Towards finding semantic inter-object relations

handed in
MASTER'S THESIS

B. Eng. Simon Winkler

born on the 15.11.1995
living in:
Brunnenweg 2
84180 Loiching
Tel.: 0151 - 50111335

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Univ.-Prof. Dr.-Ing. Dongheui Lee
Start:	01.04.2019
Intermediate Report:	09.09.2019
Delivery:	27.11.2019

In your final hardback copy, replace this page with the signed exercise sheet.

Before modifying this document, READ THE INSTRUCTIONS AND GUIDELINES!

Abstract

Artificial cognitive systems, such as service robots, which have to process goal-oriented tasks stated in an abstract manner need to be able to recognize semantic relations between objects involved in these tasks. In this thesis, we propose to define these semantic relations according to whether certain objects have been used within the same context. We present a methodology for building a knowledge base, which lists required, optional and non-required objects for a predefined set of actions. Therefore, we propose to employ human demonstration in order to learn from human behavior, i.e. how humans use objects. We construct a virtual environment and collect a data set which contains virtual human action data. After that, we evaluate both our method and the feasibility of using data obtained from virtual reality as input. Our results show, that this procedure in principle is suitable for semantically grouping objects which are actively handled by humans. While employing virtual reality comes with the advantage of having relevant data (such as object positions) readily available, it is not suitable for simulating complex object properties and modelling certain circumstances we face in the physical world, such as coincidence or the social situation.

Contents

Chapter 1

Introduction

Artificial cognitive systems, such as service robots, which have to process goal-oriented tasks stated in an abstract manner need to have a fundamental understanding of these tasks and how to solve them. As for humans, behavioral imitation is an important aspect for gathering this kind of knowledge. For example, every person that knows how to tie its shoes has probably obtained this knowledge by watching someone else doing that. In robotic research this kind of learning is referred to as *imitation learning* [1]. Kuniyoshi described three different cognitive strategies of imitation [2]: the first one is *appearance-based*, i.e. the movements of the demonstrator are copied one-to-one. Another strategy is *action-based*, which involves finding a causal relation between actions and their immediate effect on the target object (learn suitable action for a given primitive). One example for this strategy is the work by Caccavale et al. in the field of multimodal human robot interaction, who developed a framework that allows a robot manipulator to learn how to execute structured tasks involving multiple objects from human demonstration [3]. The third strategy is *purposive-based*. This strategy aims to learn the intention of the entire observed task, independent of individual movements of the demonstrator.

The *purposive-based* approach is the most suitable approach for dealing with tasks stated in an abstract manner, because it focuses on the desired outcome of a task rather than on movements or actions. If a robot is asked to *"get a bottle of water"*, we don't provide any information about the specific movement that is necessary for completing this task. Similarly, we don't take the specific sub-actions into account that might be involved with completing the overall task, e.g. *"opening the cabinet"*. Instead, we want the robot to understand and autonomously decide, how to approach that task.

Therefore, these systems need to observe human behavior and get a deeper understanding of the observed tasks, a process which is described as *purposive learning* in [4]. A key element of purposive reasoning is *semantic reasoning*, i.e. the ability to

reason about certain things (such as actions or objects) in order to achieve a better understanding of the meaning of these concepts. Recent research employs semantic reasoning for real-time action recognition [5] and for an interpretation of human actions, i.e. whether the action performed was successful [6]. However, robotic systems still lack the ability to recognize semantic relations between the objects involved in these tasks. For instance, given the input command *"cut the apple"* and a set of available tools, a robot has to infer the information, that for cutting the apple one has to use a knife.

Contribution and organization of this work: In our thesis, we face the challenge of finding semantic inter-object relations. We propose to find and define these relations in the context of an action that can be conducted by using them. Therefore, we present a methodology for observing humans who conduct predefined actions with a given set of objects. After extracting objects that are involved in a certain action, a knowledge base can be created. This knowledge base lists several actions and the objects that are required for conducting them. Thus, it can be queried in order to infer information that is not explicitly available to a cognitive system, e.g. that for cutting an apple one has to use a knife. In order to implement and test the method, we build a learning framework and a virtual environment, collect virtual human action data and test our approach by evaluating the obtained knowledge base.

The remainder of this work is structured as follows: Chapter 2 provides background information and describes related work, in chapter 3 we present our proposed methodology theoretically. After that, we describe our test setup and evaluate our method in chapter 4. The obtained results are discussed in chapter 5 and the thesis is concluded in chapter 6.

Chapter 2

Background and Related Work

2.1 Object affordances and object functions

For many decades, research areas like object detection, object classification or object affordances have attracted the interest of scientists in computer vision and robotics. In this section, we provide a brief overview of the field of visual object affordance and function understanding, which has become increasingly popular, as the growing research activity for this topic shows (see Fig. 2.1).

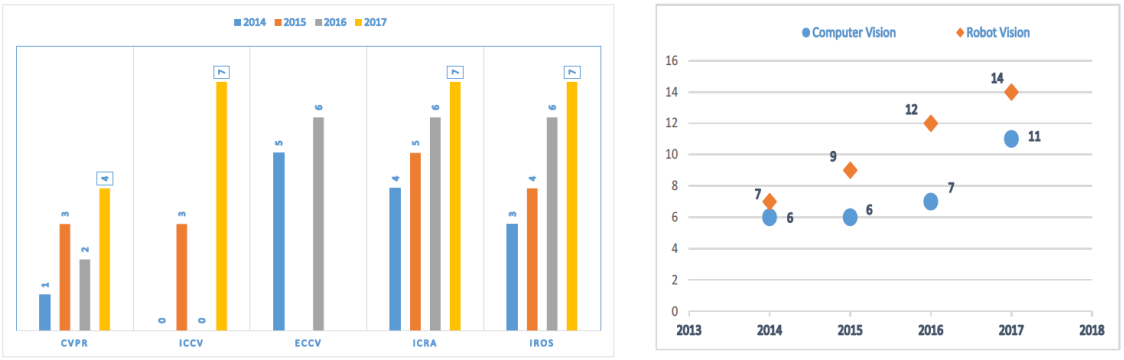


Figure 2.1: Growth in the number of papers on visual affordance in computer and robot vision literature in the recent years (from 2014 to 2017) [7].

The term *affordance* was coined by ecological psychologist James Gibson, who described it as "opportunities for actions", i.e. what can an actor do with the object [8]. As an example, the object *glass* has the affordance *contain*, since it offers the opportunity for pouring something in. Object functions on the other hand, describe possible tasks which can be conducted using the object, e.g. a *knife* has the function *cut*. Thus, object affordances and object functions describe the relation between objects and actions. Detecting and understanding these object affordances and functions is an important step towards fully autonomous robots which interact

with the world. The corresponding research area is broad and consists of multiple sub-tasks. An extensive overview is given in [7], different approaches to this challenge are displayed in Fig. 2.2.

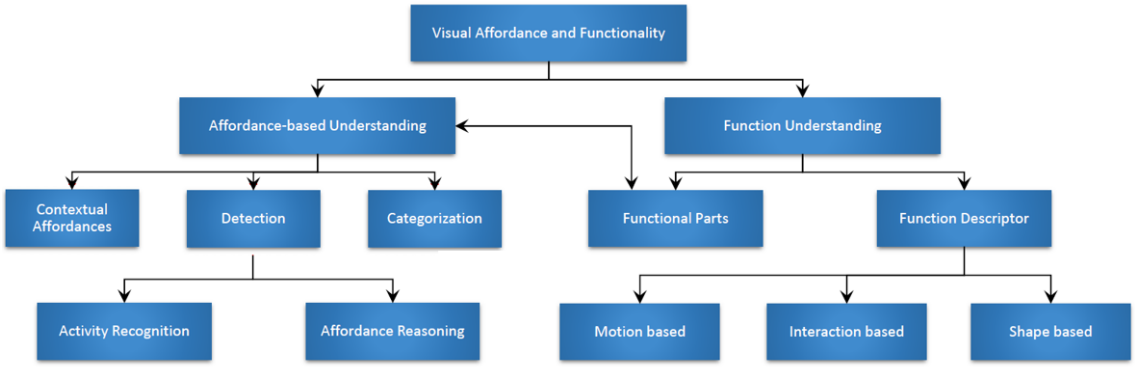


Figure 2.2: Survey taxonomy shows the structure of methods which have been used to solve affordance issues [7].

Additionally, these approaches can be grouped according to at which level they aim to learn affordances or functions:

Scene level: functional scene understanding deals with the challenge of finding out, how a robot can interact with its surrounding environment. This involves clustering the environment and assigning affordances or functions to the resulting regions. Kaiser et al. present a method for detecting environmental elements that allow interaction [9]. For instance, a wall which provides the affordance "lean" can be used by a robot for stabilizing itself in a cluttered environment (see Fig. 2.3). Other work on functional scene understanding can be found in [10, 11, 12].

Object-part level: these approaches deal with finding out, how to use an object or how to interact with it [13, 14]. Therefore, the object parts are segmented and assigned the corresponding affordance. Consider a *pan*, which has the affordances *contain* and *grasp*. The task here is to find out, which part of the object is used for grasping it (i.e. the handle) and which part for pouring something in (i.e. the pot) (see Fig. 2.4).

In our thesis, we neither consider entire scenes, nor object parts. Instead, we solely focus on relations between objects, which means we aim to learn these relations on **object level**. The classical approach to define or describe objects from a computer science or robotics perspective is appearance-based ("a cup is a cup because it looks like a cup"). However, it has been proposed to utilize the object function for describing objects [15, 16]. Instead of trying to describe every single kind of different cup

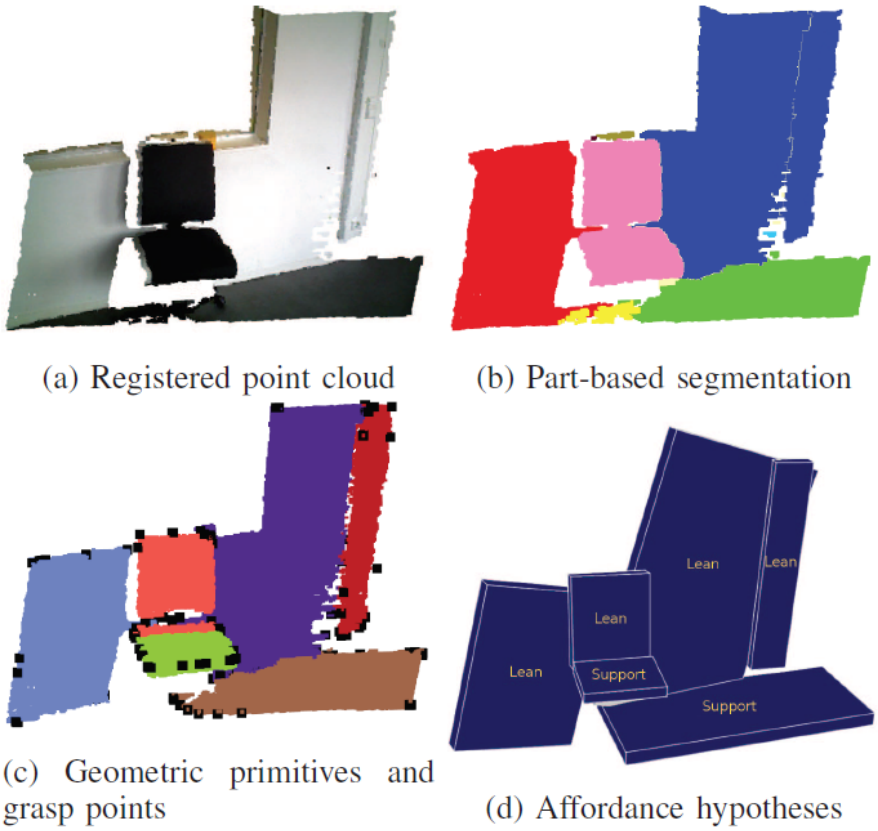


Figure 2.3: The intermediate steps of the perceptual pipeline. (a) RGB-D images are registered and merged into a combined point cloud. (b) The point cloud is segmented based on the convexity of surface patches. Each color represents one segment. (c) Geometric primitives are fitted into the segmented point cloud. (d) Affordances are extracted from the primitives [9]

type, the object *cup* can also be defined using its functionality, namely *containers, that are easy to drink from.* This approach led to a number of scientific publications that employ object functions as the binding element between objects and actions in order to improve cognitive systems.

For example, Castellini et al. improved visual object recognition by adding the information, how a human would grasp the object (e.g. cylindric power grasp, flat grasp, pinch grip, ...) [17]. Gall et al. proposed a method for object categorization via object functionality. Therefore, they captured human motion data of the upper body observed during object manipulation, inferred the object functionality (i.e. how the object is typically handled) and used this information for assigning the objects to pre-defined categories [18]. Similarly, Kjellstroem et al. present an approach for categorizing manipulated objects and human manipulation actions in

context of each other. Several humans performing given tasks with given objects are recorded. In the next step, both human hand actions (hand pose and velocity) and the objects involved are extracted. After that, the actions can be classified by analyzing the objects involved (and vice versa) [19].

All of these approaches share the idea of finding out more about objects by inspecting humans interacting with them (human demonstration). A common limitation of these methods is the assumption, that the interaction with an object always looks the same, e.g. for calling someone with a mobile phone one always moves the hand the same way. This doesn't hold anymore for more complex actions which possibly involve more objects. Finding a single representative hand movement trajectory for the task *sandwich making* is impossible, since there is an infinite number of ways for conducting the action. This is, why we focus on *whether* an object is used in a certain context, rather than *how*.

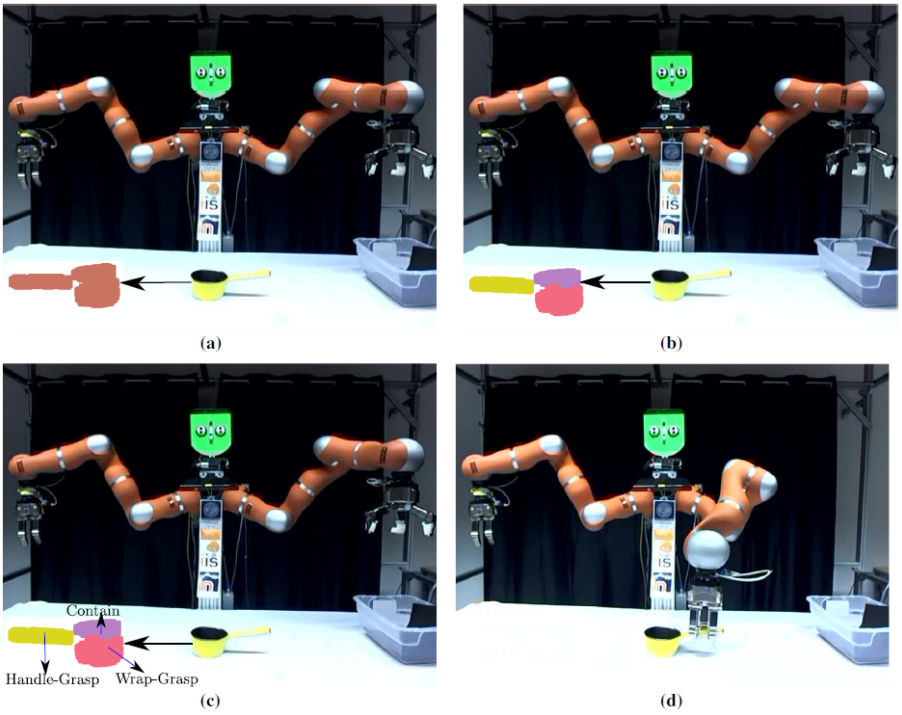


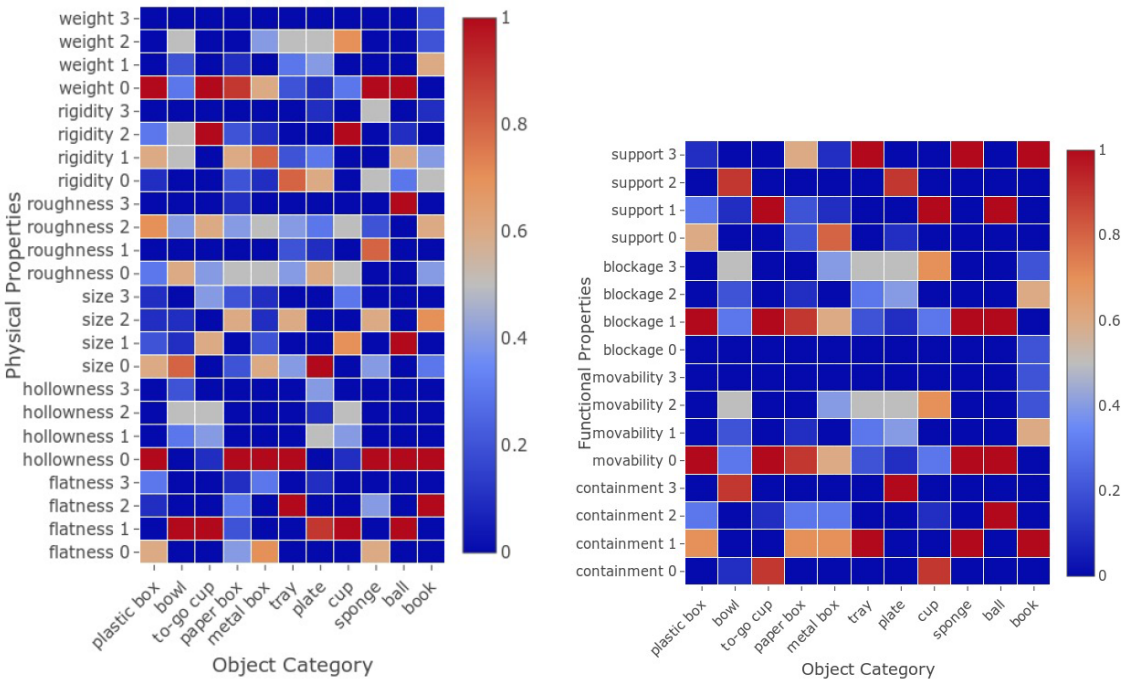
Figure 2.4: The robot is asked to grasp the object on the table using the handle-grasp affordance. It gets an input from the kinect on its chest (a), segments the object into parts (b) and detects its affordances (c). It then uses the handle-grasp affordance for grasping the object (d). The pointcloud and part segmentation of the object are shown based on the view of the robot's kinect [14].

In conclusion, we aim to find semantic inter-object relations on **object level**. Therefore, we employ the idea of using *function* rather than *appearance* for describing

objects. In the following section, we investigate the different abstraction levels of inter-object relations.

2.2 Levels of inter-object relations

After having shown, that instead of appearance objects can also be described using their function, we now discuss the different levels of inter-object relations. One recent work aims to gather conceptual knowledge about household objects by analyzing their physical and functional properties [20]. This conceptual knowledge is essential for tool selection tasks. After extracting the physical (e.g. size, weight, rigidity, ...) and functional (e.g. containment, support, ...) properties of several objects, these numerical values are converted into symbols using a clustering algorithm. Finally, the obtained knowledge about the single objects is generalized to the corresponding object classes. The results are displayed in Fig. 2.5.



(a) Knowledge about Physical Properties (b) Knowledge about Functional Properties

Figure 2.5: Physical and functional properties (0 = lowest value, 3 = highest value) are assigned to object classes. The colors indicate the number of objects belonging to a certain cluster (red = high, blue = low). E.g., most of the objects of the class *plastic box* belong to the cluster with the lowest weight [20].

Using this knowledge we can for example reason, that any object of the object class *plastic box* can be substituted by an object of the class *metal box*, because both classes offer the functionality *containment* in a similar intensity. This approach also uses functionality rather than appearance in order to define objects (as discussed in Sec. 2.1). The relation between objects is defined via sharing the same physical / functional properties. However, this is not sufficient for more complex relations. If someone wants to switch on the TV but the included remote controller is missing, it cannot be replaced by another object with similar physical / functional properties. Even if the other object would also be a remote with the exact same shape and weight as the original one, it could belong to some other electronic device (e.g. sound system) and thus be useless for operating the TV.

Additionally, since the method is set in the domain of tool selection / substitution, it is not suitable for describing relations between two fundamentally different objects, such as door and key. By solely investigating the object properties it is impossible to find any link between these two objects. A cognitive system would have to get a deeper understanding of both objects in order to understand, that a door can be (un-)locked using a key. Hence, we introduce a three-level hierarchy of inter-object relations, which is pictured in Fig. 2.6.

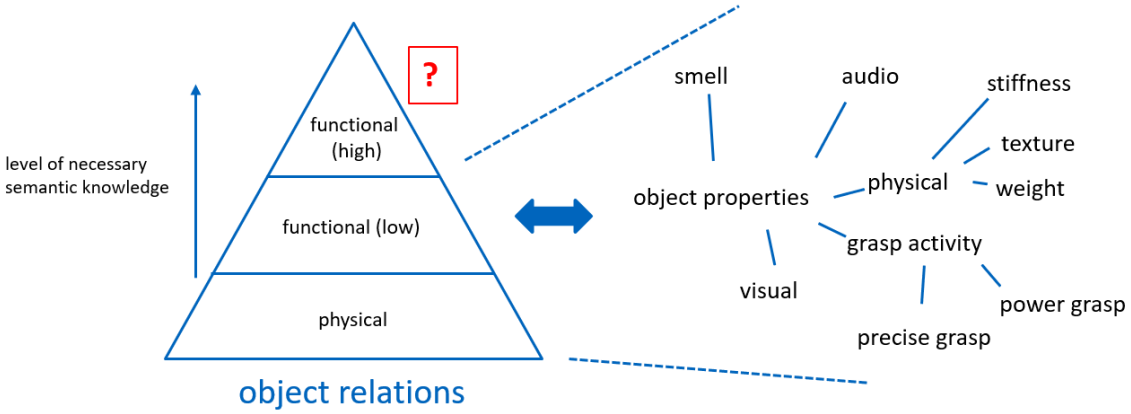


Figure 2.6: Proposed three-level hierarchy of inter-object relations. Basic object properties are sufficient for defining *physical* and *functional (low)* inter-object relations, but are not sufficient for more abstract relations (*functional (high)*).

The two levels at the bottom correspond to the properties outlined in [20]. Object relations can be defined on a physical (e.g. size, weight, ...) or a functional level (e.g. containment, support, ...), which is slightly more complex. We refer to the second level as *functional (low)*, since the corresponding inter-object relations are based on object functions, but can be described by solely analyzing the object properties (such as texture, shape, ...). On the contrary, for defining inter-object relations on

a *functional (high)* level, we need to have a deeper understanding of the objects themselves, since analyzing the object properties is not sufficient anymore. Every object has some sort of relation to many different kinds of other objects, possibly on even more than just one level. For example, a *TV* and a *remote* can have the following relations:

- physical level: both objects are made out of plastic
- functional (low) level: both objects are electronic devices
- functional (high) level: the remote is used for operating the TV

As we already mentioned, inter-object relations on the lower two levels can be found by analyzing object properties. In our thesis, we face the challenge of finding inter-object relations on the *functional (high)* level, which we refer to as **semantic** inter-object relations. As for humans, this kind of knowledge about objects is what we would consider *common sense*, i.e. in many cases the relations are obvious to us. If someone shows us a lighter and a candle we can conclude, that the relation between these two objects is the action *lighting*. Similarly, we know that for locking a door a key is needed. However, for an artificial cognitive system it is a challenging task to reach the same stage of understanding.

Additionally, it is hard to exactly define semantic inter-object relations, because they can depend on culture, personal experience and the social situation. If an Asian person is asked what - besides a plate - is needed for eating, it would probably ask for chop sticks, whereas a European person would ask for a knife and a fork. Another example: for a firefighter it is completely normal to open a door with an axe in an emergency scenario, whereas this procedure wouldn't be considered socially acceptable in a non-emergency scenario. These examples show, that semantic inter-object relations (e.g. plate & cutlery, door & axe) are very abstract and for defining them a lot of circumstances need to be considered. Instead of explicitly stating all parameters and conditions under which a semantic inter-object relation is valid, we propose to learn from human demonstration. If a human being is using several objects in a given context, for example while conducting a certain action, this context can be seen as the semantic relation between the objects.

We conclude, that there are different levels of inter-object relations. In our thesis, we focus on semantic inter-object relations which, unlike "low-level" relations, cannot be defined by solely investigating object properties. Instead, we propose to learn these relations from human demonstration.

2.3 Semantic Reasoning

As discussed in Sec. 2.2, it is a challenging task to teach artificial cognitive systems the kind of knowledge that humans refer to as common sense (e.g. for unlocking a door a key is needed). However, achieving a deeper understanding (i.e. conceptual knowledge) about objects and how humans interact with them is a crucial step for finding semantic inter-object relations. Therefore, an artificial cognitive system needs to be enabled to conduct semantic reasoning on observations of human actions. Recent advances in this field employ knowledge bases for storing information about objects and abstract concepts, such as space, action or time [6, 21]. There is a variety of approaches for building and using knowledge bases [22, 23] and giving a detailed insight is beyond the scope of this thesis.

Generally, a knowledge base contains hand-crafted symbolic knowledge that reflects how humans perceive and describe their environment. Contrary to that, a robot's view on the world is solely based on raw data coming from its sensors (camera, radar, tactile sensors, ...). In order to enable a robot to understand abstract concepts and reason about objects and human activities, these two perspectives need to be connected. This process is called **symbol grounding**. Figure 2.7 shows, how symbol grounding is achieved in the knowledge base KnowRob [24].

On the top, the virtual symbolic view is displayed in the form of an ontology. According to [26], an ontology is a specification of a conceptualization. This means, that abstract concepts are specified in an explicit manner. By inspecting the ontology we can see, that PR2 is a robot (which is a mechanical device) and this robot can be at a certain pose (which is a place). On the bottom, the robot-internal data structures regarding its location estimation are shown. This location estimation is a probability distribution, since it is based on sensor data which are not perfectly accurate. The symbolic knowledge and the raw robot data are connected using so-called computable predicates (hand-crafted PROLOG queries). For instance, the pose of the robot is extracted by analyzing the raw data and retrieving the maximum peak of the probability distribution, i.e. the most probable pose. In order to find out, whether the robot is well-localized it is checked, whether there is more than one peak in the probability distribution. In this case there are three different peaks which means, that there are three probable poses. Thus, the robot is not well-localized. This simple example shows how symbol grounding connects the human perspective (abstract concepts, symbolic knowledge) with the robot's perspective (raw data). The system is now enabled to conduct reasoning on a more abstract level since it has "learned" how to integrate its own perspective in the knowledge base. Instead of dealing with raw data anymore, one can simply input queries, such as "if the robot is **certainly at Point A**, conduct action B".

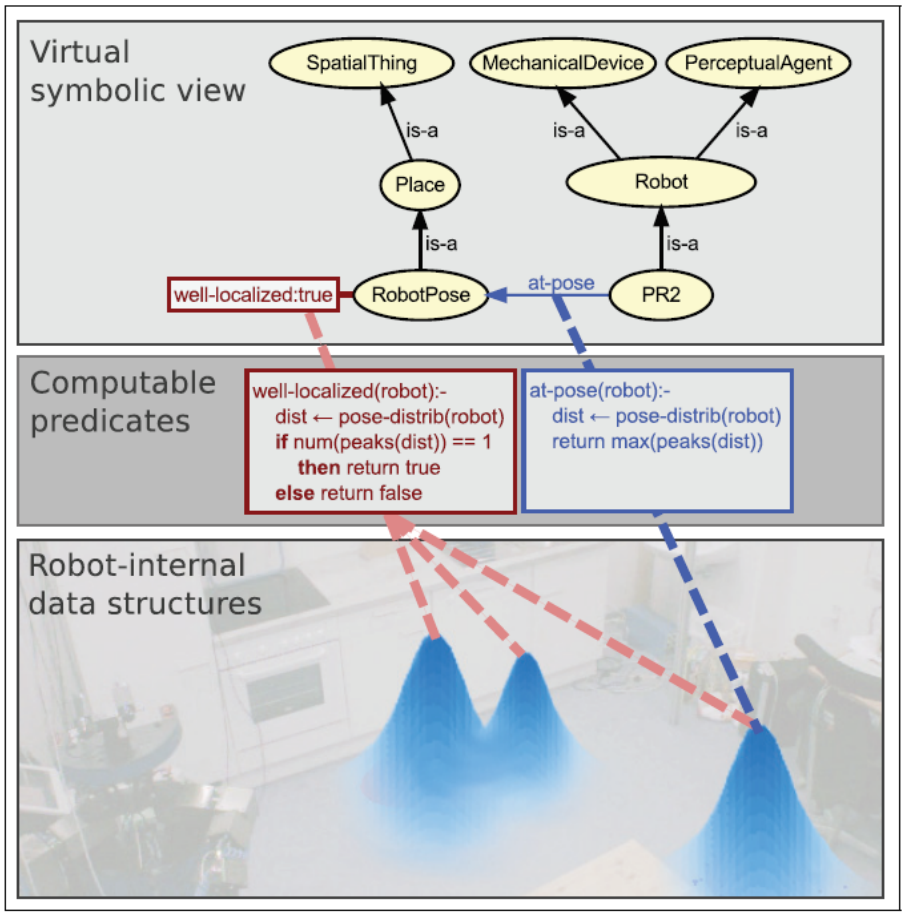


Figure 2.7: Information from the robot's control program, such as its location estimate, can be integrated into a symbolic knowledge base by defining a "virtual knowledge base" (computable predicates) on top of the continuous data [25].

One related work by Ramirez-Amaro et al. employs the knowledge base KnowRob for extracting semantic representations from observations of human activities [21]. These representations are used for learning and classifying actions via ontology-based reasoning on motion parameters and the objects involved in the actions. In a first step, three different hand movement types and two different object properties are introduced (see Fig. 2.8). These abstract concepts are grounded in raw data (video sequences of humans interacting with objects) using the position data of the hands and the objects. For example, the *object in hand* is the object, which has a distance of approximately zero to the hand. After that, a decision tree based on these three parameters (*hand motion*, *object in hand*, *object acted on*) is learned (see Fig. 2.9). The ground-truth data was obtained by manually segmenting recordings of humans acting with objects into hand motions, object properties and human activities. This decision tree can be seen as a set of semantic rules for classifying actions based on the objects involved.

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 $o_h(t) = bread$

Figure 2.8: Definition of human motions and object properties [21].

However, the proposed approach also comes with a few limitations. Due to the nature of the parameters in Fig. 2.8, the method is limited to actions with two or less objects involved. Additionally, it doesn't cope with actions that involve holding two objects simultaneously (such as a knife and an apple while cutting the apple). Another limitation is the focus on distance-based criteria for inferring the objects involved. Some objects can play a crucial role for an action without being intentionally grabbed or approached with another tool. A TV can be switched on by using a remote controller. However, the TV is neither touched, nor does the action require the remote controller being close to it.

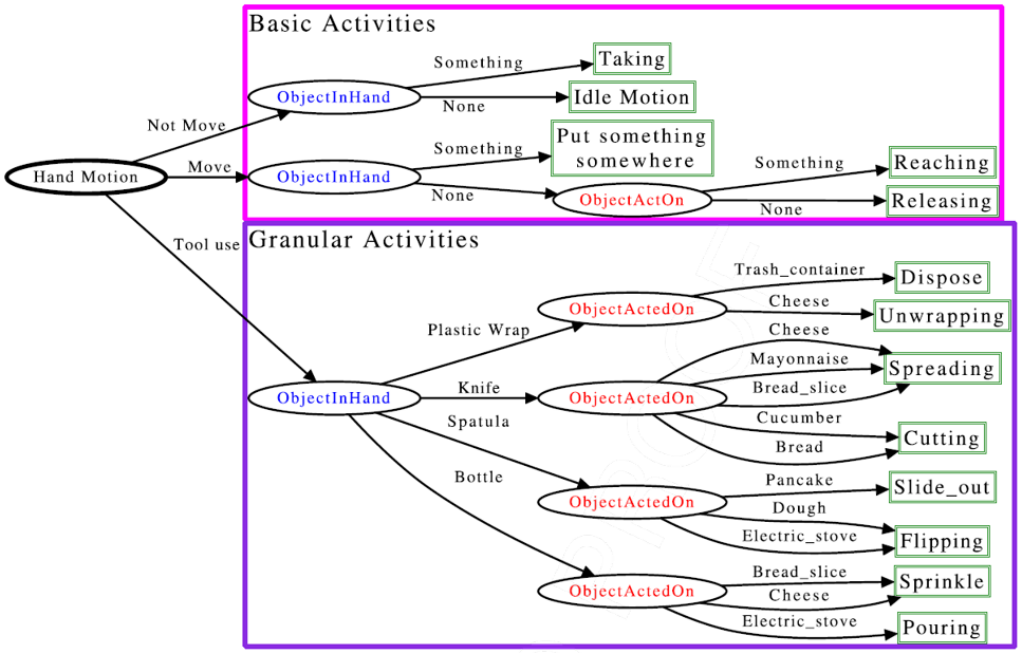


Figure 2.9: Obtained decision tree: the system is capable of conducting semantic reasoning on the hand motion and the objects involved in order to infer the corresponding action [21].

In this section we have shown, how a knowledge base can be used for conducting semantic reasoning about observations of human actions on a more abstract level. The raw data can be integrated in the knowledge base via symbol grounding. In the next chapter, we employ this procedure in order to formulate a method for finding semantic inter-object relations. Our overall aim is to construct a knowledge base that contains the information, which objects have been used within the same context.

Chapter 3

Theoretical Methodology

3.1 Overview

In this chapter we provide a methodology for finding semantic inter-object relations on object level. As we have described in Sec. 2.2, relations based on object properties are not sufficient, because often objects are connected on a more abstract level. We propose to learn object relations from human demonstration, i.e. by investigating how humans interact with objects. After asking multiple test persons to conduct predefined actions using a given set of objects and recording the episodes, we analyze, which objects have been used within the same context. Thus, the semantic relation between a given set of objects (e.g. *glass* and *jar*) is defined by the action that is conducted using these objects (*refilling*). Finally, a knowledge base can be constructed, which contains the learned inter-object relations. Via applying the resulting knowledge base, a cognitive system can be enabled to infer both necessary objects for conducting a given action and reasonable actions for a given set of objects.

The remainder of this chapter is structured as follows: Sec. 3.2 describes, how to obtain a database which can be used for extracting the required object-use information, in Sec. 3.3 we elaborate on how to create a knowledge base based on the collected data. After that, the procedure for applying the knowledge base in order to conduct semantic reasoning is explained in Sec. 3.4.

3.2 Data collection

The starting condition for our methodology is a set of different objects. Although in this thesis we focus on objects in the household domain, the methodology can - in principle - be applied to arbitrary objects that humans can interact with. As stated in Sec. 3.1, our main idea is to define the relations between objects according to human interaction. More specifically, if there exists an action that involves two or more of the given objects, this action can be seen as the binding element

between these objects. Consequently, we also have to define a set of actions that provide links between objects, i.e. can be conducted by using them. It must be possible to execute any of these actions with the objects defined in the initial object set.

After having defined the object and the action set accordingly, we need to gather multiple episodes of humans using the given objects for conducting the defined actions. In order to get a generalizable impression of how humans use these objects, it is important to analyze the behavior of multiple persons. Additionally, we propose to create different test set ups, each with shuffled initial positions of the objects. This procedure comes with two advantages: firstly, we do not need as many test persons for obtaining a significant number of recorded episodes, because each test person can conduct the same action once for every set up; and secondly, we can exclude the effect of object positions when examining object use. If for a certain action multiple persons use the same objects - regardless of their initial position - we can conclude, that the conducted action resembles a semantic relation between the objects involved.

In order to avoid having to deal with action segmentation, we propose to only record a single action per episode. By this means, we also do not have to apply an action recognition algorithm, since the conducted action is already known beforehand (the test person is asked to conduct a particular action before the recording starts) and can be included in the file name of the recording or a separate label. To sum it up, for building a knowledge base we need to collect action demonstrations of several test persons. Each person conducts each action once in every test set up. By this means, we gather a representative impression of human object use. In the next section, we describe how to create a knowledge base using the obtained data.

3.3 Creating a knowledge base

After having collected the required data as described in the previous section, we can now use it in order to create a knowledge base. This knowledge base puts the actions and objects which were defined in the previous section into context. We propose to employ a two-dimensional string array structure (dim: *number of objects* x *number of actions*). Each entry of this structure is either *"required"*, *"optional"* or *"not required"*. Thus, the knowledge base can be queried in order to retrieve required objects for a given action or propose suitable actions for a given set of objects (see Sec. 3.4). In the following, we elaborate on how to obtain this knowledge base.

First of all, we need to find a proper definition for an object being "involved" in an action. While most of the objects that are used for an action are usually manually handled, some objects are not even touched (e.g. when handled with a tool). This means, checking whether an object was actively grabbed by a person is

not sufficient. We can overcome this issue by also observing the objects' position. If an object within a test set up significantly (i.e. more than the background noise of the measurement system can cause) changes its position during a certain action, we can conclude that this object is also relevant. Finally, there are some objects that are still not covered by these two criteria. For instance, a TV is usually neither touched nor does it change its position when it is turned on. The only way to detect the relevance of this particular object is to monitor its state (i.e. turned on / off). Consequently, we consider an object as relevant for an action, if it fulfills at least one of the following criteria:

- the object has actively been grasped by the user
- the object has (significantly) changed its position
- the object has changed its state

Objects that fulfill neither of these criteria are considered passive and thus not required for conducting the action. When examining the individual recorded episodes, each of these criteria is rated with either *true* (= 1) or *false* (= 0), thus resulting in a three-digit value that we refer to as *involvement_type*. The *involvement_type* describes, how an object was involved in a certain action. For instance, Fig. 3.1 shows that object 2 has actively been grasped and also changed its position while the user was conducting action 1.

object	in_hand	changed_pos	changed_state	action
object_1	0	0	1	action_1
object_2	1	1	0	action_1
object_3	0	0	0	action_2
...	...	...	...	...

involvement_type

Figure 3.1: Exemplary object involvement rating scheme. Each criteria (in_hand, changed_pos and changed_stated) is rated individually for every action.

However, these criteria alone are not sufficient. While we can be fairly certain, that an object which has actively been grasped and moved by a human plays an important role for the action conducted, this doesn't necessarily hold for an object that

just changed its state. If a person is cutting an apple and during this process someone else is switching on the TV, the TV changed its state but it is still not relevant for the action *cutting an apple*. On the other hand, a TV changes its state every time it is turned on, but usually isn't grasped or moved by a person in order to turn it on. Additionally, as we aim to learn from realistic human behavior, we cannot sort out episodes where an action was conducted wrongly or in an unreasonable manner (e.g. by also moving around irrelevant objects or accidentally conducting the wrong action). This means, we have to consider not only the way an object is involved in the action, but also at how many occurrences it was involved. Since our data base contains multiple episodes of different persons conducting each action multiple times (once for every test set up), we can filter out unwanted (e.g. coincidental or accidental) effects.

For obtaining the knowledge base, we first of all examine every single episode of our data set and extract the involvement type of each object for every action. By cumulating this information over the entire data set, we get a general overview of the object use. In the next step, we determine whether an object is *required*, *optional* or *not required* for an action.

We consider an object as *required* for a given action, if it fulfills at least one of the following conditions:

- it is grasped by the user in at least 90 percent of recorded episodes of that action
- it (significantly) changed its position in at least 90 percent of recorded episodes of that action
- it has changed its state in at least 90 percent of recorded episodes of that action

On the other hand, an object is considered *not required* for an action, if it fulfills neither of the three criteria (grasped, changed position, changed state) in at least 70 percent of recorded episodes of that action. If an object is neither *required* nor *not required*, it is considered *optional*. At this point, these threshold values are only estimation values which need to be further evaluated.

After analyzing the recorded episodes according to this scheme, we can construct a knowledge base which categorizes the objects into *required*, *optional* and *not required* for each action separately. The formal algorithm for obtaining the knowledge base given a data set as described in Sec. 3.2 can be found in the appendix (DVD: algorithms/algorithm 1).

3.4 Applying the knowledge base

As described in the previous section, the knowledge base puts the objects and the actions into context. Thus, it can be used for inferring required objects for a given action or for recommending actions for a given set of objects.

Examining this knowledge base column-wise, we can - separately for every action - retrieve the objects that are required or optional for conducting it. By applying this procedure, a cognitive system could automatically infer the objects it needs for a given task. For instance, given the exemplary knowledge base pictured in Fig. 3.2, it could infer that object 2 and object 5 are required for conducting action 3. The formal algorithm for extracting both required and optional objects from the knowledge base can be found in the appendix (DVD: algorithms/algorithm 2).

	action_1	action_2	action_3
object_1	required		
object_2	required		required
object_3	optional	required	
object_4		required	
object_5			required

Figure 3.2: Exemplary knowledge base.

Vice versa, a cognitive system could also use the knowledge base for inferring suitable actions for a given set of objects. Given the exemplary knowledge base displayed in Fig. 3.2 together with object 3 and object 4 as input, the system could propose to conduct action 2 since both objects apparently are highly relevant for this action. One use case for inferring the action would be a human pointing at a glass and a bottle without providing any additional information. Querying the knowledge base which contains the relevant information, a robot could be enabled to automatically infer what to do (i.e. refill the glass). The formal algorithm for inferring actions for a given set of objects can be found in the appendix (DVD: algorithms/algorithm 3).

Chapter 4

Evaluation

4.1 Test setup

4.1.1 General information

In order to evaluate our theoretical methodology, we need to build a test setup that enables humans to interact with objects. One possibility is to construct a physical test environment, which contains multiple objects a test user could interact with. This kind of setup would require a calibrated camera for recording the episodes. Additionally, we would need object marker or a similar measure for object detection and computer vision algorithms for object tracking in order to extract the position data. Instead, we propose to conduct our experiments using virtual reality (VR). This means, we construct a virtual environment with a game engine and the user can act within this environment using virtual reality equipment. This procedure comes with a number of advantages, for instance both the object labels and the object position data are inherently available in the game engine.

For our experiments, we use the game engine Unity3d [27] for constructing the virtual environment, because it currently seems to be the most versatile tool and it is widely used. In order to being able to analyze the raw position data of the virtual objects, we employ the robot operating system (ROS) framework. The software ROS-sharp is used for linking Unity3d and ROS [28]. The last component of our setup is the virtual reality equipment. For our experiments, we use the HTC Vive head mounted device (HMD), which comes with two hand-held controllers. The entire setup is displayed in Fig. 4.1 and explained in more detail in the following subsections.

4.1.2 Connecting ROS and Unity3d

As the first step, we need to connect Unity3d with ROS. This step is necessary to transmit the position data of the virtual objects which is inherently available in the

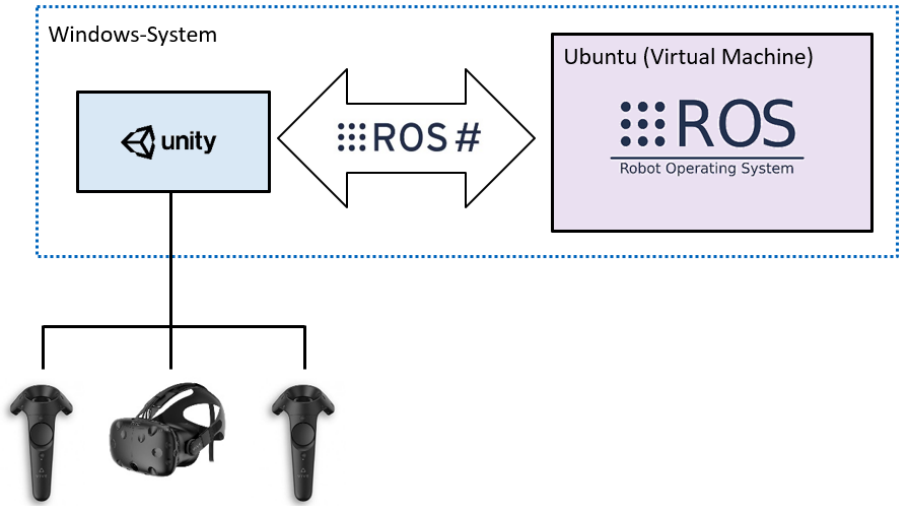


Figure 4.1: Overview of our evaluation setup.

game engine to ROS for further use. Since both Unity3d and ROS are widely used powerful tools, there is a number of approaches for connecting them [28, 29, 30]. In essence, the general idea of these approaches is identical. All of them employ the ROS software *rosbridge*, which provides a JSON API to ROS functionality for non-ROS programs [31]. In our case, the non-ROS program is Unity3d. At the beginning, we aimed to build the entire test setup on an Ubuntu workstation, as ROS is not available for Windows. Because the approach presented in [30] was also realized on an Ubuntu workstation, we employed this implementation in a first attempt. However, we were facing issues when trying to integrate the VR equipment later on. After that, we realized that Unity3d and its VR components run more reliably on a Windows machine.

At the new attempt, we installed Unity3d on a Windows workstation (host) and the ROS framework on a virtual machine (guest). This time, we employed the implementation *ROS-sharp* by Siemens, which is designed for communicating with ROS from .NET applications (in particular Unity3d) and - according to discussions on github - has been successfully used in various projects [28]. In order to set up the connection, there is a number of things to do on both the guest side (ROS running on Ubuntu VM) and the host side (Unity3d running on Windows):

Ubuntu-VM (guest):

- install the ROS software *rosbridge* [31]
- clone the ROS-sharp repository from github [28]
- create an empty catkin-workspace

- copy the folder "ros-sharp-master\ROS\file_server" to the "src" folder of that catkin-workspace
- build the catkin-workspace
- source the catkin-workspace and launch the file "ros_sharp_communication.launch" from the "file_server" package

Result: This procedure starts a websocket server running on the virtual machine (on localhost, port 9090). After that, non-ROS programs can connect to that server and send data in JSON format. The incoming data is converted to ROS messages and published on the specified ROS topics.

Windows (host):

- clone the ROS-sharp repository from github [28]
- copy the folder "ros-sharp-master\Unity3D\Assets\RosSharp" to the "Assets" folder in the Unity3d project
- create an empty GameObject in the relevant Unity3d scene
- add the script *RosConnector.cs* to that GameObject
- import the IP-address of the virtual machine to that script

Result: This procedure makes sure, that Unity3d connects to the websocket server running on the Ubuntu VM after pressing the "play"-Button (i.e. starting the virtual environment). Additionally, the RosSharp folder provides a number of useful scripts that can be used for publishing data to ROS.

After setting up the connection between Unity3d and ROS it is possible, to directly publish position data of virtual objects in the game engine to ROS topics in real time. Within the ROS framework the data can be further processed (e.g. visualized, recorded and so on).

4.1.3 Integrating VR equipment

To this end, there is no possibility for a test person to interact with the virtual objects. Therefore, we have to integrate our VR equipment. This is achieved via SteamVR [32], a system which is available within the Steam platform. After plugging in the HMD, turning on the two hand-held controllers and starting SteamVR, the user is instructed to define the available room (in the physical world) and to put the two controllers on the floor in order to calibrate the tracking stations. These stations track the position and orientation of the HMD and the controllers and map

it to the virtual world. Now that we have set up our VR equipment, we need to import their functionality to our Unity3d project. Therefore, we import the SteamVR plugin from the Unity Asset store [33]. This plugin is used for loading 3d models for VR controllers and handling input from those controllers. On top of that, it contains an extensive interaction system example scene which provides many exemplary functions on how to design the virtual world so that users can interact with it. Via analyzing this example scene and reusing some of the relevant scripts we can make sure, that the objects in our virtual world behave in a realistic way and can be grabbed and moved by a test user.

After having finished the setup, we are able to construct a virtual environment which contains several objects a test user can interact with. Additionally, the position data of both the objects and the controllers can be directly published onto ROS topics for further use.

4.2 Test environment

In order to evaluate our proposed methodology for finding semantic inter-object relations, we build a virtual environment and collect data of multiple test persons who interact with objects. Having set up our framework as described in Sec. 4.1, we can construct so-called virtual scenes with Unity3d. After that, a test user (equipped with a HMD and hand controllers) can immerse himself / herself directly in the virtual world. We then record the performance of the test user, after he / she has received the command to conduct a certain action.

First of all, we define the set of objects that are available in the virtual world. We have chosen a set of eleven objects, that can typically be found in the household domain (see Fig. 4.2, top). These objects have been selected, because they are rather ordinary, cannot be confused by the test users and it is clear, what they are used for. Similarly, we define a set of basic actions, that are easy to understand and to perform (see Fig. 4.2, bottom). Since we aim to find inter-object relations based on human interaction (i.e. which objects have been used for a particular action), it is important to take into account, that the actions can be conducted by using the available objects.

In the next step, we construct a virtual scene which contains a table with the objects displayed in Fig. 4.2 on top. There are two different options for adding new objects to the scene. One possibility is, to search for the object in the Unity Asset store. This store contains plenty of different prefabricated items, materials, effects and so on (some of them for free). If a suitable item is found, it can be imported into the relevant scene. As an alternative, one can design and build an object from scratch.

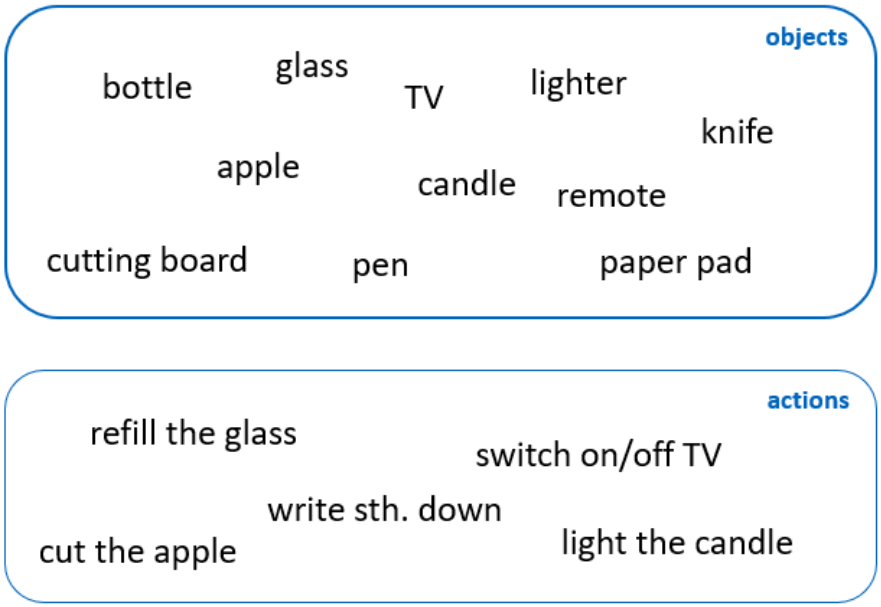


Figure 4.2: Predefined set of objects and actions used for our evaluation. The objects (top) can be used by the test persons in order to conduct the actions (bottom).

Whenever possible, we used the prefabricated items from the Asset store, since this process is more convenient and requires less time. After having imported and placed the objects accordingly, we need to make sure that they follow the law of physics (orientation of gravity, not moving through other objects, ...). Additionally, it must be possible for a test user to interact with the objects by grabbing them and moving them around. These requirements can be fulfilled by applying a number of settings and linking the items to several scripts, that provide the demanded functionality. At this point, we analyzed the provided example scene within the SteamVR plugin and reused the necessary scripts. Figure 4.3 displays an overview of the different settings and scripts that are needed for every single object, e.g. *Rigidbody* (makes sure, that the object experiences gravity) or *Interactable* (defines, how an object is grabbed). Furthermore, we need to create a new component (named RosConnector in our project) for establishing the connection to ROS on our Ubuntu VM, as described in Sec. 4.1.2. Besides the *ROS Connector* script, we add the script *Pose Stamped Publisher* (also available within the *RosSharp*-folder) for each virtual object and also the two hand-controllers. In this way we make sure, that the position data of every object and the hand-controllers (which represent the hands, i.e. the end effector in the virtual world) are published on predefined ROS topics.

As a result, we have a virtual scene which contains interactable objects. In order to get a better representation of how humans use these objects, we create two more scenes with a shuffled initial object setup. By doing so we make sure, that the

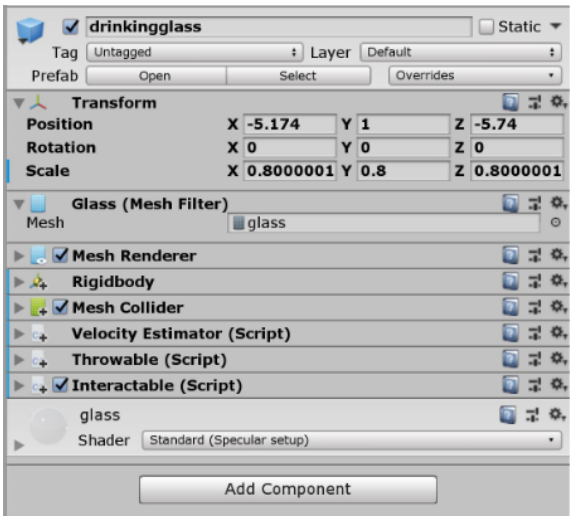


Figure 4.3: Overview of necessary settings and scripts for each object in the Unity3d scene (here: object "drinking glass" as example). These functionalities are required for ensuring a realistic interaction experience in VR.

initial position of an object doesn't play a role for whether or not it is used for a particular action. Additionally, we create a basic training scene, which contains five cubes. This scene is used for familiarizing the test users with the VR experience and enables them to learn how to grab and move objects. Figure 4.4 displays the training scene, Fig. 4.5 one of our three test scenes.

Limitation: Note, that the objects in our virtual environment cannot change their state. This means, for instance, that the TV doesn't change its appearance, when a test user points the remote controller towards it. Similarly, putting the lighter towards the candle will not produce a visible flame. This poses a limitation to our evaluation, as analyzing an object's state is an essential means within our proposed method. We decided to omit this kind of animations in our virtual environment, because the state change in a virtual world wouldn't reflect the state change in the real world. If we would write a script that changes the appearance of the TV when a user points the remote at it and presses a certain button, this state-change would happen every time the user conducts this action. Thus, we could always observe a state-change. On the other hand, in real life there is a number of reasons, why trying to turn on the TV could fail, such as an empty battery in the remote or the TV being unplugged. This example shows, that a fully controlled virtual environment is not suitable for simulating coincidental effects that we happen to encounter in reality.

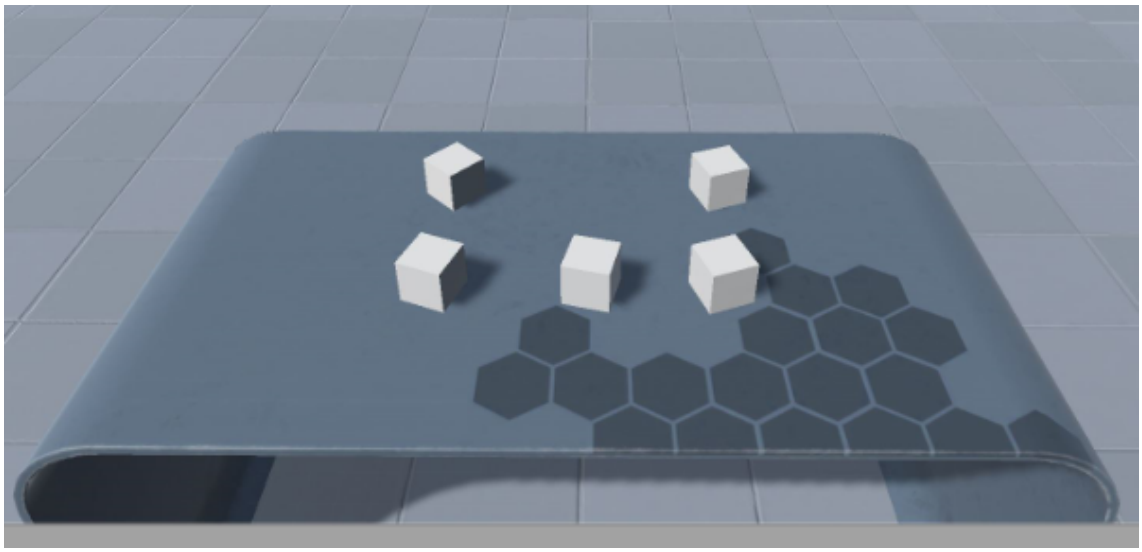


Figure 4.4: Screenshot of our training environment. Participants of our study can get familiar with the VR experience via interacting (e.g. moving, stacking, throwing) with these cubes before the actual data collection phase.

4.3 Data collection

For collecting our experimental data set, we have asked ten test persons to take part in our study (five female, five male). Out of these ten persons, seven declared to have experience in VR. For every participant, we applied the following procedure:

1. immerse the test person into the training scene and ask her/him to play around with the cubes in order to become familiar with the VR experience (approx. 3-5 minutes)
2. immerse the test person into one of the three test scenes and ask her/him to wait for instructions
3. ask the test person to conduct one of the actions displayed in Fig. 4.2 (bottom) after receiving the "Go" command and to tell, when the action is finished (by saying "Done")
4. start screen recording (first person view of test person onto VR scene)
5. start recording the data transmitted to ROS (i.e. position data of objects and hand-controllers)
6. give "Go" command to test person
7. wait for "Done" - command from test person
8. stop recordings

9. shut down VR scene
10. repeat for all 5 actions for all 3 test scenes (go to 2.)

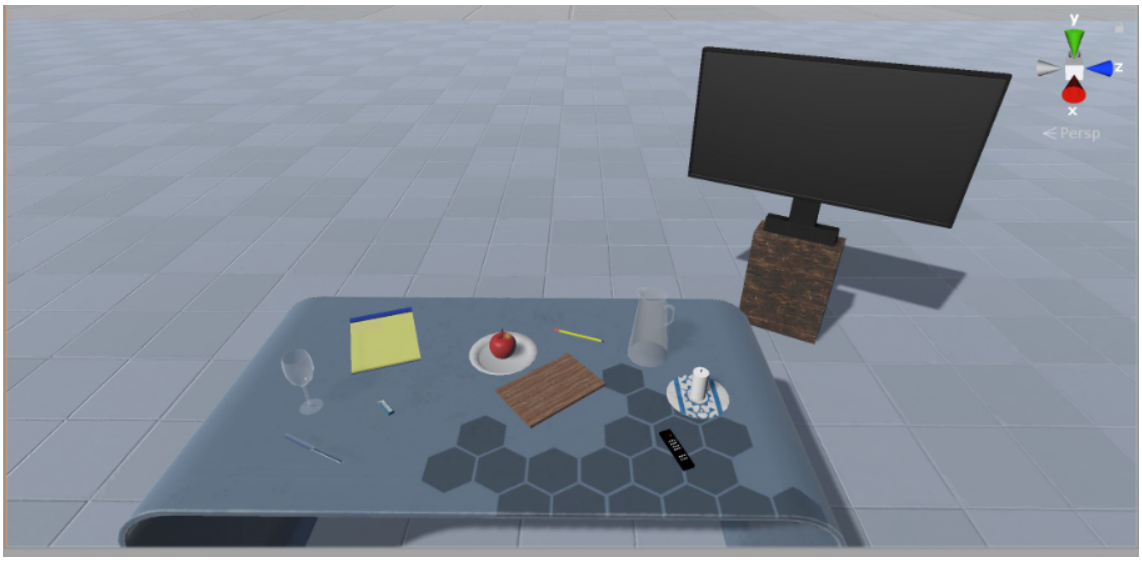


Figure 4.5: Screenshot of one of our three test environments, which contain the predefined set of objects. During the data collection phase, the participants are immersed in these three test environments.

In total, we collected fifteen short episodes per participant (5 actions, each conducted three times). Our final data set consists of 150 episodes (ten participants, fifteen episodes per participant). For each episode, we recorded a *.mp4-file (screen video) and a *.bag-file (raw data). The *.bag-files contain the position data of all objects and the hand-controllers. Note, that this also includes the object labels, since the data is published on the corresponding ROS topic (e.g. the position data of the *lighter* is published onto the ROS topic */lighter*). Also, we do not have to consider action segmentation, because every episode contains only a single action. This data set provides a solid basis for investigating, whether semantic inter-object relations can be inferred from human interaction, as proposed in our theoretical methodology.

4.4 Data analysis

In this section, we analyze our recorded data and apply our proposed method in order to build a knowledge base which contains semantic inter-object relations. Therefore, we manually inspected all of our recorded episodes and investigated, which objects were grabbed or significantly changed their position during a given action. We extracted this information by watching all 150 video episodes and filling in a

table accordingly. Note, that it is also possible to implement scripts, that extract this information from the raw *.bag files. However, watching the videos enables us to get a better insight into how humans actually act in VR and is sufficient for evaluating our method. If the test person actively grabbed an object, we consider the criteria *in hand* fulfilled. As there is no possibility for an object within a virtual environment to change its position other than the test user being actively involved (e.g. by pushing the object with another object), we consider the criteria *changed position* fulfilled, if we can observe a visible movement. Note, that this procedure can be too sensitive for being applied on human interaction performed in the real world, as there might be unwanted effects, for instance light objects being moved by wind. In these cases, one would have to define a threshold for the criteria *changed position*. As pointed out in Sec. 4.2, we did not consider state changes. Our results are displayed in Fig. 4.6.

	cut the apple		light the candle		refill the glass		turn on the TV		write sth. down	
	hand	*pos*	*hand*	*pos*	*hand*	*pos*	*hand*	*pos*	*hand*	*pos*
knife	30	30							1	1
apple	30	30								
board	12	12								3
lighter			30	30						
candle										
glass					21	21				
jar			8	9	30	30	1	1		1
remote			1	1			30	30		
TV										
pen	3	3							30	30
paper									28	28

Figure 4.6: Results of manually annotating our recorded episodes (number of occurrences) according to object use. Errornous occurrences are displayed in red - e.g. a knife isn't needed for writing something down (see top right corner).

We can observe, that in most of the cases object use matches the conducted action. For instance, during all 30 recordings of the action *cut*, the *knife* and the *apple* were grabbed and moved by the user. On the other hand, sometimes the object use doesn't match the conducted action. These instances are displayed in red in Fig. 4.6. There is a number of reasons for these effects. The *pen* was grabbed three times during the action *cut* (and the *knife* once during the action *write*), because these two items were put very close to each other in one initial setup. Thus, the test users accidentally grabbed the wrong object. Another aspect is the fact, that the *jar* was grabbed eight times (i.e. nearly a third of all episodes) during the action *light*, although it is not necessary for conducing this action. This happened, because in one of the three object setups the *lighter* is positioned behind the *jar* and the test users put the *jar* aside in order to reach for the *lighter*. These effects show,

that the object setup plays an crucial role when analyzing object use. Additionally, in some cases objects were moved or knocked over when handling another object (e.g. knocking over the *jar* while handling the *lighter*). This is due to the fact, that virtual objects are weightless and the test person doesn't get any haptic feedback while handling these objects.

Nevertheless, these effects do not pose a limitation to our proposed method, as we analyze all recordings as a whole in order to infer the information, which objects are *required*, *optional* or *not required* for a certain action. We consider an object as *required*, if it was grabbed or moved in at least 90 percent of the recorded episodes (of that action). If it was neither grabbed nor moved in at least 70 percent of the episodes, the object is considered *not required*. In every other case we consider the object *optional*. By applying this procedure we obtain Fig. 4.7. From this table we can conclude that a pen is *not required* for the action *cut*. Thus, the three errornous incidents have successfully been sorted out.

	cut the apple	light the candle	refill the glass	turn on the TV	write sth. down
knife	required				
apple	required				
board	optional				
lighter		required			
candle					
glass			optional		
jar			required		
remote				required	
TV					
pen					required
paper					required

Figure 4.7: Resulting object requirements for a given action.

Figure 4.7 contains the information that we have extracted from human interaction in VR by analyzing which objects have been used within a certain context, i.e. a given action. Thus, it can be seen as a knowledge base, that describes how objects are connected through actions. For example, we can see that the objects *pen* and *paper* are both required for the action *writing something down*. After inputting these two objects to a robot (e.g. by having a human pointing at them and extracting the target objects from the environment) the robot can inspect the generated knowledge base and infer, that the semantic relation between the objects is the action *write something down*.

Similarly, a robot can query the knowledge base for missing objects. If it is asked to conduct the action *cut the apple*, the robot can inspect the knowledge base and

obtain the information, that for performing this task a *knife* is also required (and a *board* is optional). By this means, we can equip a cognitive system with the ability to find semantic inter-object relations via learning from humans.

However, as stated in Sec. 4.2 our evaluation of the method is limited due to the fact, that we don't consider the state change of an object. Because of this, we are not able to infer the information, that the *TV* is involved in the action *turn on the TV* and the *candle* is involved in the action *light the candle*. In general, this limitation holds to all objects that are neither touched nor moved, but still involved in a certain action. Additionally, in Fig. 4.7 the *glass* is considered as *optional* for the action *refill the glass*, although it was involved in every single episode of that action. This is, because some test persons left the glass at the initial position, didn't touch or move it at all and only grabbed the *jar*. Thus, the *glass* wasn't handled in enough episodes to be considered *required*. Similarly, in some episodes of the action *cut the apple* the test users directly placed the *apple* on the *board*, without touching or moving the *board* beforehand. Therefore, the use of this particular object couldn't reliably be detected. In the next section we will further discuss our results.

Chapter 5

Discussion

In this chapter we discuss both our results presented in Sec. 4.4 and some insights, that we have gained during evaluating our method. One important decision within the scope of this work was, to do the evaluation using VR instead of building a physical test setup. This procedure is suitable, if humans are interacting with objects the same way in VR as in the physical world. Otherwise, the knowledge obtained in VR is useless in the physical world. We could observe, that in general the participants were confident with acting in a virtual world. In most of the cases, they chose the appropriate objects for conducting a given action which means, that by analyzing their behavior the action can be extracted as the link between the objects involved. Although human behavior in VR is not perfectly natural (no haptic feedback, objects are weightless, ...), this procedure proves sufficient for our work, since we are solely interested in the question, whether or not a certain object was used. Additionally, a virtual setup can be constructed more quickly, is easily reproduceable and the position data is inherently available within the game engine. We also inspected the raw data of randomly selected episodes and compared them with what can be seen in the videos. For every inspected recording, the raw position data perfectly matches with what we could observe in the video. Thus, our procedure for extracting semantic inter-object relations from human behavior in VR is scalable. One can easily construct a more complex environment with more objects and extract the object use via implementing a script for inferring object use from the raw data automatically.

During our evaluation we noticed, that in some episodes object use doesn't match the conducted action. For instance, we sometimes couldn't detect object use, because the object was neither grabbed nor changed its position. We believe, that the number of errornous episodes can be reduced by having more complex environments with more variance regarding the initial object setup. In our environment, all objects were readily available on a table right in front of the test person. This means, that the participants do not have to reach for the *glass* in order to conduct the action *refill the glass*. They can simply grab the jar, move it towards the glass and leave the

glass untouched. Thus, it is not possible to detect, that the *glass* was also involved. We think, that if the *glass* would be at a more "uncomfortable" position, such as in a shelf, the participant would naturally reach for it and put it on the table / on the counter before starting to refill it. This would enable us to detect the object use. Additionally, we sometimes incorrectly registered an object as involved in an action. In one of our test scenes, the *lighter* was positioned behind the *jar* and almost every participant moved the jar in order to reach for the *lighter*. This leads to a systematic error, as in many cases the *jar* is falsely detected as an involved object. Thus, we have to consider to construct several environments with significantly different initial object setups. This ensures, that the results regarding the objects involved solely depend on their relevance for the conducted action instead of the object setup.

The most challenging aspect of our methodology is analyzing the objects' state. We theoretically proposed to also observe object states, because for some objects this is the only possibility to detect, whether they are involved in a certain action. For instance, a TV is neither touched, nor does it change its position while being turned on. Thus, these two criteria are not sufficient and we actually have to check, whether the TV is on or off. As discussed in Sec. 4.2, VR is not suitable for realistically modelling object states, since it is a fully controlled environment. Every object has a huge number of properties and it is cumbersome to simulate these in a virtual world. Additionally, it is nearly impossible to model effects like coincidence or the social situation virtually. In general, humans unconsciously take many aspects into account, when dealing with objects. When picking a glass to fill and drink from, we wouldn't choose a broken or a dirty glass. Additionally, we might also take into account, whether the glass type fits the drink (e.g. wine glass for wine) or if it is already used by another person. This kind of knowledge is what we commonly refer to as common sense. In this work, we solely analyzed object use based on whether the objects have been grabbed or changed their position. This procedure is useful for gathering basic knowledge, e.g. which tool needs to be selected for a given task. However, in order to equip robots with the ability to conduct deep and abstract reasoning on objects, we have to find a way to teach them human common sense.

Chapter 6

Conclusion

In this thesis we studied the idea of finding and defining semantic inter-object relations by analyzing human behavior. Therefore, we proposed a methodology in order to find out, which objects are involved in certain actions. We consider an object as involved, if it was grabbed, changed its position or changed its state during the action. The overall idea of this approach is, that an action that can be conducted with a set of objects poses a semantic link between these objects. After extracting the involved objects, we propose to build a knowledge base that lists several actions and the objects that are required for conducting them. This knowledge base can be used by a cognitive system for conducting semantic reasoning, as one can - for instance - infer a suitable action for a given set of objects.

We evaluated this thought via constructing a virtual environment, collecting a data set which contains human behavioral data and manually performing our proposed method on this data. For our evaluation we focused on the criteria, whether an object was grasped or moved by a user, since a virtual environment is not suitable for simulating such complex properties like object states. We found out, that most of the objects could reliably be assigned to the corresponding action which means, that this information can be stored in a knowledge base for further use. This procedure is limited to objects, which are actually grabbed or moved by a person during an action - we couldn't detect the use of the TV since we were not observing object state. However, there is only a few number of objects for which this limitation applies. We conclude, that analyzing human behavior in VR is a viable means for analyzing object use in order to build a knowledge base for further semantic reasoning.

In this work, we defined a small set of possible objects and actions that are being considered and our virtual environment is rather simple. Since our approach is scalable, possible future work would include constructing more complex environments which are more realistic and contain more objects and interaction possibilities. Additionally, one could automatically detect involved objects by implementing scripts that analyze the raw data in real time. Eventually, we want to emphasize that this

procedure in general is only capable of observing *that* an object is used. Enabling a cognitive system to deeply reason about objects would require them to understand, *why* an object was used - this challenge remains unsolved.

Appendix A

DVD

The DVD attached to this thesis contains the following material:

- referenced papers
- mid-term report and presentation slides
- final report and presentation slides
- Unity3d project (including ROS software)
- algorithms
  - algorithm 1: building the knowledge base
  - algorithm 2: inferring required and optional objects for a given action
  - algorithm 3: inferring actions for a given set of objects
- recorded data set

List of Figures

Bibliography

[1] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] Y. Kuniyoshi, "Learning from examples: Imitation learning and emerging cognition," *Humanoid Robotics and Neuroscience: Science, Engineering and Society, G. Cheng, Ed.(CRC Press, Boca Raton, FL, 2015)*, pp. 223–250, 2015.

[3] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction," *Autonomous Robots*, vol. 43, no. 6, pp. 1291–1307, 2019.

[4] G. Cheng, K. Ramirez-Amaro, M. Beetz, and Y. Kuniyoshi, "Purposive learning: Robot reasoning about the meanings of human activities," *Science Robotics*, vol. 4, no. 26, p. eaav1530, 2019.

[5] T. Bates, K. Ramirez-Amaro, T. Inamura, and G. Cheng, "On-line simultaneous learning and recognition of everyday activities from virtual reality performances," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3510–3515, IEEE, 2017.

[6] A. Haidu and M. Beetz, "Action recognition and interpretation from virtual demonstrations," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2833–2838, IEEE, 2016.

[7] M. Hassanin, S. Khan, and M. Tahtali, "Visual affordance and function understanding: A survey," *arXiv preprint arXiv:1807.06775*, 2018.

[8] J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[9] P. Kaiser, M. Grotz, E. E. Aksoy, M. Do, N. Vahrenkamp, and T. Asfour, "Validation of whole-body loco-manipulation affordances for pushability and liftability," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 920–927, IEEE, 2015.

[10] V. Delaitre, D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros, "Scene semantics from long-term observation of people," in *European conference on computer vision*, pp. 284–298, Springer, 2012.

[11] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic, "People watching: Human actions as a cue for single view geometry," *International journal of computer vision*, vol. 110, no. 3, pp. 259–274, 2014.

[12] M. Grotz, P. Kaiser, E. E. Aksoy, F. Paus, and T. Asfour, "Graph-based visual semantic perception for humanoid robots," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 869–875, IEEE, 2017.

[13] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1374–1381, IEEE, 2015.

[14] S. R. Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Towards affordance detection for robot manipulation using affordance for parts and parts for affordance," *Autonomous Robots*, vol. 43, no. 5, pp. 1155–1172, 2019.

[15] P. H. Winston, T. O. Binford, B. Katz, and M. Lowry, *Learning physical descriptions from functional definitions, examples, and precedents.* Department of Computer Science, Stanford University, 1983.

[16] L. Stark and K. Bowyer, "Achieving generalized object recognition through reasoning about association of function to structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 1097–1104, 1991.

[17] C. Castellini, T. Tommasi, N. Noceti, F. Odone, and B. Caputo, "Using object affordances to improve object recognition," *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 3, pp. 207–215, 2011.

[18] J. Gall, A. Fossati, and L. Van Gool, "Functional categorization of objects using real-time markerless motion capture," in *CVPR 2011*, pp. 1969–1976, IEEE, 2011.

[19] H. Kjellström, J. Romero, and D. Kragić, "Visual object-action recognition: Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81–90, 2011.

[20] M. Thosar, C. A. Mueller, G. Jaeger, J. Schleiss, N. Pulugu, R. M. Chennaboina, S. V. Jeevangekar, A. Birk, M. Pfingsthorn, and S. Zug, "From multi-modal property dataset to robot-centric conceptual knowledge about household objects," *arXiv preprint arXiv:1906.11114*, 2019.

[21] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artificial Intelligence*, vol. 247, pp. 95–118, 2017.

[22] M. Thosar, S. Zug, A. Skaria, and A. Jain, "A review of knowledge bases for service robots in household environments," in *6th International Workshop on Artificial Intelligence and Cognition*, 2018.

[23] D. Paulius and Y. Sun, "A survey of knowledge representation in service robotics," *Robotics and Autonomous Systems*, vol. 118, pp. 13–30, 2019.

[24] M. Tenorth and M. Beetz, "Representations for robot knowledge in the KnowRob framework," *Artificial Intelligence*, vol. 247, pp. 151–169, 2017.

[25] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[26] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[27] Unity-Technologies, "Unity." `https://unity.com/`. last accessed: 14/11/2019.

[28] M. Bischoff, "ROS-sharp." `https://github.com/siemens/ros-sharp`. last accessed: 14/11/2019.

[29] Y. Mizuchi and T. Inamura, "Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks," in *2017 IEEE/SICE International Symposium on System Integration (SII)*, pp. 948–955, IEEE, 2017.

[30] A. Hussein, F. García, and C. Olaverri-Monreal, "ROS and Unity based framework for intelligent vehicles control and simulation," in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, IEEE, 2018.

[31] Open-Source-Robotics-Foundation, "rosbridge." `http://wiki.ros.org/rosbridge_suite`. last accessed: 14/11/2019.

[32] Valve-Corporation, "SteamVR." `https://www.steamvr.com/`. last accessed: 14/11/2019.

[33] Valve-Corporation, "SteamVR plugin." `https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647`. last accessed: 14/11/2019.