extract meaningful information from the task with Artificial Intelligence methods, e.g., using semantic representation techniques. In the following subsections, we analyze these problems in more detail to solve the issue of activity recognition based on observations.

2.1. Trajectory level

In the robotics community, there has been a tendency to use trajectory level representations, i.e., the Cartesian and Joint spaces, to segment and learn a correct model of human motions. Thus, the robots learn and extract the parameters involved in a task/skill based on demonstrations, typically from several trials of the same task and mostly in one specific scenario [25,26], e.g., using programming-by-demonstration techniques [18], which are powerful and well-established methods that are used widely in the robotics community to teach robots new activities based on observations. In addition, an interesting approach was presented by Billard et al. [18], that identifies a general policy for learning the *relevant* features of the task, where they identified *what to imitate* by detecting the time-invariants of the demonstrator [27]. Recently, Ude et al. [28] proposed the concept of using a library of *dynamic motion primitives* (DMPs), which allows the generalization of DMPs to new situations. The advantage of this approach is that it considers perturbations and includes feedback [29]. Therefore, *relevant* parameters are identified to reproduce motions similar to those made by the demonstrator, while similar motions can be adjusted based on the parameterization of the given goal, where the goals are generally provided manually in this case, i.e., the system (robot) will not be able to extract the meaning of the action because they do not possess reasoning capabilities. Therefore, the transfer of the models obtained to new situations is not straight forward.

Some techniques have been proposed for classifying human motions based only on the shape of the trajectory, without considering the object's information into account, e.g., using similarity measurements such as dynamic time warping [30] or using a multi-step hierarchical clustering algorithm motivated by classification and regression trees [31]. These latter techniques rely on the generation of trajectories depending on the locations of objects, so the trajectories will be altered completely if a different environment is being analyzed, thus new models must be acquired for classification. Another drawback of this approach is that segmentation of the trajectories into meaningful classes is mostly conducted by the researchers, i.e., it is performed manually. Another approach was presented by Beetz et al. [11] where a hierarchical action model is constructed from observed human tracking, based on linear-chain Conditional Random Fields, which uses pose-related features applied to labeled training data. The latter approach considers the information related to objects, but several parameters need to be adjusted in advance, which means that recognizing activities from the same class in a different environment is not possible.

These type of trajectory level techniques are very useful for extracting *relevant* information from activities, as well as for transferring these models to artificial systems such as robots. Transferring the models acquired from human demonstrations to robots is another challenging task when building adaptive and autonomous robots, mainly because it requires the generation of task-specific robot motions that should fit naturally in a human environment. This means the robot should generate motions based on the current task, the object type and the current state, while also considering information related to the context and environment. This requires the analysis of stereotypical and pre-planned human motion patterns in order to acquire the desired task [32]. Therefore, the major advantage of these approaches is their ability to analyze the details of human movements [10]. However, this analysis can only be performed by using very sophisticated perception systems to identify the poses of human joints, e.g., motion capturing systems or state-of-the-art tracking systems [33]. Another drawback of these methods is the inability to generalize the learned models to new situations, mainly since they depend on the correct identification of human poses that are difficult to extract from 2D videos, which is the problem addressed in this study.

2.2. Semantic representation

Recent studies have focused on determining the levels of abstraction to extract meaningful information from a task to determine *what* the specific task might be and *why* it is recognized. Hierarchical approaches are capable of recognizing high level activities with more complex temporal structures [10]. These approaches are suitable for the semantic-level analysis of humans and/or objects. They can also cope with less training data by incorporating prior knowledge into their representations. This prior knowledge is mostly included manually by an expert, who gives a semantic meaning to the sub-activities that comprise the high level activity. These mechanisms help to understand the meaning of the recognized task and allow the system to be more flexible and adaptable to new situations. This area is the main focus of our research and a more extensive analysis of these techniques is provided in the following.

A pioneering study of high level representations was introduced by Kuniyoshi et al. [34], who proposed the mapping of continuous real-world events onto symbolic concepts, using an active attention control system. A similar study by Jakel et al. [35], employed a (partially) symbolic representation of manipulation strategies to generate robot plans, based on pre- and post-conditions. However, these frameworks cannot reason about the intentions of the user or extract the meanings of actions. Another study that addressed this problem was presented by Fern et al. [36], who introduced a logic sub-language to learn specific-to-general event definitions, using manual correspondence information.

With respect to the inclusion of the term *semantics* in the recognition of human behaviors, Park et al. [6] defined semantic descriptions using the linguistic “verb argument structure” based on ⟨*agent–motion–target*⟩ triples for the recognition of human behaviors defined by the relationships of subject–verb–object relationships. To obtain these triples, the authors associated visual features with natural language verbs and symbols from a defined vocabulary to build the semantic descriptions of video events. An advantage of natural-language descriptions is the rich syntactic and semantic structure used to represent domain-free rules and contexts. However, the number of triples must be defined in advance and multiple triples may be needed for a specific action depending on the complexity of the action. Therefore, the re-usability of these triples in new situations is not possible.

From a robotics viewpoint, Takano et al. [37] proposed an approach for encoding observed trajectories based on hidden Markov model (HMM) as mimesis models in order to segment and generate humanoid robot motions through imitation. They transformed the motion patterns into location proto-symbols in a topological space, called the proto-symbol space. However, one of the weak points of this mimesis model is the need to find a physical meaning for each dimension of the proto-symbol space. This issue was addressed by Inamura et al. [17] where a physical meaning was obtained using natural language. This system allows the generation of novel motion patterns, but its use is limited to the joint angles when creating the proto-symbol space. This means that this system cannot work with the datasets proposed in our study, because the joint angles are not available.

Another interesting definition of semantic representations was given by Turaga et al. [5], where they suggested the semantics of human activities requires higher level representations and reasoning methods. They discussed the following approaches: graphical models (belief networks, Petri nets, etc.), syntactic approaches (grammars, stochastic grammars, etc.), and knowledge and logic approaches (logic-based approaches, ontologies, etc.). Therefore, the semantic definition of these activities will depend on the approach used. For example, context-free grammars and stochastic context-free grammars have been used by previous researchers to recognize high level activities [10]. These grammars are typically used as a formal syntax for the representation of human activities. This means, these grammars directly describe the semantics of activities.

The concept of object–action complexes (OACs) introduced by Wörgötter et al. [13] is employed to investigate the transformation of objects by actions, i.e., how object A (cup-full) changes into object B (cup-empty) by executing action C (drinking).[2] This approach was used recently to segment and recognize an action from a library of OACs using the preconditions and effects of each sub-action, which allows a robot to reproduce the demonstrated activity [38]. However, this system requires a robust perception system to correctly identify the attributes (full-empty) of the objects, which are obtained and executed off line. Analogous to OACs and based on the affordance principle, Aksoy et al. [14] presented the *semantic event chain*, which determines the interactions between the hand and objects, where they are expressed in a *rule–character* form. These interactions are based on the changes in the visual environment represented in a dynamic graph, where the nodes are the centers of the image segments and the edges are defined based on whether or not two segments *touch* each other. In other words, the spatial relationship between the graphs is stored in a transition matrix, which represents the *semantic event chain*. One drawback of this technique is that it is highly dependent on time and the perception system to define the object interactions. Thus, if the computer vision system fails, this approach will be affected greatly.

Recently, Koppula and Saxena [22] modeled the activity and object affordances to anticipate or predict future activities. They introduced an anticipatory temporal conditional random field that models the spatio-temporal relationships by object affordances based on the concepts and methods introduced in [12]. This technique performs very well, but the type of

[2] This action is defined by the current attribute of object A.

interaction will define the affordances that are considered from a predefined library. Based on this previous studies, the same authors [23] proposed the sampling of the spatio-temporal structure in addition to the future nodes to enhance temporal segmentation and to address the anticipation problem by considering multiple graph structures. However, the structure of the graphs needs to be fully known and this method relies on correct temporal segmentation. We consider that the latter approach can be enhanced by using our method to segment and infer activities. Another study that addressed the problem of human intentions was presented by Gehrig et al. [39], where their framework combined motion, activity, and intention recognition by humans using visual information obtained from a monocular camera combined with the knowledge domain. This system is restricted to manual annotations of the domain (time of day and the presence of the object). In addition, the relationship between the activity and motion is neglected.

Yang et al. [40] introduced a system that can *understand* actions based on their consequences, e.g., split or merge, but the core of this technique relies on a robust active tracking and segmentation method, which can detect changes in the manipulated object, its appearance, and its topological structure, i.e., the consequences of the action. Subsequently, this system was improved by including a library of plans that comprised primitive action descriptions, as presented in [41]. However, this system was not implemented in a robot and it will fail if the plan is not known *a priori*. Another study based on plan recognition by Kautz et al. [42] states that human behavior follows stereotypical patterns, which can be expressed as preconditions and effects. However, these constraints must be specified in advance, which is a problem when trying to use them in different domains.

Ontology-based activity recognition was proposed by Patterson et al. [9], where a model can generalize object instances by its classes using abstract reasoning. One problem with this model is that the activities are misclassified in some cases because the activities belong to the same class as the object. For example, the activities *doing laundry* and *getting dressed* are misclassified because they have the same object class (clothes). In terms of recognition using knowledge representations, the method introduced in [43] employs a practical approach for defining robot knowledge, which combines description logic knowledge bases with data mining and (self-)observation modules. The robot collects experiences while executing actions and uses them to learn models and aspects of action-related concepts, which are grounded in its perception and action system. However, in this knowledge representation system, it is necessary to manually specify the object–action relationships.

The semantic or symbolic representation techniques used to recognize human activities combine the information obtained from image sequences with their trajectories, thereby yielding more accurate systems for recognizing human activities in real scenarios. These techniques allow us to abstract the recognition problem by mapping continuous motions into symbolic events which allow us to recognize human activities as well as segmenting them over time. These transformations depend greatly on the context because they need to define the preconditions, postconditions, effects, and/or affordances produced by the activities, which are not always the same. For example, the *lifting* activity could produce different effects, e.g., *lifting for a cup from the table* will mean that the top of the table is empty while the effect of *lifting a fork from a drawer* will result in the drawer being lighter. Thus, these activities will be classified as different, although they represent the same behavior (*lifting*). Another disadvantage of these techniques is, they also depend on a highly accurate perception system to detect the effects of the activity performed or the object affordances.

Action recognition and segmentation are very difficult and challenging problems. As noted above, the computer vision and machine learning communities are focused mainly on solving the problem of recognition while disregarding the problem of segmentation, which is generally performed manually. However, the robotics community needs to solve both problems in an effective, reliable, and fast manner to allow robots to make the best decisions. In the present study, we address the non-trivial problems of the segmentation, recognition, and transference of human activities to a robotic system. Thus, our system has the advantages of the techniques mentioned above but it successfully overcomes their problems to obtain good performance, with accuracy greater than 85% when segmenting and recognizing human activities in real scenarios. These findings are presented in the following sections.

3. Definition and extraction of visual information

Transferring the observed tasks from humans to robots typically requires sophisticated perception methods [6,40,22,12,7,44]. These methods need to process the visual input as rapidly and accurately as possible to automatically extract and interpret information from incoming sources, e.g., videos. This is a very complex problem since these techniques need to segment the continuous stream of human behavior and object information into meaningful classifications, preferably *online*. Nevertheless, most of the available techniques focus only on obtaining the best accuracy [7] and they typically work with *offline* applications [44].

In, this section, we describe the first module of our proposed framework (see Fig. 1) where the main goal is to perceive and extract the *relevant* information *online* from the observed human activities, which also includes information about objects. To achieve this aim, we propose a new hierarchical approach for recognizing *high level* human activities based on simple motions. Therefore, in this study, we split the complexity of the recognition problem into two parts: the first gathers (perceives) information from the objects using simple perception techniques, e.g., color-based tracking, which we explain briefly in this section; and the second part handles the difficult problem of assigning the perceived information to meaningful classifications using an inference module. The levels of abstraction in our approach are presented in Section 4.

Table 1
Definitions of *human motions* and object properties.

Name	Meaning	Formula	Example
Hand motions (m)			
Move	The hand is moving	$\dot{x} > \varepsilon$	Moving from position A to position B
Not move	The hand stops moving	$\dot{x} \to 0$	Hold the bread
Tool use	Complex motion using two objects: one is used as a tool and the second is the object that receives the action	$o_h(t) = knife$ and $o_a(t) = bread$	Cutting the bread where the objects are the knife and bread
Object properties			
Object acted on (o_a)	The hand moves toward an object	$d(x_h, x_o) = \|x_h(t) - x_o(t)\| \to 0$	Reaching the bread, where $o_a(t) = bread$
Object in hand (o_h)	The object is in the hand, i.e., o_h is currently manipulated	$d(x_h, x_o) \approx 0$	Hold/take the bread, where $o_h(t) = bread$

An important aspect of module 1 is defining *what* is considered to be *relevant* information, i.e., what is the highest level of abstraction for recognizing the intentions of humans based on observable data. For example, in the case of *low level* hand motions (m), we can segment them into three main categories: *move*, *not move*, or *tool use*. These simple motions can be recognized in different scenarios but they cannot define an activity by themselves. Therefore, we need to add contextual information, such as the objects in the scene and their properties. Thus, the motion information combined with the object properties has more meaning than the separate entities. The properties which can be easily recognized from observations are *ObjectActedOn* (o_a) and *ObjectInHand* (o_h). These properties are implemented in our knowledge base as *Computable Properties* for which the argument is the time frame t, as explained in Section 4.2. Definitions and examples of identified motions and object properties are given in Table 1.

In our previous study [44], we demonstrated this hierarchical approach using an unsupervised state-of-the art learning algorithm based on independent subspace analysis (ISA) [8] to extract spatio-temporal features from videos. The results suggested the accuracy of correctly identifying *human behaviors* exceeded 70%, which is very good compared with single-layered approaches (around 25%). Another advantage of our hierarchical approach is that time required for the training process is reduced from 2–3 days to 1–2 hours.[3] However, the ISA algorithm needs to be executed *offline* to classify human motions. Another disadvantage of this method is that segmentation of object properties required manual labeling annotations. Therefore, in order to solve the issues described above and to demonstrate that our system does not rely on a complex perception system, we propose the use of the simplest computer vision technique to perform *online* recognition of simple human motions, including the object properties. Thus, we employ the well-known *color-based tracking*[4] algorithm, which is typically used to detect, segment, and track an object in an image. Developing a new perception technique was beyond the scope of this study.

To implement the *color-based* algorithm, we used the Open Source Computer Vision (OpenCV) library [45] to obtain the color features (f_v). This technique is used to determine the hand position (x_h), which is used to compute its velocity ($\dot{x}_h$). The velocity is then used to define hand motions (m). A similar procedure is performed for the objects in the scene, to determine the object properties (o_a or o_h). The algorithms and more detailed information about our *color-based* system are presented in [46]. It is important to note the recognized object (o_i) can only satisfy one of the object properties mentioned above, i.e., $o_a(t) = o_i$ or $o_h(t) = o_i$, and not both in the same frame t. However, it is still possible to have more than one object in the scene, e.g., $o_1 = pancake$ and $o_2 = spatula$, where the object properties are $o_a(t) = o_1$ and $o_h(t) = o_2$. If more than two objects are recognized in the scene and one of them is true for o_a and the other is true for o_h (as in the previous example), then the motion is classified as *tool use*.

4. Inference of the demonstrator's goal

This module is the most important in our framework. As inputs, this module receives the hand motion (m) and the properties of the object(s), (o_a or o_h), identified by the previous module (see Fig. 1). To produce the output, it infers one of the following human activities: *reaching, taking, cutting, pouring, releasing*, etc. If the inference process fails, the system obtains the parental class of the object from the knowledge base and use this new class to infer the human activity. Therefore, we use the parental class rules to infer the correct human activity that is best associated with the current object. This last step represents the enhancement of our system compared with classic approaches.

4.1. Extracting semantic representations

Semantics refer to the *study of meaning* and several methods have been used to define *semantics* in the domain of human behavior such as linguistic descriptions [6,17], syntactic approaches [5,10], and graphical models [13,22]. In general, the

[3] The experiments were performed with a PC-desktop with 8 GB RAM and an Intel® Core™ i7.
[4] The color-based algorithm extracts image blobs based on the color of the objects and it will segment the video by thresholding the given color(s), i.e., the color feature is an attribute of an specific object which is previously defined.

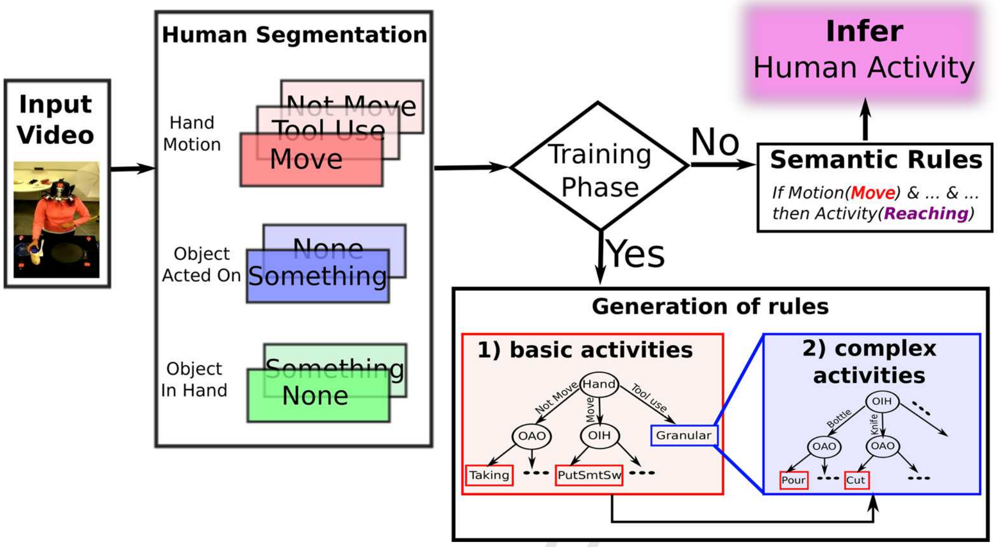


Fig. 3. Method for inferring human activities. First, the input video is segmented into hand motions and two object properties. Next, we compute the rules if we are in the training phase, but if not, we use the rules obtained to infer human activities.

semantic representations used in previous methods are given *a priori* by an expert [10]. Therefore, we propose a method that automatically identifies the *semantics of human behavior*, which involves finding meaningful relationships between human motions and object properties in order to understand the activity performed by humans, i.e., the *semantic representations* are used to interpret a visual input to understand human activities. Next, to define the *semantics of human behavior*, we propose two levels of abstraction: *low level* abstraction, which describes generalized motions (obtained from module 1) such as *move*, *not_move*, or *tool_use*, and *high level* abstraction, which represents *human behaviors* such as *idle, reach, take, cut, pour, put* something somewhere, *release*, etc. The hierarchical approach that we propose uses information about general *motions* and object properties to infer *human behaviors*.

A decision tree classifier is used to learn the mapping between *human motions* and *human behaviors* based on object information. A previous study also used decision trees to define domain-specific rules to determine meaningful semantic representations with spatial and temporal constraints [6]. However, this approach only recognized interactions with two sequential activities, where the individual leaf node represents an interaction type that can be obtained with different combinations of $\langle$*agent–motion–target*$\rangle$ triples, and thus the semantics of a human behavior are not unique. Therefore, this method cannot guarantee that the learned *rule* can be used in a different situation. This point is very important when defining *semantics*, i.e., the possibility of re-using an extracted meaning in different scenarios. Hence, our system focuses on solving the problem by using a decision tree classifier but we employ a different definition of triples to make the *semantics of human behavior* generalizable to new scenarios.

The method proposed in this study is depicted in Fig. 3 and it comprises two steps: the first step generates a tree that can determine the *basic* human activities in a general form, i.e., *reach, take, put, release, and idle*; and the second extends the tree obtained tree to recognize more *complex* activities. We refer to these types of activities as *granular* activities, e.g., *cut*, *pour*, *spread*, and *flip*. The major differences between these types of behaviors is the environment (context) as explain in the following subsections.

4.1.1. Method for identifying basic human activities

To learn the decision tree, we require a set of training samples D, which comprise a set of *instances* (S) and the target concept (c). The set of instances is the set of items over which the concept is defined. Each instance ($s \in S$) describes a specific state of the system and it is represented by its attributes (A). The concept or function that needs to be learned is called the *target concept*, which is denoted by c. In general, c can be any n-valued function defined over the instances S, i.e., $c : S \to \{0, 1, .., n\}$. In our method the target concept corresponds to the value of the attribute *ActivityRecognition*. When learning the target concept (c), the learner is presented with a set of training examples (D), each of which comprises one instance s and its target concept value $c(s)$. We refer to the ordered pair $\langle s, c(s)\rangle$ as a *state-value pair* to describe the training sample (D). In this study, the set of instances S is described by a triple with the following attributes:

1. *Hand_motion* (with the possible values: Move, Not_move, and Tool_use)
2. *ObjectActedOn* (with the possible values: Something, None)
3. *ObjectInHand* (with the possible values: Something, None)

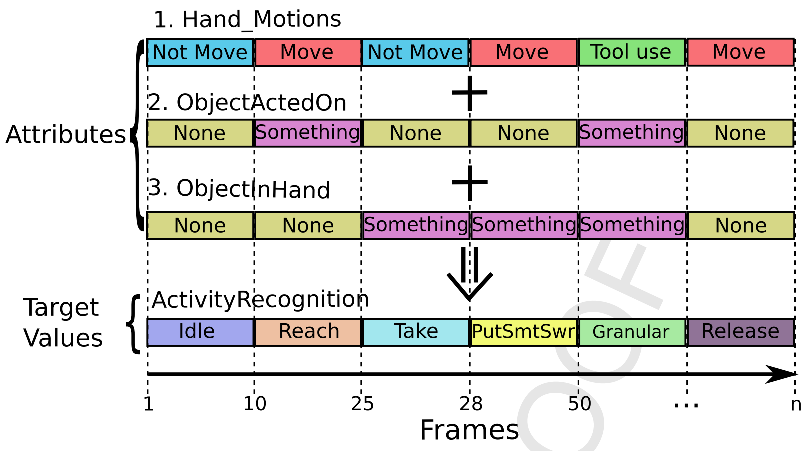


Fig. 4. Some of the *state-value* pairs used to generate the decision tree. Top: three input attributes and their corresponding values per frame. Bottom: representation of the target concept values. Note that every column describes the duration of each activity in the frames. For example, the activity *Reach* requires 15 frames and the activity *Take* is executed in three frames.

and the *target concept value*:

- Class c : *ActivityRecognition* : $S \rightarrow$ {Reach, Take, Release, Put_Something_ Somewhere, Idle, Granular[5]}

Therefore, some examples of the *state-value pair* ($\langle s, c(s)\rangle$) are:

$$\langle\{\ Not\_Move\ ,\ None\ ,\ None\ \}\ ,IdleMotion\rangle$$
$$\langle\{\ Move\ ,\ \ Something\ ,\ None\ \},\ \ \ Reach\ \ \ \rangle$$
$$\langle\{Not\_Move,\ None,\ Something\},\ \ \ Take\ \ \ \rangle$$

The first three elements correspond to the current state s, whereas the final item represents the target concept value c. Fig. 4 illustrates the training sample set.

According to the principle of *Occam's razor*: "It is preferable to choose the simplest hypothesis that fits the data" [47]. Therefore, in order to learn the target function c from a set of training instances S, we use the C4.5 algorithm [48] to compute the decision tree, where shortest trees are preferred over longer trees and it is guaranteed that attributes closest to the root achieve the highest information gain. In addition, decision trees can be represented as sets of *if–then* rules to improve human readability. The core aim of the C4.5 algorithm is to select the most useful attribute to classify as many samples as possible using the information gain measure:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entropy(S_v), \tag{1}$$

where $Values(A)$ comprises the set of all possible values of the attribute A and $S_v = s \in S|A(s) = v$ is a collection of samples for S, and the entropy is defined as:

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2 p_i, \tag{2}$$

where p_i is the probability of S belonging to class i.

The method for inferring human activities is presented in Algorithm 1, which as input takes the train or test dataset ($D = \langle s, c(s)\rangle$). Note, that steps 1–3 in Algorithm 1 are performed only once and the function **Compute_Tree**($\langle s, c(s)\rangle$, A) is called, where s represents the training instances, $c(s)$ is the attribute for which value has to be predicted by the tree, and A is a list of the other attributes that may be tested by the taught decision tree. This function returns the tree obtained T which captures the rules that can be used later to infer human activities, even in new scenarios. This leads us to the second most important point of the algorithm where steps 4–8 are performed whenever a new dataset is considered. Thus, we do not learn a new tree, but instead we use the tree obtained T to infer the target concept c for the new states of the system. Fig. 3 shows the key features of this methodology.

[5] Granular activities define classes such as flipping, pouring, and cutting. These activities are difficult to generalize because they depend on the context.

Algorithm 1 Inferring human activities.

```
Require: Dataset ($D = \langle s, c(s) \rangle$)
    {Note that Dataset can be the training set or a new set}
1: if Training phase then
2:     T = Compute_Tree($\langle s, c(s) \rangle$, $A$). {This function returns a decision tree that correctly classifies the given instances using Eq. (1) and Eq. (2).}
3:     return $T$, the decision tree that best classifies $s$ and that determines the hypothesis $h \in H$ such that $h(s) = c(s)$ for all $s \in S$. {The set of hypotheses $H$ is referred to as the semantic rules.}
4: else
5:     for each current state $s$ do
6:         Determine: the target class $c(s)$ from the set of hypotheses $H$ obtained from step 3
7:     end for
8:     return $c(s)$, the inferred human activity
9: end if
```

4.1.2. Method for recognizing complex human activities

In order to correctly infer complex activities, such as *cut*, *pour*, *spread*, and *flip*, more attributes must be considered. For instance, we may consider the type of object that is being manipulated, such as the *cut* and *spread* activities, which both use a *knife* as a tool but they represent different activities. A distinction between these two activities is that the object on which they act, (o_a), is either bread or mayonnaise, respectively. Therefore, a second stage is needed to extend our obtained tree T so we can infer these *granular* activities.

In the second step, the input comprises the activities clustered as *Granular* in the previous step and we learn a new tree, which represents an extension of our previous tree. The method employed is similar to that explained in Section 4.1.1. Thus, the set of instances S is described by the same attributes A but with different values. For example, the *Hand_motion* attribute now has only two possible values: *move* or *not_move*. The attribute *ObjectActedOn* represents the new possible values: *pancake, dough, bread, cheese, electric stove*, etc.; whereas the *ObjectInHand* attribute has four possible values: *bottle, spatula, knife*, and *plastic wrap*. Note that the values of the last attribute are the parental classes of the objects. The next subsection explains the class definitions.

Some examples of the new *state-value* pairs ($\langle s, c(s) \rangle$) are as follows:

$$\langle \{\, move\ ,\ pancake\ ,\ spatula\ \}\ , Slide\_out \rangle$$
$$\langle \{\, move\ ,\ bread\ ,\ knife\ \}\ ,\ Cut \rangle$$
$$\langle \{not\_move,\ cheese,\ bottle\},\ Sprinkle \rangle$$

The two-step method is depicted in Fig. 3, which shows the generation of the decision tree T using the two-step method during the training stage.

4.2. Knowledge and reasoning engine

In the field of robotics, especially human–robot interactions, it is important to provide robots with decision-making capabilities in order to increase their adaptability and flexibility in different environments. These capabilities can be achieved with semantic representations. The *semantics* (construction of *meaning*) can be enhanced if a knowledge base and reasoning engine are integrated into the system [49]. Developing a proper knowledge base requires the careful analysis of the domain, choosing the appropriate vocabulary, and encoding the reasoning engine to obtain the desired inferences [49]. The last point is very important because the main goals of the reasoning process are: 1) to create representations of the world, 2) to use an inference process to derive new representations of the world, and 3) to use these new representations to deduce what to do next.

Knowledge and reasoning play a crucial role in handling partially observable information because they are capable of inferring or predicting different behaviors in the same manner as humans. This is partly because the knowledge base system can combine general knowledge with the current perception of the world to infer hidden aspects of the current state. The key factor here is to define mechanisms for obtaining appropriate reasoning rules to allow the inference of meaningful relationships in the perceived environment. Therefore, the following questions arise: *how do these rules have to be defined?*, *who defines these rules (e.g., an expert)?*, *how can we guarantee these rules will be valid for different situations?*, and *what is the level of generalization for these rules?* These questions need to be answered because a good knowledge-reasoning system should be able to adapt and be flexible to changes in the environment by updating the relevant knowledge.

Abstract concepts such as action, space, time, and physical objects are sometimes defined by an ontology representation. The Web Ontology Language (OWL) is commonly employed to represent knowledge because it uses semantic data modeling, e.g., Resource Description Framework (RDF), which is an action representation based on Description Logics (DL) such as Prolog queries. Therefore, knowledge was represented by OWL in the present study. In addition, we used KnowRob [43] as our base line ontology, which is an extension of the OpenCyc ontology, a general upper ontology that covers a broad range of human knowledge. We mainly used two branches of the KnowRob ontology: *TemporalThings* and *SpatialThings*, where the first one contains the important subclasses of *Actions* and the second describes abstract spatial concepts such as places

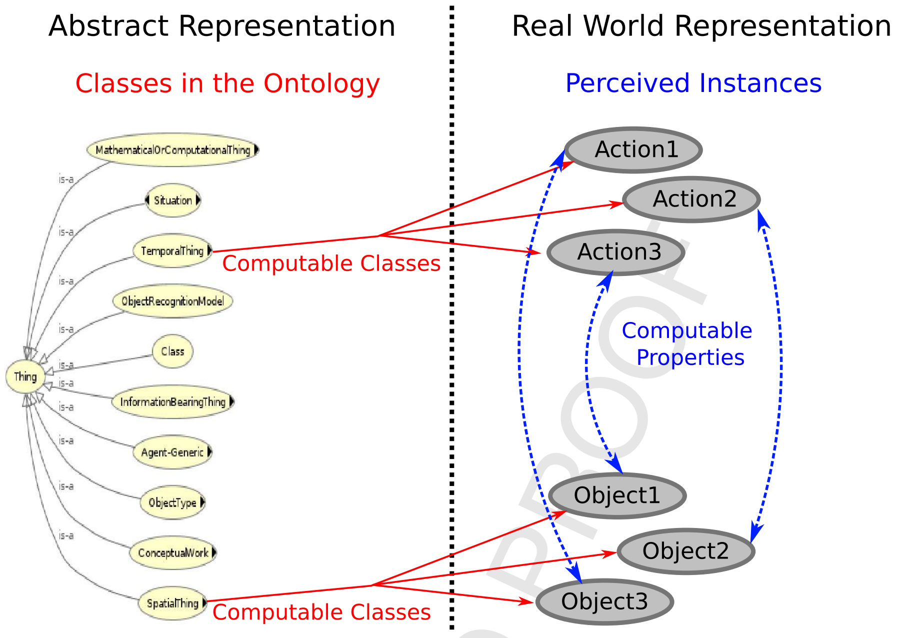


Fig. 5. Illustration of the two types of computables: *Computable Classes* and *Computable Properties*. Left: The manually created classes inside the ontology. Right: The perceived instances created on demand, using the *Computables Classes*, as well as the relationship between these instances given by the *Computable Properties*.

Algorithm 2 General representation of the inputs and outputs for a knowledge and the reasoning system.

```
Require: $KB \leftarrow KnowRob$
    {KB = Knowledge Base}
1: TELL($KB, Perceive\_Object(O_{instance})$)
2: TELL($KB, Perceive\_Action(A_{instance})$)
    {In this step, the Computable Classes are executed and new instances are generated in the KB}
3: $human\_activity \leftarrow$ ASK($KB$, $Relation\_Query(?)$)
    {In this step, the Computable Properties are obtained from the KB}
4: return $human\_activity$
```

and *object* classes. KnowRob is mainly implemented in Semantic Web Implementation (SWI) prolog [50], which loads the ontology information as RDF triples, and reasoning is performed by the Thea OWL parser library [51].

Reasoning with *Computables* is another important characteristic of KnowRob because it allows the possibility of computing new relations during the reasoning process (on demand) instead of defining them manually. This is important because the environment which we evaluated our system is mostly dynamic, i.e., the states change over time. There are two types of computables: *Computable Classes*, which create instances of their target class, and *Computable Properties*, which compute relations between instances (see Fig. 5). *Computable Classes* are to define more straightforward because they retrieve stored instances from a dataset, e.g., from MySql, or they are created when a new instance is perceived by any sensor in the system. By contrast, defining *Computable Properties* is not an easy task since they determine new relationships between objects and activities. Therefore, in the following work we focus on the appropriate definition of those properties.

4.2.1. New algorithm for expanding the ontology

In order to define *meaningful* relationships between activities and objects, we use the *semantic rules* obtained as described in the previous Section 4.1. These rules generate new instances and new relationships between instances (objects-properties). These object properties are obtained by the definition of new computables. The created instances and relationships are then added to the ontology as part of the inferred knowledge base. A very general way of representing the inputs and outputs for a knowledge and reasoning system is represented in Algorithm 2.

A key point of our reasoning engine is, that inference mechanisms are used to derive new representations about the world, i.e., these are not manually included in the knowledge base. It is important to note that in order to define our knowledge base, we focus on the domain of everyday human activities. To enhance our ontology, we made some modifications to the KnowRob ontology, mainly to reduce its complexity. In general, to obtain an instance of basic human activities, such as *reach*, *take*, and *release*, six node levels are necessary to travel in a KnowRob ontology path, and these classes are included manually in the knowledge base. By contrast, the maximum distance is four node levels in our ontology. Another important improvement in our system is that human activities are inferred and not manually included in the ontology. The new classes are added on demand (see Fig. 6). This means that our basic ontology is optimized because it will grow

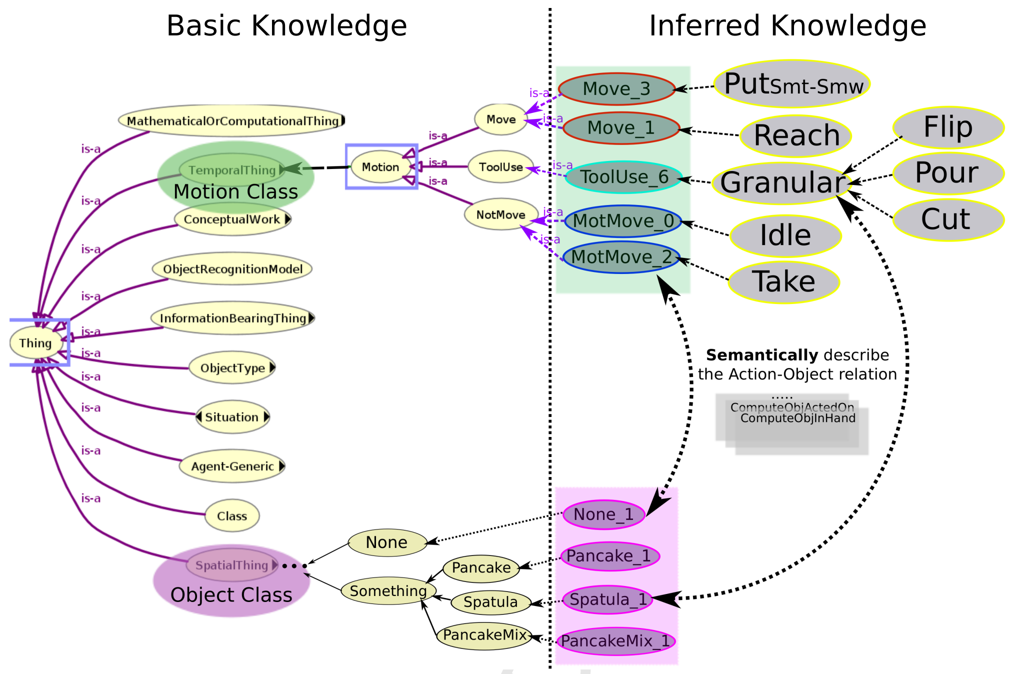


Fig. 6. Example of the semantic descriptions for activities and objects. On the right-handed side, it is possible to observe the dynamic growth of the knowledge base, using the inference mechanism described and implemented in this study. The new nodes in the ontology are created automatically on demand.

automatically as required. In order to facilitate this dynamic growth we need to define new *Computable Properties* and new *Computable Classes*.

The structure of our ontology is mainly defined based on the semantic rules obtained. The human motions are classified as *move, not_move or tool_use* and the inferred activities define new sub-classes on demand. This is very important if we want to predict the intentions of people. For the reasoning engine, the *grounding* aspect is also very important because it defines the connections between the reasoning process and the actual environment. The state of the environment is detected by the sensors in the system (robot), which we emulated using the labeling information stored in a data-base (MySql). We define three *Computable Classes*, i.e., one for each subclass of the class *Motion*. We can then retrieve the generated instances as follows:

$$rdfs\_instance\_of(?InstM, comp\_humAct{:}`\mathbf{Motion}'), \tag{3}$$

where, $?InstM$ is the instance obtained for the class *Motion* and the output is similar to: $?InstM = NotMove_0$, $?InstM = Move_1$, $?InstM = ToolUse_6$, etc. This represents the second step of Algorithm 2. An illustration of the instances obtained is shown in Fig. 6.

The next step is to define two properties of the objects: *ObjectActedOn* and *ObjectInHand*. Hence, we create one *Computable Property* for each and we can then query the properties of the instances as follows:

$$rdf\_triple(comp\_humAct{:}`\mathbf{objectActOn}', ?F, ?V), \tag{4}$$

$$rdf\_triple(comp\_humAct{:}`\mathbf{objectInHand}', ?F, ?V), \tag{5}$$

where *objectActOn* and *objectInHand* represent the target names of the defined *Computable Properties*, $?F$ represents the selected frame, and $?V$ corresponds to the value of the property, which is the name of the object. These queries are equivalent to the first step of Algorithm 2. Possible output values from the query (4) are: $?F = `o - 1'$, $?V = PancakeMix$, which means that for the occurrence 1, *ObjectActedOn* belongs to the class *PancakeMix*. In addition, the output values from the query (5) for the same time instance: $?F = `o - 1'$, $?V = `none'$, which means that for occurrence 1, the value of *ObjectInHand* is *none*. Thus, the hand is *empty* but it is acting on the *pancake mix*.

In order to semantically represent the objects inside our working environment, we need to describe them in a structured and articulated form. This is achieved by obtaining a description with the semantic map, which contains information about the properties and relationships between the objects inside the visual input. These types of information are described using OWL properties, where the objects defined in the semantic map are described as instances of the classes inside the ontology, i.e., each object instance inherits the properties and relationships of its class. For example, *Pancake_1* is an instance of the Class *Pancake* (see Fig. 6), which is defined in the semantic map and it inherits the *motion–object* relationship described in the queries (4) and (5).

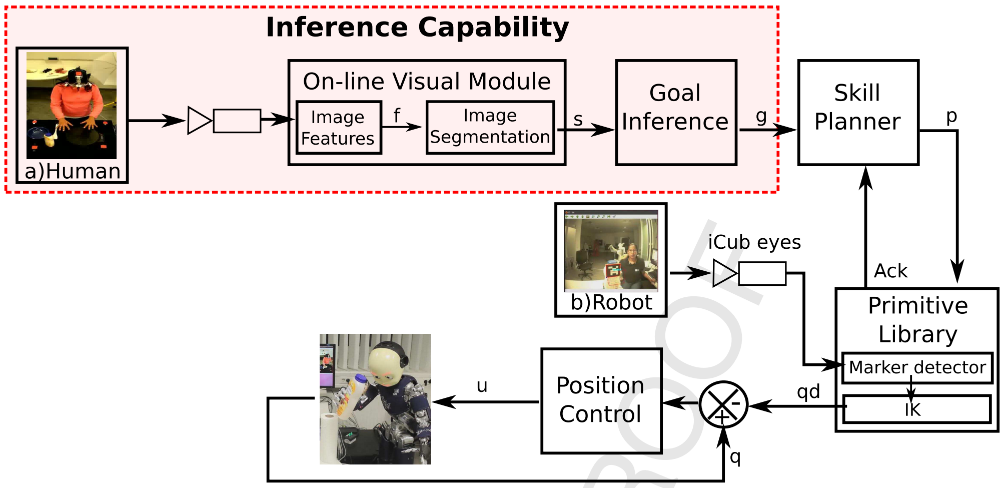


Fig. 7. Integration of the system into the control block of the iCub. This process includes information from external views obtained from videos (a) and environmental information obtain from the camera on the iCub (b). The environmental constraints are added to the desired joints to control the robot's movement. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

Our system is enhanced by adding *knowledge* and *reasoning* sub-modules, which can deal with incomplete information to infer human activities, even when we use semantic rules in an untrained scenario. Fig. 1 depicts the combination of the *semantic* module and its improvement using the *knowledge* and *reasoning* engine. For example, given the perceived *state* of the environment, our system infers the concept value $c(s)$ using the semantic rules. If the system successfully infers the human activity, then this value will define the inferred goal g. In the case of failure, the system will execute the Knowledge Base sub-module and this will generate an equivalent state of the system to infer the correct concept value, where in this case, we refer to this value as the enhanced concept value $c(s_{enhanced})$, which is later used to infer the goal g.

The procedure described above is possible because our system can backtrack the *knowledge-based* ontology until it identifies a similar parental class to that perceived; therefore, inference is possible using this new instance. For example, if we take the sandwich scenario and we assume the perception module recognizes the hand motion $m = tool\_use$ and a new object *pepper* with the object property $o_h(t)$, then for this example, our system will fail to infer the human behavior because the object *pepper* was not defined. However, our system compensates for this failure by searching the *knowledge-base* to find the parental class of this new object, where in the case of *pepper* belongs to the class *Bottle*. This produces a new instance *bottle_1* that represents the equivalent *state* of the system s_i. Using this equivalent state s_i, our system can correctly infer the human activity using the corresponding predicate from the *reasoning engine*. In this example, the possible outputs may be activity = *Pouring* or activity = *Sprinkle*, depending on the value of $o_a(t)$ (see Fig. 12, purple box).

Briefly, the main contributions of this study in terms of the *knowledge-base* and *reasoning engine* are as follows. 1) The description of a new model for the semantic environment. 2) The representation of a new hierarchical structure to define new *Computable Classes* based on the ontology in a meaningful manner. 3) The definition of new *Computables Properties* to extract the properties of an object, such as computeObjectActedOn and computeObjectInHand. 4) The implementation of new prolog predicates to relate the new obtained instances with the object properties, in order to infer human activities (as described in Section 7.2.4).

5. Goal transfer to the robot and execution

We used an iCub, which is a 53-degrees of freedom humanoid robot [52], to experimentally validate our framework (see Fig. 7). The strong humanoid design of the iCub provided an appropriate testing platform to demonstrate the similarities between the observed human motions and the robot's execution. It is important to stress that our system is not limited to a theoretical domain, but instead it provides a functional system that is capable of interacting in real scenarios. This integration was a very challenging task and its solution was not trivial because it required the implementation of interfaces between *high level* control (decision-making modules) and the *low level* control (motion control) to generate a functional system.

This section describes the final module of our system, which involves the integration of the perception and inference modules into the control loop of a robot, where several challenging and difficult problems must be addressed. For example, one important factor is the transition from *offline* learning to *online* learning systems. The perception and semantic modules can be implemented easily for *offline* systems, as we have shown in our previous study [44]. Nevertheless, the inclusion of *offline* systems in a robotic platform is not very useful because they cannot be reactive, which makes the robot very limited in terms of its functionality. Therefore, a perception module is required to work *online*, as described in Section 3.

Algorithm 3 The run() method within the thread class.

```
1: begin CtrlThread::run()
2: activity = inferActivity() {This function segments and infers the observed human activities.}
3: executePrimitive(activity) {This function commands the desired primitive for the iCub robot.}
4: return  The iCub moves its limbs to achieve a similar motion to that observed
```

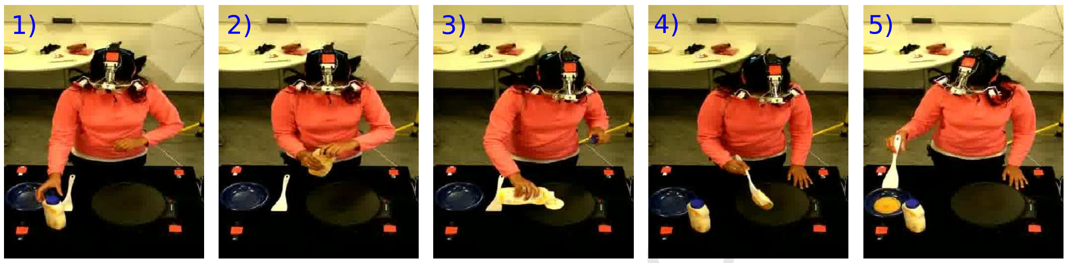


Fig. 8. View used for the *pancake-making* scenario. 1–5) Examples of human activities performed with the right hand: 1) Reach, 2) Hold, 3) Pour, 4) Flip, and 5) Slide_out.

This module is expected to be as fast and accurate as possible within the *control-loop* of any humanoid robot. In other words, the communication between the perception and inference modules must be *online*, as shown in Fig. 7.

Two main modules are implemented within a *thread-loop*: *inferActivity()* and *executePrimitive()*, as shown in Algorithm 3. The function *inferActivity()* segments and interprets the visual data from the video sources (see Fig. 7, red diagram). The red highlighted block in Fig. 7 represents the new capacity that we have implemented in the robot, which is the interpretation of human observations. These capacity will trigger (*online*) the motion primitives (*executePrimitive()*) which the robot needs to execute in order to achieve a goal similar to that observed.

The process *executePrimitive()* (line 3 from Algorithm 3) obtains visual information from the robot's cameras to detect the object(s) in its working area. This information is mapped to the joint space and used as feedback for the control loop. It is important to note that the modular architecture of the framework allows us to replace any module to more complex behaviors acquired, e.g., the vision module can be replaced with a more advanced detection system or the control approach can be replaced by a more robust and adaptive control law, e.g., [53]. The function *executePrimitive()* transforms the observed human behavior into robot motions as follows:

1. We define a plan execution library, which is given the inferred goal (g), can select the primitives (p) from the library repertoire that must be performed by the robot. For example, if the inferred goal is *reaching*, then the execution plan will comprise the following steps: 1) look for the target (o_1), 2) identify the position of o_1, and 3) move the hand toward o_1.
2. From the execution plan, we obtain n-primitives ($p(n)$) which the robot needs to execute. These primitives are retrieved from the *primitive library*. According to this example, the first primitive $p(1)$ will trigger the iCub's gaze controller [54] to look for the object. Next, $p(2)$ will find the object and the IKGaze controller will retrieve the fixation point. Finally, $p(3)$ will allow the Cartesian interface of the robot to control its arm in joint space (q).
3. These steps are then executed until the final step of the execution plan is completed.

6. Examples of everyday human activities

In order to evaluate the robustness of our system under different constraints, we tested three real-world scenarios: making a pancake, making a sandwich, and setting the table. These three tasks have different levels of complexity, because they involve several combinations of the same activities (*reach, take,* etc.) using different objects.

6.1. Making pancakes

In our first scenario, we recorded videos of a human making pancakes. This task was executed nine times by the same subject. The human motions were captured by three cameras located in different positions, but in the evaluation of our framework, we only used the information obtained from camera 2, as shown in Fig. 8. This represents another advantage of the present study compared with our previous study where three views were required as inputs for the system [44]. The pancake task involves objects that could be used as tools, such as the spatula, where these objects are important for defining the *tool use* motion.

6.2. Making a sandwich

In the second scenario, we recorded a more complex activity: making a sandwich. These recordings also contained information obtained from three external cameras. In addition this task was performed by eight randomly selected subjects

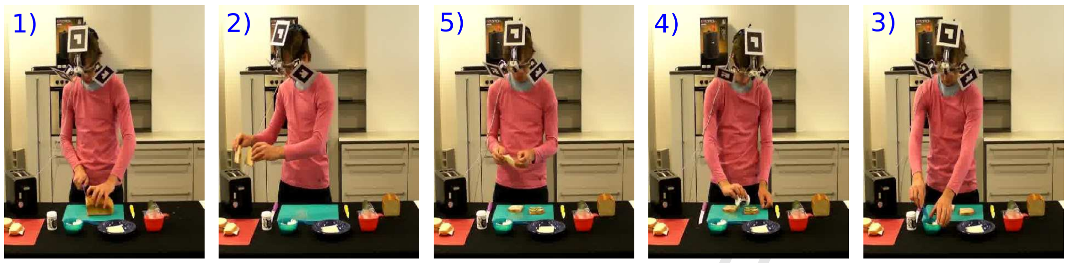


Fig. 9. View of the actions performed when making a sandwich. Some examples of the activities executed with the right hand are: 1) Cut, 2) Put something somewhere, 3) Unwrap 4) Sprinkle, and 5) Spread.

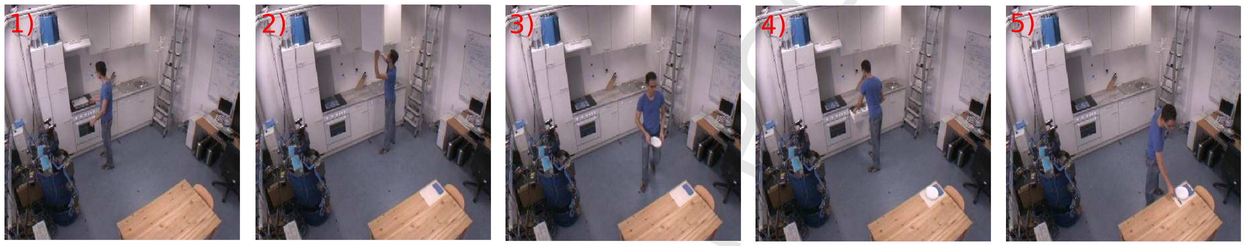


Fig. 10. Subject performing the *setting the table*. Example of activities using the right hand: 1) Take, 2) Reach, 3) Put something somewhere, 4) Open-drawer, and 5) Release.

and each subject prepared approximately 16 sandwiches, where half of the sandwiches were prepared in normal time conditions and the rest under time pressure (in a hurry). Fig. 9 shows that some activities were performed simultaneously using both the right and left hands. For example, the left hand was holding the bread, while the right hand was cutting it with a knife.

6.3. *Setting the table*

The final experimental set-up used videos from the TUM Kitchen Dataset [43], which contains observations of four subjects setting a table at least four times (Fig. 10). The subjects were randomly selected and they performed the actions in a natural manner. Thus, they required no further instructions about how to perform the action. It was possible to observe fluent transitions between sub-actions (task). Furthermore, some tasks were performed in parallel, using both the left and right hands, similar to the scenarios mentioned above.

7. Results

This section presents the results obtained after implementing our system using the iCub robot. It is important to stress that the entire process is performed *online* by the robot, which is a very challenging task to achieve. First, Section 7.1 presents the results obtained by the *real-time* segmentation of 2D videos tested in two different tasks, i.e., *sandwich making* and *pancake making*. Next, Section 7.2 presents the semantic rules, which were obtained *offline* using the *sandwich making* dataset as training. In Section 7.2.1, we describe how the rules obtained were tested *offline* in two new scenarios: *pancake making* and *setting the table*. Section 7.2.2 demonstrates the robustness and re-usability of the tree obtained, using the *sandwich making* dataset under different constraints, e.g., time conditions and labeling strategies. To the best of our knowledge, these types of validations have not been described previously, but they are very important when discussing the *generality* of the results obtained.

After demonstrating that the rules obtained could generalize to the same behaviors in different situations for *offline* datasets, these rules were programmed for an *online* inference process conducted by the robot, and the results are presented in Section 7.2.3. The datasets used for evaluation were a new video (different from the training video) showing the *sandwich making* scenario and a *pancake making* video. Subsequently, Section 7.2.4 demonstrates the enhancement of our system by including the *knowledge-base* and the *reasoning engine*. The results presented in Sections 7.1 and 7.2.3 describe the outcomes of the *inferActivity()* process *online* (step 2 from Algorithm 3). Finally, Section 7.3 describes the experimental execution of the inferred activity by the iCub robot, i.e., the *executePrimitive()* process was performed in *real-time* with the new *pancake making* video.

7.1. *Online segmentation of videos: extracting visual features*

We tested our algorithm (Section 3) using two datasets: *pancake making* and *sandwich making*, as explained in Section 6. The experiments were performed using only a subset of the whole video because our goal was to demonstrate the robot

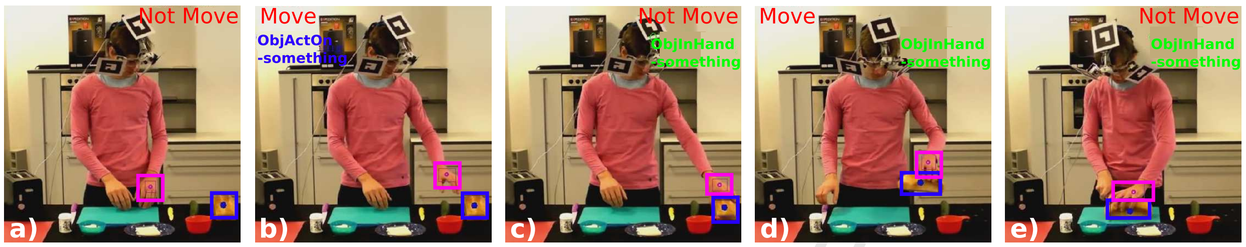


Fig. 11. Results obtained by the perception module after automatically segmenting the *human motions* and the object properties. a) The hand motion was segmented as *not moving* and object recognition lacked any *relevant* information. b) The hand was *moving* and the object had the property $o_a(t) = something$. c) and e) The same segmentation values were used for the hand (*not moving*) and the object had the property $o_h(t) = something$. d) The motion = *move* and the object property $o_h(t) = something$.

could extract simple activities from different environments. For the pancake scenario, we analyzed the video until the *pouring* action was finished. For the sandwich scenario, we split the video until *cutting the bread* was finished. Note, the sandwich scenario had time constraints: normal and fast. It was important to analyze these time conditions because the *lifetime* of similar activities will either be shorter or longer. Some techniques are highly dependent on the execution time of the observed activity, e.g., the ISA algorithm presented by [44]. For example, the *reach* activity in a normal condition required about 39 frames, whereas in the fast condition, the *lifetime* of the same activity decreased to 16 frames. These constraints are very challenging for *real-time* systems. Therefore, we executed our algorithm in these two conditions: normal and fast speeds. The analysis described above was performed for the right and left hands, as shown in Fig. 11.

The videos used for testing had not been manually segmented and the results were obtained by our system *on the fly*. However, because we used a *color-based* algorithm, we had to define some *prior* information such as the colors of objects and the thresholds used ($\varepsilon = 1.5$ and $d(x_h, x_o) = 70$ from Table 1). These values were determined heuristically. The results indicated that the human motions (*move*, *not move* or *tool use*) during *pancake making* were classified correctly in 92.76% of the cases (right hand = 92.60% and left hand = 92.93%). For the *sandwich making* scenario, we found that the average classification success rate in the normal and fast conditions was about 90.76% (right hand = 85.08% and left hand = 96.45%). In addition, the objects in the scene and their properties were correctly identified in 95% of cases for the *pancake making* scenario, but the recognition rate decreased to 85% for the *sandwich making* scenario because the objects in the scene were smaller and they were occluded by larger objects. The results represent the averages obtained after testing at least two input videos in each of the scenarios. This module is a basic visual process and it could be replaced with more sophisticated object recognition algorithms. However, further discussion of perception algorithms is outside the scope of this study.

7.2. Semantic representation results

Weka data mining software was used to generate the decision tree [55] and the *sandwich making* scenario was used as the training dataset. This scenario was selected because among the task examples presented in Section 6, this scenario had the highest task complexity due to the presence of several sub-activities. We divided the procedure into two steps during the training stage.

In the first step, we used the ground-truth data[6] for the first subject in the normal conditions while making a sandwich. We split the data as follows: the first 60% of the trials (instances of the dataset *s*) were used for training and the remaining 40% for testing. Thus, we obtained the tree $T_{sandwich}$ shown at the top of Fig. 12. This learning process captured general information about the objects, motions, and activities. In the top part of the tree obtained (see Fig. 12), it can be seen that human *basic* activities could be inferred as: *idle*, *take*, *release*, *reach*, *put* something somewhere, and *granular*.

It is important to note, the first attribute that must be correctly segmented is the hand motion, e.g., if the hand is *not moving*, we can predict the activity is either *take* or *idle*, which are defined by the object property *ObjectInHand*. Thus, using the tree obtained, we can determine six hypotheses ($H_{sandwich}$), which represent semantic rules that describes the basic human activities. Therefore, the rules obtained define the relationships between the image observations and representations, which are given by each branch in the tree obtained. The taxonomy of the tree represents the syntax of the representations. According to this syntax, we can construct *machine/human-understandable* descriptions of human activities, i.e., *semantics*. Some examples of these rules are:

$$\textit{if Hand}(\textit{Move})\ \&\ \textit{ObjectInHand}(\textit{None})\ \&\ \textit{ObjectActedOn}(\textit{Something}) \rightarrow \textit{Activity}(\textbf{Reach}) \tag{6}$$

$$\textit{if Hand}(\textit{Not\_Move})\ \&\ \textit{ObjectInHand}(\textit{Something}) \rightarrow \textit{Activity}(\textbf{Take}) \tag{7}$$

$$\textit{if Hand}(\textit{Move})\ \&\ \textit{ObjectInHand}(\textit{Something}) \rightarrow \textit{Activity}(\textbf{PutSomethingSomewhere}). \tag{8}$$

[6] The ground-truth data were obtained by manually segmenting the videos into hand motions, object properties, and human activities.

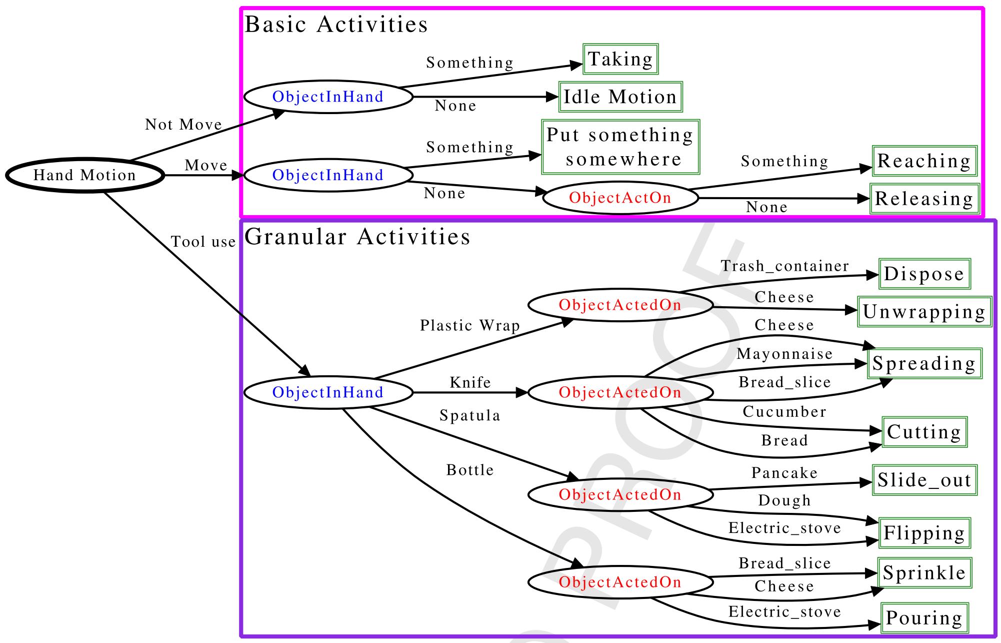


Fig. 12. The top (magenta box) shows the tree obtained from the *sandwich making* scenario ($T_{sandwich}$). The bottom (purple box) shows the extension of the tree, which represents the second stage of the learning process. In this branch of the tree, the names of the objects are used because the current context defines the activity that is executed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

For the *sandwich making* dataset, the expected activities are *cut*, *sprinkle*, *spread*, etc. However, in the first-step method, these activities were always clustered into the same class *spread*[7]:

$$\textit{if } Hand(Tool\_use) \rightarrow Activity(\mathbf{Spread}) \tag{9}$$

Thus, the C4.5 algorithm was not capable of correctly separating the *complex/granular* activities into different rules, so the second step proposed in Section 4.1.2 was required to classify these *granular* activities correctly. To achieve this, all of the complex activities in the input dataset (s) were replaced with the label of *Granular* and they were inferred with the following rule.

$$\textit{if } Hand(Tool\_use) \rightarrow Activity(\mathbf{Granular}) \tag{10}$$

Next, all of the activities classified by this rule, formed the input for the second learning step in our system. The final tree is shown in Fig. 12, where the top part (magenta box) represents the general and most abstract level of rules for determining different *basic* activities and the bottom part (purple box) represents the extension of the tree, which considers the current information related to objects to recognize *granular* activities correctly. Thus, in order to identify which *granular* activity is being executed, we need to know which objects (or classes) are identified in the scene. Some examples of the extension of the tree are as follows.

$$\textit{if } Hand(Tool\_use) \,\&\, ObjectInHand(Knife) \,\&\, ObjectActedOn(Bread) \rightarrow Activity(\mathbf{Cut}) \tag{11}$$

$$\textit{if } Hand(Tool\_use) \,\&\, ObjectInHand(Bottle) \,\&\, ObjectActedOn(Bread\_slice) \rightarrow Activity(\mathbf{Sprinkle}) \tag{12}$$

We can see that the second rule has *ObjectInHand* with the value *Bottle*, which shows that the parental class of the object was identified instead of the object itself, i.e., *pepper* $\in$ *Bottle*. This will be explained later when we describe the knowledge and reasoning results. The next step was to test the accuracy of the tree obtained $T_{sandwich}$, where we used the remaining 40% of the sandwich dataset to test the accuracy of the rules obtained. Thus, we executed steps 4–8 from Algorithm 1 to determine $c(n_{sandwich\_test})$, i.e., given the input attributes $n_{sandwich\_test} = \{Move, \ Something, \ None\}$, we determined $c(n_{sandwich\_test})$. The *state-value* pairs from the test dataset $n_{sandwich\_test}$ were in form of $\langle n_{sandwich\_test}(t), ?\rangle$, where t represents the time (*frames*). Next, the target value was determined for each state of the system $c(n_{sandwich\_test}(t))$. Finally, the results obtained showed that $c(n_{sandwich\_test}(t))$ was classified correctly in 92.57% of the cases, compared to the ground-truth data. To analyze the results, we obtained the confusion matrix shown in Table 2a, where the main diagonal of the table indicates

[7] The *spread* activity had the most instances in the training dataset, so most of the instances were correctly classified when this activity was selected.

Table 2
Confusion matrix (expressed in %) obtained from the *sandwich making*, *setting the table*, and *pancake making* scenarios, where a = Reach, b = Take, c = Put something somewhere, d = Release, e = Granular, and f = Idle.

Actual class	Classified as																	
	a) Sandwich						b) Set the table						c) Pancake					
	a	b	c	d	e	f	a	b	c	d	e	f	a	b	c	d	e	f
a	**86.9**	3.4	3.4	0.6	0	5.4	**86.9**	0	10.4	1.7	0.8	0	**94.2**	3.8	0	0	0	1.9
b	2.6	**76**	8	13.3	0	0	5.5	**44.4**	38.8	11.1	0	0	0	**93.1**	6.8	0	0	0
c	0.8	0.3	**91.1**	1.1	3.5	2.8	0	0	**99.2**	0.4	0	0.2	1.7	3.1	**92.3**	0	1.3	1.3
d	7.2	0.5	1	**46.3**	0	44.8	0	0	3.4	**63.9**	0	32.5	4.4	0	4.4	**68.8**	0	22.2
e	0	0	1.2	0	**98.7**	0	0	0	0	0	**100**	0	0	0	4.5	0	**95.4**	0
f	2.6	0	0	2.3	0	**94.9**	1.6	0	0	4.2	0	**94.1**	0.1	0	0	0.7	0	**99.1**

that human activities were classified correctly in most cases. Therefore, the semantic rules obtained from the tree $T_{sandwich}$ generalized the human activities very well for the sandwich scenario.

7.2.1. Generalization to different scenarios

The next challenge was to test the semantic rules in unknown scenarios, where we used the set of hypotheses $H_{sandwich}$ to infer human activities in new situations, and thus we did not generate new hypotheses to infer human *basic* activities. Two new scenarios were used for testing: *pancake making* and *setting the table* using manually annotated labels. In Section 7.2.3, we present the results obtained using the automatic segmentation method from the vision module as inputs. The terminologies for these new scenarios are $n_{pancake}$ and $n_{setTable}$, where we aimed to obtain the target values $c(n_{pancake}(t))$ and $c(n_{setTable}(t))$. First, we entered the data from $n_{setTable}$ into our Algorithm 1 and performed steps 4–8, which showed that $c(n_{setTable})$ was classified correctly in 91.53% of cases, and the corresponding confusion matrix is shown in Table 2b. This table shows the best results are along the main diagonal, which indicates that most of the instances were classified correctly. However, we can also see that the off-diagonal elements in this table are better than the results obtained for the *sandwich making* scenario (see Table 2a). For example, for the activity *Taking*, 38.8% of the instances were classified incorrectly as the activity *PutSomethingSomewhere*. From the tree obtained, $T_{sandwich}$ (see Fig. 12), we can see that the difference between these two activities was the value of the attribute *Hand_Motion*, which in this case was labeled incorrectly as *Move*, and thus the activity was classified incorrectly as *PutSomethingSomewhere*.

The second new scenario $n_{pancake}$ was tested and we found that $c(n_{pancake})$ was classified correctly 97.15% of the cases (Table 2c). These results indicate that the tree obtained, $T_{sandwich}$, could generalize the definition of human basic activities in kitchen scenarios. We should stress that a similar tree was obtained when we used the labeled information obtained from the *setting the table* or *pancake making* actions as training set, as shown in [56]. Therefore, the tree obtained extracted the meaning of the basic human activities in a general form.

7.2.2. Testing the robustness of the semantic rules

The previous results were obtained using the label information when one human performed different activities with the right hand. Thus, the labeling process was executed by one expert user. In order to test the robustness of the hypothesis obtained, $H_{sandwich}$, we conducted more experiments and analyzed the results, so the following two experiments were performed.

1. Different conditions and people using both hands. In this experiment, we tested the variation in possible styles when performing the same activity by analyzing the activities performed by several people (males and females) and by constraining the time conditions during the execution of these activities. This analysis used the *sandwich making* scenario because it had more subjects and different time constraints, as described in Section 6.2.

In this experiment, eight people who were not involved with our project were instructed to prepare a sandwich. Half of the subjects were females and the other half were males (Fig. 13). Each subject prepared the sandwiches under two conditions: normal and fast speeds. Fig. 13 shows that one of the male subjects (S8) was left-handed, which implies the *granular activities* (cut, sprinkle, etc.) were performed using the left hand instead of the right hand, in contrast to the other subjects. We should also note that the meaning of the *sandwich making* task was not interpreted in the same way by all of the subjects. For instance, subject 3 prepared the sandwich with one bread slice (a half sandwich without toasting the bread). The following results demonstrate that our system robustly identified the activities performed even when there were variations.

In general, we found that each person had their own style and interpretation of the required task, i.e., preparing a sandwich. Thus, the following two questions must be considered: *is there a general principle followed by humans when executing basic activities? and do humans interpret the motions of other humans in the same way?*

In order to answer the first question, we performed the following experiment. A different person from those who prepared the sandwiches manually labeled the attributes of the human motions and the object properties. This was performed

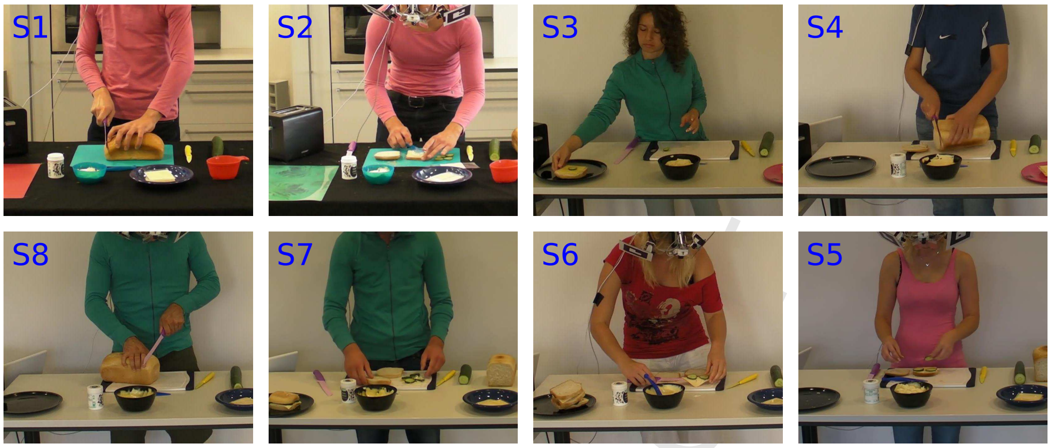


Fig. 13. Different styles of preparing a sandwich. Subjects 1, 2, 7, and 8 were males and the remainder were females.

by each of the eight subjects, in the two conditions, and for each hand; therefore, 32 videos were labeled in total[8] by the same person. We then used this information to test our obtained hypothesis $H_{sandwich}$. The results suggested that although the subjects performed the task in different ways, the semantic rules obtained were still valid with a high accuracy of greater than 90%. Furthermore, the rules were not affected when the subject executed the task in the normal (94.75%) or fast (95.02%) conditions. These rules were also not gender-dependent (females = 95.6%, males = 93.85%). These results demonstrate that we all follow similar principles when performing a task, thereby answering the first question. Moreover, the rules obtained were also valid when the motions of the left hand were analyzed by the expert who labeled the videos. The average accuracy was 95.05% in this case. To answer the second question, we performed the following experiment.

2. Labeling strategies of different subjects: In the previous experiment, only one expert person labeled 32 videos. The next challenge was to test whether the results obtained were affected when the labeling process was performed by different people. The goal was to determine whether the activities performed by other people were perceived in different ways and if this was reflected in the labeling process. We expected that people would use different labeling strategies to segment human activities, but that the rules obtained would be the same.

Therefore, we performed an experiment where we asked four students,[9] who were not involved with the project, to label four videos of the same subject (S1): one in normal speed condition with the right hand, the second in the same condition with the left hand, and the other two in the fast condition with the left and right hands. Note that the students who labeled the videos received no instructions.

The results are presented in Fig. 14, which show that accuracy of recognition decreased to 74.62% in the normal condition and to 78.75% in the fast condition. These two percentages represent the worst results, but it is interesting that both results corresponded to labels for the right hand. This lower accuracy may have occurred because we do not always perceive when another person stops their motion, which indicates that a robot is not expected to automatically segment these motions with 100% accuracy, but instead they should perform in the same manner as humans. In addition, we found that the labeling strategy employed by females and males was fairly similar, as we expected.

The results obtained after testing human activities performed by both hands demonstrated our comprehensive approach. We found that rules obtained using the right hand ($H_{sandwich}$) were also valid for inferring the activities performed by humans with the left hand. Therefore, the obtained semantic rules were independent of the hand performing the activity because the meaning was the same.

7.2.3. *Online recognition by the iCub*

In the next step, we tested the information obtained from module 1 (see Section 7.1), i.e., from the automatic segmentation of human motions and object properties. The results after employing the semantic rules as the input for the *online* perceived information in the *pancake making* scenario were accurate in 91.32% of cases (right hand = 89.39% and left hand = 93.25%) and in 83.57% of cases in the *sandwich making* scenario (right hand = 75.75% and left hand = 91.39%). The errors in activity recognition occurred because objects were misclassified by the perception module, especially for the right hand in the *sandwich making* scenario when the *knife* was occluded by the hand and the bread. However, it is clear that the results obtained were better than those when the non-experts labeled the data (see Fig. 14).

These results demonstrate that the definitions of rules allowed the inference of basic human activities in different scenarios with high accuracy. Fig. 14 presents the results obtained under all of the test constraints, where the accuracy ranged

[8] These data labels were then treated as the ground-truth data and the person who labeled these data was considered an expert because they experienced a training stage.

[9] The educational levels of the students ranged from high school to master students, where two students were females and the rest were males.

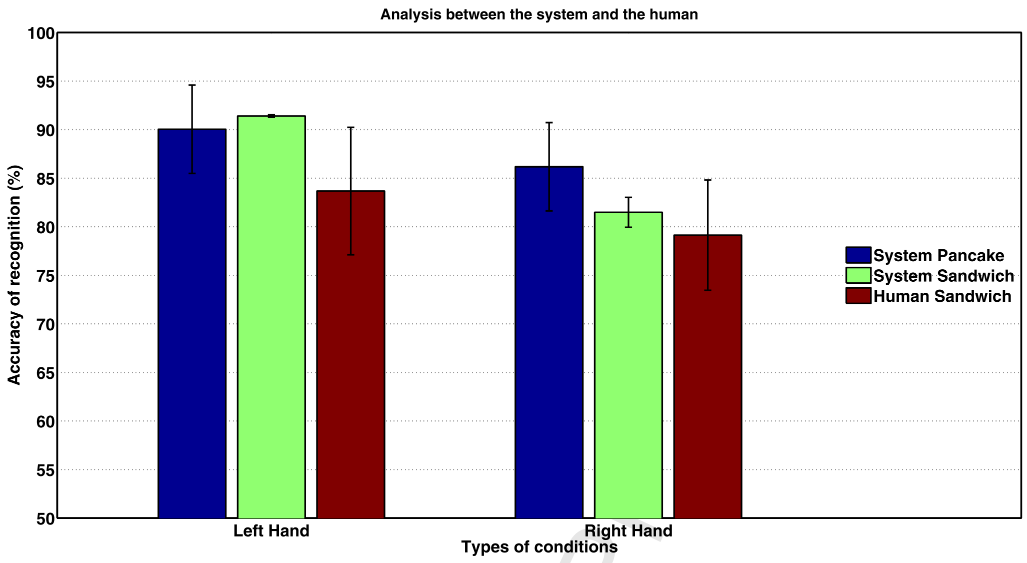


Fig. 14. Results obtained by the system (the robot) and human under several labeling constraints. The results show that recognition was performed more accurately by the system than the non-expert human.

from 74.62% when external people labeled the data up to 93.25% when the system used very simple hand and object recognition to automatically segment the motions and object properties. These results represent the first step toward the generalization of these types of activities.

7.2.4. Knowledge base and reasoning engine results

In the previous subsections, we demonstrated that semantic rules obtained for human activities did not change under different constraints, e.g., time constraints, different scenarios, gender, or the hand used to execute the activity. In this subsection, we present the results obtained when we used the semantic rules to improve our reasoning engine and the knowledge base.

A very important characteristic of our system is that we can define the structure of our ontology in a dynamic form using the semantic rules obtained, which facilitates the prediction of human behaviors. For example, if the activity is a sub-class of *not_move*, then the human activity will either be *take* or *idle* according to the semantic rules (see the rules in Fig. 12).

The next step was to semantically relate the instances from the class *Motion* and the object properties described previously in *Computables* (3), (4), and (5), which was achieved by using the rules obtained in Section 7.2. For example, to infer the activity *Take*, we define the following *prolog predicate*:

$$
\begin{aligned}
&humanAct(?Occ,take):-\\
&\quad rdfs\_instance\_of(?InstM, comp\_humAct\text{:`}\mathbf{Motion}\text{'}),\\
&\quad InstM = \text{`}NotMove\text{'},\\
&\quad rdf\_triple(comp\_humAct\text{:`}\mathbf{objectInHand}\text{'}, Occ, ?V),\\
&\quad (V = \text{`}Something\text{'}; V \backslash = \text{`}none\text{'}),
\end{aligned}
$$

where ?*Occ* is the occurrence number that we want to infer and *take* is the name of the inferred class. From the predicate above, we can see how the instances of the class *Motion* and the objects with the property of *ObjectInHand* are semantically described and represented. This exemplifies the last step of our Algorithm 2. Similar *prolog predicates* are defined for the remaining rules. An illustration of the growth of the knowledge base due to the instances inferred from the relationship computed between the *Motions* and the object *properties* is shown at the right of Fig. 6.

With our system, we could ask the following query: *humanAct*(?*Occ*, ?*Activity*), when the input value of $Occ = 1$, then the output for the *inferred* activity is ?*Activity* = *Reach*, i.e., *humanAct*('1', ?*Activity*) = '*Reach*'. In this case, the instance of *PancakeMix_1* had the property of *ObjectActedOn*, whereas none of the objects had the property of *ObjectInHand*. Thus, when a *reaching* activity was identified, we first moved toward the object that we wanted to manipulate, but we did not have the object in our hands yet. Similar results were obtained for the remaining queries.

Another advantage of our system is the possibility of handling missing data. For example, we could submit the same query to the system where both of the variables are unknown, i.e., *humanAct*(?*Occ*, ?*Activity*). Thus, the output would be similar to $Occ = 2$ and ?*Activity* = *Take*, or $Occ = 3$ and ?*Activity* = *Put*, or ?$Occ = 11$?*Activity* = *Release*, etc. The properties of the objects would also be retrieved. We also found that if we inferred the activities performed in consecutive instances,

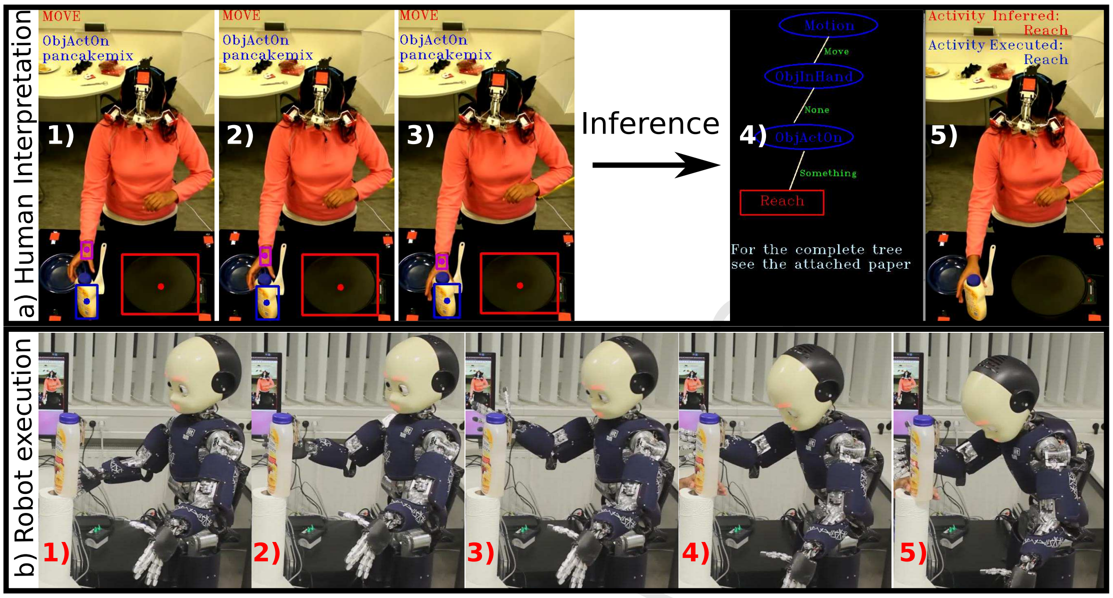


Fig. 15. Reaching activity performed by the human, which was inferred and executed by the iCub robot.

these activities followed a pattern that could define the plan which the human followed during the execution of the final task. Further analysis is performed on this subject, which is outside the scope of the current study.

7.3. Execution of the iCub in real-time

Several experiments were performed to validate and evaluate the proposed method using a humanoid robot in realistic scenarios. To illustrate the different contributions of this study, we present the results obtained by the proposed framework for the *pancake making* scenario, using the rules generated from the *sandwich making* scenario. As mentioned in Section 1.1, this system comprises three main subsystems, 1) image observation and segmentation, 2) interpretation of human intentions and 3) activity imitation by the robot. These subsystems were implemented in an iCub following the control block shown in Fig. 7, where the results of these subsystems were as follows.

1. **Extracting relevant data from videos:** The pancake video was played and the perception module (see Figs. 15.a.1–3)) retrieved the recognized *human motions* and the object properties (o_a or o_h). In this case, motion $=$ *move* and $objectActedOn(o_a(t)) = pancakemix$.
2. **Interpreting human intentions:** After the relevant information was recognized, the *goal* of the human was inferred. Thus, one branch of the tree obtained from Fig. 12 (see Section 7.2) was executed as shown in Fig. 15.a.4), i.e., the human activity inferred by the system, i.e., $activity = Reach$, was sent to the robot (see Fig. 15.a.5)).
3. **Skill execution by the robot:** Based on the inferred activity, a module selected the execution plan that needed to be performed. The plan indicated the motion primitives that the robot had to execute in order to achieve a similar goal to the human. If the inferred activity was *reaching*, then a position-based visual servoing module was executed to reach the current position of the *pancake mix*[10] in the robot's environment. This module extracted 2D visual (image) features from a stereo vision system with Augmented Reality (AR) markers. We used the ArUco library to detect the AR markers, which is based on OpenCV. The 3D position and rotation with respect to the camera frame (iCub right eye) were obtained from the image features, using the camera's intrinsic parameters. When the marker was detected, the next primitive consists of moving the right arm of the robot toward the desired Cartesian position, which was achieved by using the inverse kinematics, as shown in Fig. 15b). In addition to pick and place activities, such as reach, take, put something somewhere, and release, our system can handle more specific activities such as pouring (see Fig. 16). These motion primitives followed a similar procedure to that described above.

In summary, during *run-time*, the robot could correctly segment and infer the human behaviors involved in the *sandwich making* scenario with an average accuracy of 83.57%. This high accuracy was expected, since this was the dataset used, to learn the decision tree, using ground-truth data. However, when a new video was shown to the robot, for which it

[10] When the *reaching* activity was inferred, the *ObjectActedOn* value observed in the video was used as a target object for the robot. If an *ObjectInHand* property was detected, this determined the type of object that the robot needed to grasp.

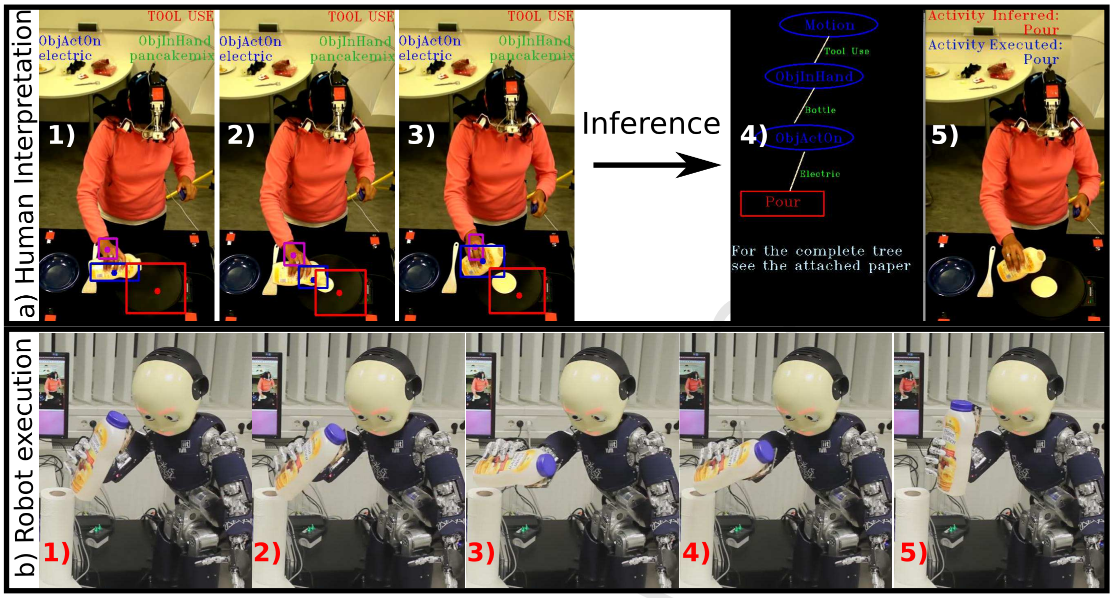


Fig. 16. Pouring activity performed by a human, and that inferred and executed by the iCub.

had not received training (see Figs. 15 and 16), the recognition accuracy in this new scenario for *online* segmentation and recognition was 91.32% when recognizing *basic* activities compared to the ground-truth, where the *pouring* activity was classified as *granular* using the rule from Eq. (10). The accuracy was higher than the *sandwich making* scenario because the system only considered the recognition of *basic* behavior and the sandwich dataset had produced more errors due to the use of the *color-based* technique during the classification of objects, e.g., the *knife* was sometimes occluded; therefore, the properties were not obtained correctly, thereby leading to misclassified activities. The higher accuracy of the results in the new *pancake making* scenario were not due to an overfitting problem because the data from this new task were not used for training.

The results presented in Sections 7.2.1 and 7.2.2 were obtained by *offline* recognition using ground-truth information. However, the presentation of these results is highly relevant for explaining how the proposed method can generalize and the possibility of implementing the results obtained in *online* systems, which was also addressed in the present study. In particular, the results in Section 7.2.2 required the segmentation of the ground-truth by an expert, where the test used random people to segment the data. Similarly, Koppula and Saxena [23] provided an interesting analysis of the ambiguity in temporal segmentation, where they considered various types of segmentation and they concluded that their proposed segmentation method was the closest segmentation to the ground-truth; however, no further analysis has made to include different segmentation performed by random people. Therefore, our findings complement these results by including this missing analysis. Our results demonstrate the segmentation of human motions might not always be the same due to the fact that different people have their own strategies for segmenting human behaviors. Therefore, we cannot expect 100% accuracy when an artificial system performs segmentation automatically. The results suggest that the worst case scenario should be similar to that achieved by random people when segmenting motions (around 76.68% accuracy). Thus, the results in Section 7.2.3 were better than expected (average accuracy of 87.44%) when using the simplest color-based algorithm to enhance our semantic representations.

Testing algorithms in new scenarios has several challenges such as recognition of new behaviors, re-usability of models, and accuracy. Our system can handle most of these challenges, but one limitation is the recognition of a new activity, which is always clustered as *granular* in the present study. Thus, in future research we will analyze mechanisms for learning new behaviors *on-demand*. Our first attempt to solve this issue by including memory and an *active-learning* module was presented in our recent study [57]. However, the inclusion of the *knowledge-based* ontology proposed in the present study will also help to control the growth of the decision tree, but this is still under investigation. A video that provides more details of the experimental results can be found via the following link: http://web.ics.ei.tum.de/~karinne/Videos/AIJ13ramirezK.mp4.

8. Conclusions

Correctly identifying human activities is a challenging task for the robotics community, but its solution is very important due to it is the first step toward more natural human–robot interaction. In this study we proposed a method for extracting the meaning of basic human activities by combining hand motion information and two object properties. This information is used as an input by our framework and the output is the inferred human activity, which can be executed by a robotic system in *real-time*.

We described the proposed method for generating semantic rules from human observations, which we tested under different constraints. Overall, the results suggest that the best accuracy was obtained by our system with an accuracy classification of 92%, whereas the worst was obtained by a human labeling another human with 74.62%. This demonstrates that our approach can find rules to generalize basic human activities. Therefore, we developed a new (meaningful) method that allows the extraction of semantic representations, which can be used to transfer skills to humanoid robots.

By adding new capabilities to the reasoning engine, we showed that it could compute new relationships between objects and activities, thereby improving the dynamic growth of the knowledge base in a meaningful manner, which is necessary because we cannot guarantee that knowledge stored manually in the system will be valid under different constraints/scenarios. We found that the activities followed predefined plans and that these plans could be obtained from the semantic representations proposed in the present study. Another important characteristic of our system is that inference rules obtained can be implemented in systems such as humanoid robots to allow more natural interactions with human. Therefore, these semantic rules could enhance the planning process for robotic systems.

8.1. Contributions

The main contributions to this study are summarized as follows.

- Proposal and implementation of a multilevel framework that allows the flexible imitation of human behaviors based on observations.
- Definition of a new method for obtaining semantic rules that capture the *essence* of human activities.
- The dynamic growth of the ontology-based knowledge representation was improved by using the semantic rules in the reasoning engine.

Acknowledgements

K. Ramirez-Amaro was supported by a CONACYT–DAAD scholarship. This study received funding from the European Community Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 609206 and it was supported (in part) by the DFG cluster of excellence *Cognition for Technical systems CoTeSys* and by the ICT Call 7 ROBOHOW.COG (FP7-ICT) under grant agreement no. 288533.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.artint.2015.08.009.

References

[1] S. Schaal, Is imitation learning the route to humanoid robots?, Trends Cogn. Sci. 3 (6) (1999) 233–242.
[2] C.L. Nehaniv, K. Dautenhahn (Eds.), Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions, Cambridge Univ. Press, Cambridge, 2007.
[3] M. Carpenter, J. Call, The question of what to imitate: inferring goals and intentions from demonstrations, in: K. Dautenhahn, C.L. Nehaniv (Eds.), Imitation and Social Learning in Robots, Humans and Animals, MIT Press, 2007.
[4] C. Nehaniv, K. Dautenhahn, The correspondence problem, in: K. Dautenhahn, C. Nehaniv (Eds.), Imitation in Animals and Artifacts, MIT Press, 2002, pp. 41–61.
[5] P.K. Turaga, R. Chellappa, V.S. Subrahmanian, O. Udrea, Machine recognition of human activities: a survey, IEEE Trans. Circuits Syst. Video Technol. 18 (11) (2008) 1473–1488.
[6] S. Park, J. Aggarwal, Semantic-level understanding of human actions and interactions using event hierarchy, in: 2004 Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '04, 2004, p. 12.
[7] R. Poppe, A survey on vision-based human action recognition, Image Vis. Comput. 28 (6) (2010) 976–990.
[8] Q.V. Le, W.Y. Zou, S.Y. Yeung, A.Y. Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, in: CVPR, IEEE, 2011, pp. 3361–3368.
[9] D.J. Patterson, D. Fox, H.A. Kautz, M. Philipose, Fine-grained activity recognition by aggregating abstract object usage, in: ISWC, IEEE Computer Society, 2005, pp. 44–51.
[10] J.K. Aggarwal, M.S. Ryoo, Human activity analysis: a review, ACM Comput. Surv. 43 (3) (2011) 16.
[11] M. Beetz, M. Tenorth, D. Jain, J. Bandouch, Towards automated models of activities of daily life, Technol. Disabil. 22 (2010) 27–40.
[12] H.S. Koppula, R. Gupta, A. Saxena, Learning human activities and object affordances from RGB-D videos, CoRR, arXiv:1210.1207.
[13] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, B. Porr, Cognitive agents – a procedural perspective relying on the predictability of Object–Action-Complexes (OACs), Robot. Auton. Syst. 57 (4) (2009) 420–432.
[14] E.E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, F. Wörgötter, Learning the semantics of object–action relations by observation, Int. J. Robot. Res. 30 (10) (2011) 1229–1249.
[15] D. Vernon, G. Metta, G. Sandini, A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents, IEEE Trans. Evol. Comput. 11 (2) (2007) 151–180.
[16] Y. Kuniyoshi, The science of imitation – towards physically and socially grounded intelligence, in: Special Issue TR-94001, Real World Computing Project Joint Symposium, Tsukuba-shi, Ibaraki-ken, 1994, pp. 95–96.
[17] T. Inamura, T. Shibata, Geometric proto-symbol manipulation towards language-based motion pattern synthesis and recognition, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, 2008, pp. 334–339.
[18] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Survey: robot programming by demonstration, in: Handbook of Robotics, 2008.

[19] F. Hermens, S. Gielen, Posture-based or trajectory-based movement planning: a comparison of direct and indirect pointing movements, Exp. Brain Res. 159 (3) (2004) 340–348.
[20] C. Schuldt, I. Laptev, B. Caputo, Recognizing human actions: a local SVM approach, in: 2004 Proceedings of the 17th International Conference on Pattern Recognition, vol. 3, ICPR 2004, IEEE, 2004, pp. 32–36.
[21] T. Kanda, D. Glas, M. Shiomi, N. Hagita, Abstracting people 's trajectories for social robots to proactively approach customers, IEEE Trans. Robot. 25 (6) (2009) 1382–1396.
[22] H.S. Koppula, A. Saxena, Anticipating human activities using object affordances for reactive robotic response, in: Robotics: Science and Systems, RSS 2013, 2013.
[23] H.S. Koppula, A. Saxena, Learning spatio-temporal structure from RGB-D videos for human activity detection and anticipation, in: JMLR Proceedings, JMLR.org, vol. 28, ICML 13, 2013, pp. 792–800.
[24] J. Lei, X. Ren, D. Fox, Fine-grained kitchen activity recognition using RGB-D, in: The 14th International Conference on Ubiquitous Computing, Ubicomp 2012, 2012.
[25] C. Atkeson, S. Schaal, Robot learning from demonstration, in: Proc. Int. Conf. on Machine Learning, 1997, pp. 12–20.
[26] S. Calinon, A. Billard, Incremental learning of gestures by imitation in a humanoid robot, in: ACM/IEEE International Conference on Human–Robot Interaction, ACM, 2007, pp. 255–262.
[27] A. Billard, Y. Epars, S. Calinon, G. Cheng, S. Schaal, Discovering optimal imitation strategies, Robot. Auton. Syst. 47 (2–3) (2004) 67–77.
[28] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Trans. Robot. 26 (5) (2010) 800–815.
[29] A.J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: ICRA, IEEE, 2002, pp. 1398–1403.
[30] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, M. Beetz, Imitating human reaching motions using physically inspired optimization principles, in: Humanoids, IEEE, 2011, pp. 602–607.
[31] D. Nyga, M. Tenorth, M. Beetz, How-models of human reaching movements in the context of everyday manipulation activities, in: ICRA, IEEE, 2011, pp. 6221–6226.
[32] D. Wolpert, Z. Ghahramani, Computational principles of movement neuroscience, Nat. Neurosci. Suppl. 3 (2000) 1212–1217.
[33] J. Bandouch, F. Engstler, M. Beetz, Accurate human motion capture using an ergonomics-based anthropometric human model, in: Fifth International Conference on Articulated Motion and Deformable Objects, 2008.
[34] Y. Kuniyoshi, M. Inaba, H. Inoue, Learning by watching: extracting reusable task knowledge from visual observation of human performance, IEEE Trans. Robot. Autom. 10 (6) (1994) 799–822.
[35] R. Jäkel, S.R. Schmidt-Rohr, M. Lösch, R. Dillmann, Representation and constrained planning of manipulation strategies in the context of programming by demonstration, in: ICRA, IEEE, 2010, pp. 162–169.
[36] A. Fern, J.M. Siskind, R. Givan, Learning temporal, relational, force-dynamic event definitions from video, in: R. Dechter, R.S. Sutton (Eds.), AAAI/IAAI, AAAI Press/The MIT Press, 2002, pp. 159–166.
[37] W. Takano, Y. Nakamura, Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation, in: 2006 6th IEEE–RAS International Conference on Humanoid Robots, IEEE, 2006, pp. 425–431.
[38] M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter, R. Dillmann, Action sequence reproduction based on automatic segmentation and object–action complexes, in: IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2013.
[39] D. Gehrig, P. Krauthausen, L. Rybok, H. Kuehne, U.D. Hanebeck, T. Schultz, R. Stiefelhagen, Combined intention, activity, and motion recognition for a humanoid household robot, in: IROS, IEEE, 2011, pp. 4819–4825.
[40] Y. Yang, C. Fermüller, Y. Aloimonos, Detection of manipulation action consequences (MAC), in: CVPR, IEEE, 2013, pp. 2563–2570.
[41] A. Guha, Y. Yang, C. Fermuuller, Y. Aloimonos, Minimalist plans for interpreting manipulation actions, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2013, pp. 5908–5914.
[42] H.A. Kautz, H.A. Kautz, R.N. Pelavin, J.D. Tenenberg, M. Kaufmann, A formal theory of plan recognition and its implementation, in: Reasoning About Plans, Morgan Kaufmann, 1991, pp. 69–125.
[43] M. Tenorth, M. Beetz, KnowRob: a knowledge processing infrastructure for cognition-enabled robots, Int. J. Robot. Res. 32 (5) (2013) 566–590.
[44] K. Ramirez-Amaro, E.-S. Kim, J. Kim, B.-T. Zhang, M. Beetz, G. Cheng, Enhancing human action recognition through spatio-temporal feature learning and semantic rules, in: 2013 13th IEEE–RAS International Conference on Humanoid Robots, 2013.
[45] G.R. Bradski, A. Kaehler, Learning OpenCV – Computer Vision with the OpenCV Library: Software That Sees, O'Reilly, 2008.
[46] K. Ramirez-Amaro, M. Beetz, G. Cheng, Automatic segmentation and recognition of human activities from observation based on semantic reasoning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014, IEEE, 2014.
[47] T.M. Mitchell, Machine Learning, McGraw–Hill, 1997.
[48] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[49] S.J. Russell, P. Norving, Artificial Intelligence: A Modern Approach, Prentice–Hall, Upper Saddle River, New Jersey, 1995.
[50] J. Wielemaker, Z. Huang, L. van der Meij, SWI-prolog and the web, Theory Pract. Log. Program. 8 (3) (2008) 363–392.
[51] V. Vassiliadis, J. Wielemaker, C. Mungall, Processing OWL2 ontologies using Thea: an application of logic programming, in: R. Hoekstra, P.F. Patel-Schneider (Eds.), CEUR Workshop Proceedings, CEUR-WS.org, vol. 529, OWLED, 2008.
[52] G. Metta, G. Sandini, D. Vernon, L. Natale, F. Nori, The iCub humanoid robot: an open platform for research in embodied cognition, in: PerMIS: Performance Metrics for Intelligent Systems Workshop, 2008, pp. 19–21.
[53] E. Dean-León, V. Parra-Vega, A. Espinosa-Romero, J. Fierro, Dynamical image-based PID uncalibrated visual servoing with fixed camera for tracking of planar robots with a heuristical predictor, in: 2nd IEEE International Conference on Industrial Informatics, 2004, INDIN'04, IEEE, 2004, pp. 339–345.
[54] U. Pattacini, Modular Cartesian controllers for humanoid robots: design and implementation on the iCub, Ph.D. thesis, 2010.
[55] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[56] K. Ramirez-Amaro, M. Beetz, G. Cheng, Extracting semantic rules from human observations, in: ICRA'13 Workshop: Semantics, Identification and Control of Robot–Human–Environment Interaction, 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013.
[57] K. Ramirez-Amaro, M. Beetz, G. Cheng, Understanding the intention of human activities through semantic perception: observation, understanding and execution on a humanoid robot, Adv. Robot. 29 (5) (2015) 345–362.

XMLVIEW: extended

Appendix A. Supplementary material

The following is the Supplementary material related to this article.

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: MMC 1
caption: The following video provides more details about the proposed method for extracting the meaning of basic human activities by combining hand motion information and two object properties. Furthermore, this video shows the real-time results of a robotic system to infer human activities using the proposed method.
link: **VIDEO : mmc1**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

Sponsor names

Do not correct this page. Please mark corrections to sponsor names and grant numbers in the main text.

Seventh Framework Programme, *country*=Belgium, *grants*=609206

DFG, *country*=Germany, *grants*=