Affordances for and from manipulation; A developmental approach to tool use learning on iCub.

Tanis Mar

ISTITUTO ITALIANO
DI TECNOLOGIA

DOCTORAL SCHOOL

ON LIFE AND HUMANOID TECHNOLOGIES

Affordances for and from manipulation: A developmental approach to tool use learning on iCub

by

Tanis Mar

Dissertation presented in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Advisors:
Vadim Tikhanoff
Lorenzo Natale

iCub facility
Istituto Italiano di Tecnologia (IIT)

February 2017

Abstract

The ability to use tools can significantly increase the range of activities that an agent is capable of. Humans start using external objects since an early age to accomplish their goals, learning from interaction and observation the relationship between the objects used, their own actions, and the resulting effects, i.e., the tool affordances. Humanoid robots capable of autonomously learning affordances in a similar way would be much more versatile and easier to design, by being able to take advantage of a world, as ours, populated with tools and devices designed by and for humans. Such robots would be able to overcome the restrictions of their bodies by taking advantage of external elements to perform tasks for which their own manipulators are insufficient or inefficient. Motivated by this prospect, in this Thesis we introduce a set of methods to represent, learn and generalize tool affordances.

The presented procedure consists of three phases; tool incorporation, tool representation and tool affordance learning. The first phase enables the robot to recognize a tool in its hand, reconstruct its geometry, estimate the pose with which it is being grasped, and determine the position of the tooltip. This information is critical for the robot in order to attach the tool to its body kinematics, a requirement for accurate interaction. Tool representation is achieved by means of functional features, specifically devised for interaction scenarios. For that end, we introduce robot-centric functional features, which characterize the tool's functionality in terms of its geometry and the way in which it is grasped by the robot. Finally, we introduce and evaluate a series of cognitive architectures which, making use of the previously described techniques, allow the iCub to learn tool affordances from interaction. These architectures represent an incremental development towards more capable methods for tool affordance prediction, in terms of both accuracy and generalization.

Together, the proposed methods constitute a novel and coherent robot-centric framework for the study of tool affordances, and advance the state-of-the-art with respect to their representation, implementation and learning.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Tool use in humans

Tool use is a cornerstone of human civilization. Tools impregnate our daily lives to an extent where it is difficult to imagine any common human activity that is not mediated in one way or another by them. Tool use allows, in essence, to overcome the restrictions of our bodies and use external elements to achieve goals that otherwise would have been impossible, or to render their completion more efficient. As such, the ability to learn how to use tools is arguably one of the most powerful mechanisms that humans and other animals have for adaptation and problem solving.

What is more, there is evidence that the relationship between humans and tool use runs even deeper, as the competitive advantage that it provided to primitive tribes in order to access food sources benefited their survival and reproduction. This advantage generated in turn an evolutionary drive towards two-legged locomotion in order to free the hands for tool use, and increased brain size in order to remember tool material sites, perform complex tool manufacturing, and pass this knowledge to subsequent generations [31, 196, 4, 21].

Advanced tool use seems, indeed, to be a characteristic unique to humans; while many other animals are also known to use and even manufacture tools (see [12] and [156] for a comprehensive catalogue), only humans are able to use tools to manufacture new tools [104]. The occurrence of this decisive process, which evidence suggest happened first around 3.4 million years ago [33], established the beginning of human technology. Around this time, humans started developing tools as a means for more effective hunting and foraging (axes, knives, etc) which form the base of the first tools that have been known to exist. Thereafter, in parallel with the increase in the cognitive capabilities of primitive humans, and their ability to design and fabricate tools, these steadily grew in sophistication, eventually leading to the technology driven world in which we live today.

As a matter of fact, most of our routine activities nowadays are performed with tools, such as eating (cutlery), writing or drawing (pen, pencil, brush, etc), cleaning ourselves (toothbrush, etc) or our environment (broom, sponge, mop, etc), as well as most crafting skills such as woodwork (saw, chisel, hammer, etc) and sawing (needle, hook, knit, etc), to name just a few.

Yet, although the notion of what a tool is and what is not seems quite intuitive for us, finding a formal definition of what constitutes a tool and what is tool use has originated an intense debate among scholars of different disciplines including sociology, anthropology, psychology, and even, in recent times, robotics.

Many definitions have been proposed, focusing on the diverse aspects of tool use, such as sociological, developmental, control, etc. One of the most accepted definitions was provided by Beck in his seminal study of tool use in animals [12], and recently updated with minor changes which endorse some of the critiques [169, 162] in which he defines tool use as follows [156]:

"Tool use is the external employment of an unattached or manipulable attached environmental object to alter more efficiently the form, position, or condition of another object, another organism, or the user itself when the user holds and directly manipulates the tool during or prior to use and is responsible for the proper and effective orientation of the tool."

Examining this definition, we can observe three main ideas of what defines tool use that will be influential during the rest of the present Thesis.

- Tools are freely manipulable external object, either attached or unattached.
- The tool needs to be properly oriented by the user.
- Tool use leads to an alteration of the environment, i.e. it generates an effect, either intended by the user or not.

This definition does not encompass more complex tools result of advanced technologies such as cars or computers, which are referred as Constructors by the authors of the definition [12, 156]. Therefore, it is not only more suited for the study of tool use in animals, as originally intended, but also for any kind of tool use approach based on manipulation and interaction, such as the one proposed in this Thesis.

1.2. Tool use in humanoids

Humanoid robots, despite appearing in fictional stories since almost a century, have only became a reality in the last few decades, and most recent years have witnessed an unprecedented increase in their numbers and capabilities. Although at first it might seem that attempting to build humanoid robots – instead of building them with optimized morphologies for specific task – is just a futile exercise of self-adulation, advantages of building humanoid robots are manifold.

On a conceptual level, if we accept that our concepts and view of the world arise not in an purely abstract symbolic way, but through the interaction with the environment mediated by our own bodies, similar bodies are a requirement if we expect robots to form the same kind of concepts that we do. Only in this case, if robots eventually achieve full autonomy, will we be able to communicate with them on similar terms.

On a more practical level, there are also several reasons for studying and developing humanoids: First, from the mechanical point of view, research in humanoid limbs and human prostheses can learn much from each other, and maybe at some point produce interchangeable parts. Secondly, imitation learning from human teachers is eased by similar body structures, as observed actions can be almost directly mapped into the robot's body frame. Also, concerning human psychology, it has been shown that anthropomorphic bodies, and specially faces, facilitate human identification with robots, basic for natural and smooth interaction. Finally, and most importantly for the purpose of this Thesis, robots with a scale and motor capabilities close to human ones could take advantage of a world, as ours, populated with tools and devices designed by and for humans.

Being able to benefit from the available tools will allow humanoid robots to be more versatile and carry out a wide range of tasks for which they had not necessarily been equipped, or even designed. Moreover, it would favour shared workspaces between humans and robots, where both could work alongside without the need for specific and separate setups. However, humanoid morphology on its own is not enough to enable proper utilization of human tools and devices. Naturally, knowledge on how to manipulate them is also required for successful tool use. Engineered approaches, where the required knowledge is provided externally by the user or designer, can be sometimes easily applied and provide good results in constrained scenarios. However, they fail quickly in uncontrolled situations with unpredictable elements.

Autonomous learning through interaction and exploration, on the other hand, can render the learning process more flexible and potentially open ended. This kind of approach, usually referred to as developmental, has shown in the past the potential to yield robust and adaptive systems for a variety of tasks and applications, specially suited for unstructured scenarios or where the problem could not be explicitly encoded by the experimenter [93, 152, 9, 167, 22].

The ability to autonomously learn how to take advantage of external tools in order to perform tasks for which their own manipulators are insufficient or inefficient, will dramatically increase the range of activities that humanoid robots are capable of, as well as making them easier to design. Moreover, such autonomous learning strategies based on the robots own experiences could one day enable them to discover tools and uses beyond humans', empowered by their increased strength, speed or precision.

Therefore, in the recent years we have seen a rapid increase in the amount of groups and studies tackling the problem of robotic tool use from a developmental standpoint. The present Thesis is one of them. Its aim, to advance the state-of-the art in this particular field by developing techniques by means of which robots can learn about tool use in a developmental fashion. Specifically, to enable the iCub robot to learn from observation the consequences of its own actions and how they depend on the tool it is holding, as well as its pose.

To this end, we have decided to tackle this problem from the perspective offered by the Theory of Affordances [54], which provides a convenient framework whereby the different elements of tool use can be related together. This is made explicit in the application of *functional features*, which constitute an implementation of the *direct perception* notion put forward by the theory, as well as in the learning methods presented, which are based on a formalization of the same theory.

1.3. Outline of the Thesis

The current Thesis is organized as follows: Chapter 2 provides a comprehensive survey of the existing literature regarding Affordances, beginning with the discussion about its definition and formalization, and then moving on to previous applications of this concept in robotics, focusing specially on the ones for tool use learning. The chapter finishes with an overview of the concept of Functional Features and a review of studies which have previously applied them. Chapter 3 describes the hardware and software

elements applied throughout the rest of the study, namely, the iCub Humanoid robot, its middleware YARP, and the relevant modules and external libraries used.

On Chapter 4 we present a novel method for tool incorporation, that is, estimation of the tool's geometry, reach and pose with respect to the iCub's hand, and its attachment to the robot's kinematic chain. Next, on Chapter 5 we introduce two different sets of Functional Features devised specifically to represent tools and their pose in interaction scenarios, based on 2D and 3D information, respectively. Chapter 6 constitutes the main part of this Thesis. It presents our proposed approaches for tool affordance learning and their experimental evaluation, for which we make use of all the elements described in the preceding chapters. Chapter 7 describes a working demo which demonstrates how all the methods described above can be integrated in order to produce useful and robust behaviors. Finally, in Chapter 8, we summarize all the contributions present in this Thesis, as well as its shortcomings and prospectives for future work.

2. Literature Review

2.1. What are affordances?

The concept of affordances was introduced by J.J. Gibson in 1977 in his article *The Theory of Affordances* [53], and was further refined in his posterior book *An Ecological Approach to Visual Perception* [54], where it constituted a central element in his theory of direct perception and ecological psychology. In it, Gibson challenges the mainstream view about perception at that time: "orthodox psychology asserts that we perceive these objects insofar as we discriminate their properties or qualities" (Gibson 1979: 134), and proposes instead "... that what we perceive when we look at objects are their affordances, not their qualities" and "If so, to perceive [entities] is to perceive what they afford". Accordingly, he defined the term *affordances* in the following way:

> "The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. I mean by it something that refers to both the environment and the animal in a way that no existing term does" (Gibson 1979: 127).

Although Gibson's was the first definition, his work shows great influence from earlier philosophers such as Heidegger or Maurice Merleau-Ponty, and included concepts that have been explored in psychology previously, such as John Dewey's views on the tight interplay between action and perception, put forward on his book The Reflex Arc Concept in Psychology (1896) [37], the idea of Funktionale Tönung of objects conceived by Jakob von Uexküll in his 1920's book *Kompositionslehre der Natur* [194] or Koffka's proposed law of functional relevance disclosed in the influential 1935's book *Principles of Gestalt Psychology* [80], as well as by contemporaries like Neisser (1976) in his book *Cognition and Reality* [120].

In the subsequent years after Gibson's seminal work, several authors attempted to provide less abstract and hence more usable formalizations of the concept of affordances.

Michaels & Carello (1981) and Shaw, Mace & Turvey (1982) defined affordances as dispositional properties of the environment, which offer the potential for action, and introduced the concept of *effectivities*, as dispositional properties of the agents which relate to their abilities [28, 154]. The actualization of this pair occurs through interaction, when both the environment's affordance and the agent's effectivity juxtapose [176]. This approach, however, conflicts with the idea put forward by Gibson that "It is equally a fact of the environment and a fact of behavior. It is both physical and psychical, yet neither. An affordance points both ways, to the environment and to the observer." (Gibson 1979: 129). In fact, as Dotov et al. would argue much later, the discussion seems to be purely semantical, as the affordance-effectivity dual on Shaw's formalism corresponds in meaning to the affordance on Gibson's definition [40].

The discussion about whether affordances were properties of the environment, of the agent, or both, continued in subsequent years with further developments of the idea described above by Shaw, Mace & Turvey, as in [176], as well as contributions from other authors such as Greeno, who proposed that affordances are constraints in the environment to which agents are attuned to [62], and Stoffregen, who argued that affordances are not properties of the environment, but rather emergent properties of the animal-environment system [165]. Influenced by the experiments that Warren had carried out in the 80s [197], Stoffregen and Heft gave great importance to the body-metrics aspect of affordances, i.e., how agents perceive affordances of the same object or environment differently depending on their size and constitution [66, 166]. An alternative formalization was proposed by Steedman, whose background in dynamical systems, distinct from the cognitive psychology background of most previous authors, rendered its view less concerned with the origin and ontology of affordances, and more focused instead on the applications of affordances for planning. According to him, the affordance-set of a particular object schema is constituted by the different actions associated with that object schema, as well as their pre-conditions and consequence, and can be populated by learning. Thus, affordance-sets map objects to functions that relate the preconditions and effects of the action to the object [164], which allow their use in planning.

In 2003, Chemero proposed a new definition which took elements from many previous formalizations by other authors (and indeed it was accepted by most of them later), which characterized affordances as "relations between the abilities of organisms and the features of the environment" (Chemero 2003:181), such that: Affords-Φ (feature, ability), where Φ is the afforded behavior [26]. He also integrated the issue of body metrics with the realization that while they indeed have a great influence on how agents

perceive affordances, it is only insofar body metrics determine to great extent the agent's abilities. Therefore, when we refer on the remainder to an entity or an entity's feature affording a function, it has to be understood that it does so only if the agent is able to perform the adequate behavior; an affordance can only happen if both the object and the behavior are present, as it is the relationship between both.

Despite Chemero's definition settled to a great extent the discussion about where affordances are, another crucial point of his definition reopened another debate which is not yet resolved, namely, that of how affordances are perceived or learned by an agent. Some authors, like Chemero himself and Murphy, support Gibson's initial proposal by arguing that affordances are "directly perceivable without any high level deliberation", and "they don't require memory, inference or reasoning, or interpretation of a scene." [114, 26]. Others, however, argue instead that what might seem automatic in biological agents is actually complex sensorimotor processing taking place in the brain, and that indeed all invariance detection inherently requires some degree of learning and processing [102, 94].

Indeed, later work by Eleanor J Gibson highlighted the importance of development in order to perceive incrementally complex affordances. Her view agreed with her husband (J. J. Gibson), in that the information required to detect affordances is real and always available to the agent. However, she advocated that perception consisted in the ability to extract relevant information to guide action, i.e., to perceive relevant affordances, and this ability improves through experience. Moreover, she argued that perceptual learning its not a passive process, as affordances depend on the agents capabilities, and hence, they develop together with the agent's skills and embodiment, as well as with its ability to generate and detect the appropriate perceptual information [50, 134, 51, 52, 2]. For example, infants are only able to detect the three dimensional shape of objects after they can produce coordinated visual-manual exploration [159].

These ideas were well accepted by the early community of robotic researchers attempting to implement models of affordances for robot control, as they provided a roadmap for affordance learning, and resonated with the then emergent ideas of developmental robotics and embodied cognition. Indeed, the necessity of roboticists to achieve working implementations of models of affordance learning, rather than purely theoretical as provided by psychology, provided them with useful insights that sparked new perspectives on the topic.

As a consequence, many recent attempts at formalizing affordances come from the

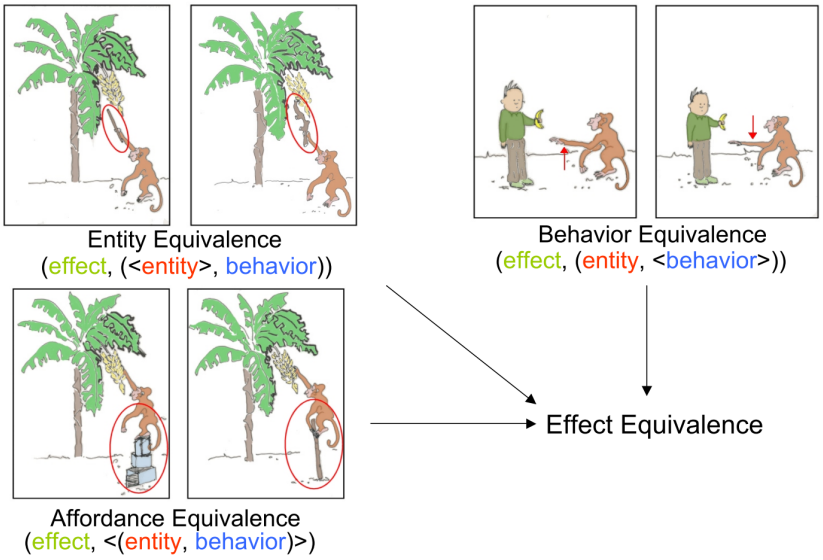


Figure 2.1.: Equivalence Classes according to Sahin's formalization of affordances (image from [20]).

robotic community. The first approach in this sense was the pioneer work by Fitzpatrick & Metta [48], which represented affordances as the possible effects that could be achieved on a certain object with a certain action, formalized thus as the tuple:

$$\{object, action\} \rightarrow effect \tag{2.1}$$

While in essence being very similar to Chemero's formalization (although substituting the term *features* by *object*, and *ability* for *action*), it made explicit the necessary existence of an effect as a consequence of the action on the object, which in turns enables using the affordance knowledge for planning.

Sahin et al. emphasised further the necessity of an effect element to characterize affordances, defining them as "an acquired relation between a certain effect and an (entity, behavior) tuple, such that when the agent applies the behavior on the entity, the effect is generated.", and proposed a tuple formalization [150]:

$$affordance = (< effect >, < (entity, behavior) >), \tag{2.2}$$

In his formalization, the terms *entity* and *behavior* were used instead of *features* or *object* and *ability* or *action*, respectively, to provide less restrictive meaning of each of the elements, and remarked the importance that all three terms corresponded to the perceptual representation by the agent. Moreover, these broader terms, and their grouping

into two subgroups, enable the discovery of equivalences among entities, behaviors or affordances with respect to their effect, as displayed graphically in Figure 2.1, and the use of these affordance equivalences for learning and planning. Also St. Amant & Horton proposed a definition of tool use based on their experience in robotic affordances that permeated the works of psychologists and anthropologists too [162, 156].

An ambitious approach was taken in more recent years with the introduction of *Object Action Complexes* (OACs), as a general model for grounding, learning and executing sensorimotor processes, to which affordances belong [49, 84]. Formally, an OAC is defined as a triplet *(id,T,M)*, where *id* is the OAC identifier, T a prediction function coding the systems belief of how the environment (and the agent) will change through the OAC (which hence represents the affordances), and M a statistical measure representing the success of the given OAC during a given time window. Each OAC is itself built up or generalized from execution of similar *instantiated state transition fragments* (ISTF). OACs enable consistent and repeatable hierarchies of behaviour to be learned, based on statistics gained during real-world interaction, that can also be used for probabilistic reasoning and planning. This framework ensures the grounding of an OAC in sensory experience by means of incremental verification and refinement. A different extension to the concept of affordance in robotics was proposed in [151], where they introduce affordance gradients (AG). The AG are in essence a continuous representation of the effect that agent actions could have on an object, which allow for generalization to unobserved actions.

In Section 6.1, we propose a novel formalization of the concept of affordances, which explicitly considers the embodiment of the agent, while simultaneously reducing the number of independent elements of the affordance tuple. More detailed analysis of the history of affordances, its controversies and interpretations can be found on Dotov et al. [40], Sahin [150], [92], and the recent survey by Jamone et al [76]. On the following, we will review how the concept of affordances has been applied in the field of robotics, and study with particular interest those works which have done so to model interaction with objects and tools, for its direct relevance with this PhDs topic.

2.2. Affordances in robotics

Prior to the application of the concepts of affordances itself, a few authors realized the connections between Gibson's ecological approach and the then new paradigms of

situated robotics and embodied cognition. Indeed, opposite to previous approaches to AI, where cognitive processes were modelled as symbolic computations, these new paradigms advocate for minimally cognitive, model-free based behaviors, where actions arise as dynamic interactions with the environment [3, 13, 25]. As Brooks famously put it in his seminal paper *Intelligence Without Representation* [18], "The world is its own best model". As some authors like Chapman and Agre realized, these concepts resonate perfectly with the ideas of direct perception put forward by Gibson, which states that affordances are directly perceived by the agent from the environment, and hence can be exploited without the need of mental representations. During the decade of the 90's, these connections were further extended by the field of active perception, which claimed that perception depends on the current intentions of the agent [8]. In other words, an object is not perceived in terms of its category, but rather in consideration of the actions it allows for the specific situation. The consequence is that perceived affordances are agent and effect related and so can not be studied from the environment alone.

Towards the end of that decade, some groups started applying affordance-based approaches for robot navigation. In the last nearly two decades since then, the field has extended to include not only methods for navigation, but also grasping affordances, object affordances, and recently, tool affordances. On the following, we will give an overview of the research in the first two topics and dwell into detail on the related work in the latter two, given their relevance for the present Thesis.

2.2.1. Navigation

The first tentative approaches to the application of affordances for robotics concerned robot navigation, whereby the robotic agent was able to perceive certain navigational affordances of the environment, e.g. transversability, displaceability, etc, by processing information such as the optical flow [41] or the potential field [38]. A more principled approach was presented by MacDorman [94], where the robot learned sensorimotor correlations, which were considered affordances, by extracting invariant reactions to its actions from the environment, and categorizing them. Once the categorization had been made and assigned its value, the robot used the learned sensorimotor model to predict the consequences of its actions and exploit the gained affordance knowledge. Despite the deserved merit of those papers for initiating the development of affordance-based models for robotics, they apply the concept of affordances in a quite loose way, either too general (as any sensorimotor correlation [94]) or too strict (predefined action rules

[41, 38]), and on objects whose state is not modified by the agent's action.

Following these pioneering studies, Cos-Aguilera et al. proposed a method in which a simulated Kephera robot learned the affordances of its environment by relating perceived regularities in the surrounding objects with the rate of success of the desired interaction when using that object. This knowledge was then connected to internal drives for behavior selection on an ecology based scenario [30]. Under the MACS project, Ugur et al. [178] applied Sahin's affordance formalization [150] on a wheeled robot to discover and learn the transversability affordances of different objects, which after learning allowed the robot to transverse a cluttered room without previous or explicit knowledge of object categories. Dogar et al. extended the previous work to generate goal-directed behaviors upon those objects from simple motor primitives [39]. Cakmak et al. expanded the study by using the learned behaviors for planning [20], while in [177], Ugur et al. added curiosity-driven learning mechanisms for efficient exploration. Within the same project, Lörken presented an approach in which the robot was also able to learn push and lift affordances of objects in the environment [92]. In [179], the previous ideas were integrated and the approach tested on more realistic scenarios with more objects and environmental elements.

Different affordance formalizations have also been tested on robotic navigation scenarios. For example, in [151], Sanchez-Fibla et al. benchmarked their concept of Affordance Gradients using a navigating robot to push objects to the desired goal. Min et al. [107] introduce affordances within the framework of dynamic programming, which allows task decomposition and dynamic affordance learning. In [57], distributed semi-Markov Decision Process were applied to represent affordances of the environment for a wheeled robot, with which it discovers the very existence of objects and entities in terms of their action possibilities, in an on-line and state dependent way.

2.2.2. Grasp

Due to the importance of pick and place applications in the robotic industry at tasks such as organizing warehouses, sorting products from conveyor belts or moving other mechanical pieces around in factories, as well as for holding and manipulating tools, effective and robust grasping has become one of the main lines of research in the field robotics. In general, the studies in this topic try to determine the best location and orientation that the robotic manipulator should apply in order to achieve a successful grasp of a target object, and in some cases, the grasp trajectory too.

However, as pointed out in [76], not all these studies fit well under the label of affordance-based approaches, as many of them determine the grasp configuration based on purely geometrical analysis of the objects to be grasped and the manipulator, and require precise models of both to be successful [155, 14, 149, 60]. Yet, there is a body of literature in which "graspability" of objects is learned from the robot's own experience in a self-supervised manner, typically by associating low-level object features perceived by the robot with the probability of successfully grasping the object with different grasp configurations. However, grasp affordances should only be considered when the goal is itself to grasp the object, and not any further action that might be accomplished afterwards for which grasping the object is a requisite (these situations will fall into the object or tool affordances categories, reviewed below). Relevant approaches on affordance-based grasping that match this interpretation have been published by Oztop et al. [125], Barck-holst [10], Montesano et al. [111], Popovic et al. [36], Detry et al. [35] or Ugur et al. [186], to name but a few.

Interestingly, many studies, specially recent ones, use similar self-supervised data-driven approaches without explicitly referring to them as affordance based [88, 90, 135], while others do present their studies as affordance based although the proposed approaches are robot independent, usually based solely on computer vision methods [163, 61].

2.2.3. Objects

This direction of research in affordance-based robotics deals with discovering the affordances of small objects with which the agent can interact. Related affordances are for example pushability, rollability, dragability as well as graspability, understood as the general potential of the robot to perform these actions on the objects. The typical scenario consists on a upper-torso robot with grasping and visual capabilities, and objects lying on a table in front of it, although detached camera/robot-arm systems have also been used.

The first study concerning object affordances, in the way we defined them above, was published by Fitzpatrick & Metta on 2003 [48]. In their study, a robot was able to learn "movement" affordances such as rollability, pushability, etc, of several objects by randomly executing a set of actions on them and observing the results. Their approach introduced several elements in the study of affordances that had been very influential in posterior studies on the topic. On the one hand, it applied random actions (babbling)

as a way to retrieve data to guide the learning process. On the other, it made use of the formalization described in the previous section: $\{action, object\} \rightarrow effect$, which allowed predicting the possible effects of an action repertoire, and thus to select the best action on an object to achieve a desired effect. As was detailed in the previous section, the concept of affordances has gone through many formalizations and reformulations since, but the $\{action, object, effect\}$-tuple formulation has remained a cornerstone in robotic affordance research to our days.

Following up on Fitzpatrick's approach, Stoytchev conducted a similar experiment in order to study binding (grasping/pulling) behaviors [168]. His approach, however, consisted in populating an "Affordance Table" with those sequences of actions, taken from a set of exploratory behaviors, that generated the desired binding effect. Geib et al., on the other hand, tackled object affordances using the OACs approach [49]. Despite the importance of these studies in laying the groundwork on which many posterior papers would rely, their main drawback was that they learned affordances of labeled objects. This, on the one hand, means that the acquired knowledge could not be generalized to previously unseen objects, and on the other, conflicts with the original postulation that objects are recognized in terms of their affordances, without the need of prior categorization.

In order to overcome this issues, Montesano et al. proposed a method in which basic geometric object features were connected through a Bayesian Network (BN) to the possible actions and the observed outcomes of the considered interaction [112], as represented in Figure 2.2. This approach provided several advantages over previous studies: On the one hand, it was based on object features such as color, elongation, size, etc, which allowed affordances to be estimated directly from observing the objects, and generalized to previously unseen objects based on their attributes. Moreover, the application of the Bayes Theorem to the conditional probabilities modeled in the network enabled not only to predict effects based on given object and actions, but also to infer any of the elements (action, features or effects) based on the values of the rest, in a probabilistic manner.

Subsequent refinements on Montesano's approach improved the network parameter learning [113], allowed the model to cope with partial observability of sensor data by using Gaussian Mixture Models to cluster sensors and effect variables [124], studied the influence of the feature extraction phase [67], and improved network structuring [58]. Moreover, it has been applied not only for single object affordance learning, but also for imitation learning [91], for multi-object and two arm robot relational affordance learning

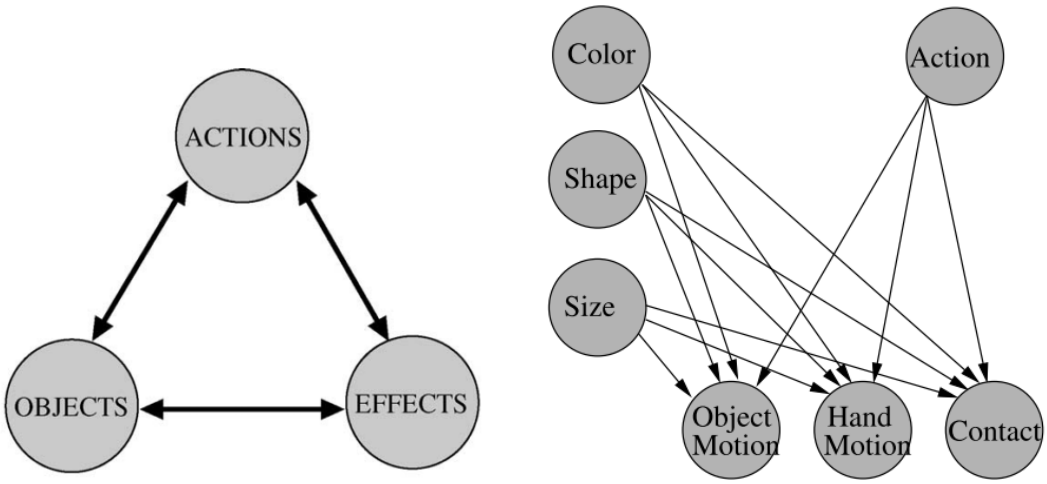


Figure 2.2.: Affordance representation proposed by Montesano et al. in [112]. (a) Affordances as relationships between Actions, Objects and Effects. (b) Bayesian Network model proposed to implement affordances.

by embedding Statistical Relational Learning in the affordance learning algorithm [110, 109], as well as for learning tool affordances [59] (as will be reviewed in detail in the next Section). Also, in recent work by Antunes et al., the output of the BN has been applied for planning using the PRADA probabilistic planner [6].

Sahin's formalization, initially used for navigational affordances as discussed in the previous section, has also been applied for object affordances. A method for enabling a robot manipulator to learn affordances in an unsupervised way, and generalize its knowledge to new objects in terms of the entity and affordance equivalences, was introduced by Ugur et al. in [185]. This approach has since undergone many modifications to extend and refine its capabilities: Support for the discovery of new effect categories [180], automatic actualization of behavioral parameters for goal oriented plan generation [181], self-structuring of the learning process [182], bootstrapping knowledge of simple affordances to more complex ones using intrinsic motivation [187, 184], and use of affordance knowledge for plan generation and execution [183].

The main disadvantage of the methods described so far is that they either use predefined effect categories, or require offline clustering with all the available data in order to discover the available effect categories. In order to overcome this drawback, a few studies have proposed approaches capable of updating their knowledge incrementally, more suited for life-long and developmental learners. For example, in [158], and adaptive hierarchical taxonomy is introduced to enable incremental and hierarchical learning

of affordances. In [141, 142], object and effect features are mapped into separate Self-Organizing Maps (SOMs), which are linked together through Hebbian connections that strengthen based on the their co-occurrence given an action. In [56], navigational and object affordances are learned incrementally using Semi-Markov Decision Processes to model the sequences of actions that lead to their activation.

A few studies have also considered multi-object scenarios, an intermediate step towards the analysis of tool affordances. Moldovan et al., cited above [110, 109] coupled Statistical Relational Learning methods with the Bayesian Network approach proposed in [112] to perform inference about more than one object. Fitchl et al. [46, 45] studied how to represent the necessary spatial preconditions that enable certain multi-object affordances. Ugur et al. focused on the task of objects stacking and proposed a method for learning mutual "stack-ability" among objects and inferring such affordances from simpler ones previously learned [183, 184].

2.2.4. Tools

In the studies described so far the robot used its own manipulator to interact with objects, and observed the effect of its actions on them. In order to differentiate multi-object affordances from tool affordances, we consider that tools correspond to objects or elements, different of the robots own manipulator, that the action is performed *with*, rather than *upon*. To the best of our knowledge, the first published papers that studied the functionality of a set of tools that could be operated by a robot (a robotic arm in this case), were published by Krotkov [83] and Bogoni [16, 17]. Although they were not explicitly framed within the Theory of Affordances, they put forward some concepts that would be a commonplace in later affordance-based studies, such as the application of features to characterize the elements of the interaction, and the observation of the effect after an action was performed by the robot.

In [15], Bicici and Amant theorized about the requirements for tool use in robotic scenarios, focusing on which characteristics of tools are more relevant for their functionality, and explicitly suggesting the Theory of Affordance as an adequate frame to tackle the problem. A decade later Guerin et al. published a complementary study where they proposed a roadmap for developmental tool use learning in robots based on infant development. Their proposal is based on Gibson and Piaget's theories, supported with insights from different fields such as neuroscience, planning and developmental psychology [65].

Regarding actual implementations, the first study to conduct a tool use learning experiment explicitly based on the affordance approach was published by Wood et al. [200], where a MMC neural network is applied to allow an AIBO robot to pick a stick of variable length and use it to reach a ball. However, the focus in this work was to study the adaptation of the body schema required to apply the tool as the robot's end-effector, rather than on the different effects that the tool might achieve depending on the action performed.

Tool affordance learning in robotics, considering the relationship between tool, action and effect, was pioneered by Stoytchev in his 2005 study [168], which was extended in his dissertation [169] and later by Sinapov in [158]. In this approach, for each tool, affordances were learned as a list containing the actions performed and the probability of success in moving an external object. However, affordances are learned for a set of labeled tools, so the approach does not allow generalization for other tools, regardless how similar. A similar approach was taken by Tikhanoff et al. [174], but on their approach a geometric reasoner is queried in order to determine the feasibility of exploiting the learned affordances on a given scenario in function of the tool's length.

As was the case with object affordances, authors soon started describing tools in terms of their features rather than labels. In [70], Horton focused primarily on the discovery of tool contour features that matched particular targets (the way the cross in a screwdriver matches the cross in a screw) and hence afford their interaction. This work allowed generalization among tools based on their geometrical similarity, but affordances were not learned or updated. Jain & Inamura, on the other hand, applied the concept of *functional features*; features that describe the tool regarding its action possibilities rather than its appearance. They predefined a set of these features based on geometric properties (corners, bars, etc), which they assumed were known by the robot, and used the Bayesian Network proposed by Montesano [112] to learn the correlation between the features, the action repertoire of the robot, and the effect achieved in a target object [74]. Their follow up publication extended their results and added imitation learning to the scenario [75]. Gonçalvez et al. made functional features of tools and objects automatically detectable by the robot by using simple 2D geometrical features (length, area, size, etc) extracted from vision [59], which were linked to actions effect also through a Bayesian Network. Dehban et al. applied the same paradigm but substituted the Bayesian Network by a Denoising Auto-encoder, which, contrary to the Bayesian Network, allows for real value input and online learning [34]. On the method we proposed in [97], detailed in Section 6.2, a larger set of candidate functional features from the 2D

contour of the tool was applied in such a way that the grasp position of the tool was also considered when learning tool affordances.

Some recent papers have extended vision based approaches with the aim of learning tool affordances by applying depth information, which reflects better the geometry of the tool than 2D images. Myers et al. collected an extensive dataset of RGB-D images of tools with associated pixel-wise human labeled affordances, and applied state-of-the-art supervised classifiers to predict the affordances from a combination of curvature, normals and depth extracted from super-pixels on the depth data [115]. The same dataset was used to train end-to-end Deep Convolutional Neural Networks within an expectation maximization framework by Srikantha et al., which enabled weakly supervised learning [161]. Zhu et al., used RGB-D videos of human tool use demonstrations to infer the physical concepts involved in specific tasks, which were then applied to predict consequences of tool use [208]. However, similar to some of the studies for robotic grasping we discussed in Section 2.2.2, these papers based purely on computer vision approaches lack the robot component, crucial to the definition of affordances. Therefore, they could only be regarded as learning human affordances, which later might, or might not, be transferable to robotic platforms.

A few other authors have proposed affordance learning methods based on full 3D representations of the tools, which are potentially more robust either 2D or RGB-D to variability induced by occlusions and perspective. In [1], Abelha et al. estimated the suitability of a set of household objects as tools for a set of given tasks by fitting the object's superquadric model to the one of the canonical tool for that task. In [96], we introduced the Oriented Multi-Scale Extended Gaussian Image (OMS-EGI) feature vector, a holistic descriptor extracted from a tool's 3D model. Unlike previous tool representations, the OMS-EGI is robot-centric insofar as it depends not only on the tool's geometry, but also on how the robot is grasping it. As detailed in Sections 6.3 and 6.4, this descriptor was used to learn the affordances of a large set of tools for a set of drag actions performed with the iCub robot.

The topic of developmental learning of tool use in robots has also been tackled by other authors using approaches not explicitly considered related to affordances. For example, Nishide et al [122, 171], studied robotic tool use with a special emphasis on using plausible neurological models, although focusing more on how tools extend the body schema than in the exploration of their affordances. To this end, they applied a SOM to learn the tool features directly from an image, and Multiple Timescales Recurrent Neural Networks to model the robots homunculus and study its variation when using

a tool. On the other side of the spectrum of biological plausibility, Brown et al. [19] proposed a method to enable a simulated wheeled robot to learn tool use by demonstration by means of inductive logic. On their method, demonstrated actions are segmented into action sub-goals, which are defined with STRIP operators to allow their reusability for planning. The same approach has been recently applied in [199], extended to use a Baxter robot and knowledge transfer between simulation and the real robot.

Readers interested in a more comprehensive review on affordances in robotics should refer to the recent surveys by Min et al. [108] and Jamone et al. [77], the latter of which also includes extensive review of affordance related studies from psychology and neuroscience.

2.3. Functional features

One of the central pillars of Gibson's Theory of Affordances was the concept of *direct perception*, which states that affordances are perceived by agents *directly* as invariant properties of the environment:

"The perceiving of an affordance is not a process of perceiving a value-free physical object to which meaning is somehow added in a way that no one has been able to agree upon; it is a process of perceiving a value-rich ecological object." (Gibson 1979:140 [54, 55])

, or in simpler terms:

"[...] I now suggest that what we perceive when we look at objects are their affordances, not their qualities."(Gibson 1979:134 [54, 55])

Before these concepts permeated the robotic community, and indeed, before the affordance based approach was fully formalized, similar ideas had been proposed independently by some robotic researchers such as Bogoni et al. [17]. However, their focus was on determining a set of features that correlates with the functionality of environmental elements in order to predict the effects of their use, rather than on the psychological aspect of how these functionalities are perceived:

"To recover the functionality of an object it is necessary to devise interactions to recognize the relationships between the intrinsic properties of an object (e.g., geometric, material, kinematic and dynamic), its environment and the manner in which it is em-

ployed. [...] The analysis of their properties allows us to identify what can be labeled as the *functional features* - features which make the object suitable for a specific task." - (Bogoni 1995: 2) [17].

Considering both approaches, functional features can be understood as an implementation of the concept of direct perception, whose purpose is to provide a means from which functionality, i.e., affordances, can be represented and predicted in artificial systems.

Yet, the problem is that the functionality of any entity is determined by its physical characteristics (geometry, stiffness, weight, etc), which are often independent from its visual appearance. Therefore, the way in which entities (environment, objects, tools, etc.) are represented in affordance learning approaches is crucial in determining how well these methods will be able to generalize the learned affordances to new entities.

Hence, on the remainder of this section we will review the most relevant functional features proposed in previous affordance-based studies in robotics. Additionally, we will briefly survey other descriptors proposed in other fields, which could nevertheless be applied for as functional features for affordance representation.

To the best of our knowledge, the study by Montesano et al. was the first affordance-based approach to apply features in order to describe objects in a way that would allow the system to predict their affordances [112]. Accordingly, the features used in this study were simple geometrical descriptors representing discretized values for color (green, yellow, blue), size (small, medium, big) and shape (ball, box). Follow-up studies improved the discretization of the feature values by adding automatic clustering and soft labels [113, 124], or allowed for continuous values by using auto-encoders instead of the original BN [34], but the represented characteristics remained unchanged. In [58], the authors showed that the same features can also be applied to predict tool affordances. In [97], we proposed a much larger set of similar 2D contour based features (detailed in Section 5.2), and applied with the aim of implicitly representing the tool orientation too.

Almost simultaneously to Montesano's seminal work, several studies by Sahin's group [178, 20, 39], applied a method to learn the transversability affordance of objects in the environment based on normal histograms and distance features extracted from range images. These components, with minor quantitative changes in implementation, have remained mostly invariable throughout Ugur's extensive work in affordances [185, 179, 180, 182].

In [141], Ridge et al. applied a combination of both approaches, using range data to describe the curvature of objects, and a set of 2D image features such as, area, eccentricity, axis length, etc., to describe their general shape. In later work, he expanded the information coming from the range sensors by adding cloud-part based information, namely quadrant centroids and planar orientation [142].

In very recent years, a few authors have proposed sets of features that rather than a list of specific descriptors, represented objects in a holistic manner. In our paper [96], developed as part of the current Thesis, we proposed the OMS-EGI descriptor in order to represent tools, which consisted in the concatenation of voxel-wise surface normal histograms, as will be described in detail in Section 5.3. In [144], merging ideas from the OMS-EGI and his own previous work, Ridge et al. proposed an object representation suited for affordance learning which comprised local information, from voxel-wise surface normal histograms; and global information, from cell centroids, curvature estimates and point count. Yet another approach comes from several authors who have proposed methods where state-of-the-art computer vision features were used in order to predict affordances of objects. These approaches fall mainly into 3 categories:

- Category prediction: Visual features are applied to categorize objects, to/for which different affordances have been assigned/learned. For example, in [79], objects are classified using HOG features, and the object categories linked with the affordances learned from recorded human action by means of CRF. In [24], AlexNet features are used to classify object images retrieved from Google with a noun-verb search, where the resulting images of the search provide the input, and the verb, the corresponding affordance label. The main drawback of this method, is that it prevents specific features to be recognised as "affordance generators", as for example the thin edge of some coins could afford screwing, although the coin would never be categorized as a screwdriver.

- Attribute prediction: Visual features are trained to predict functional features (also referred to as *attributes*) which in turn predict affordances. For example in [67] SIFTs and texture and color histograms were applied to predict size, material, color, weight, shape, etc, while in [170] the same functional features were predicted from codebooks obtained form sparse coding of RGB-D data.

- Direct prediction: Visual features are used to directly predict the affordances of an object. For example, in [63, 64], SOM-based sparse coding features from RGB-D data were applied to discriminate container from non-container objects. In [78],

Human given affordance labels of office objects were predicted by means of 3D geometry features computed from the pointcloud reconstruction of the scene. In [146] multi-scale CNNs are trained at different resolutions with RGB-D data from an affordance database. Myers et al. proposed a part based affordance detection system based on 3D features extracted from a superpixel segmentation from objects [115]. They used a combination of depth, normals and curvature features. In order to train their classifiers, a database of pixel-wise annotated affordances was also presented. Using the same database, the approach proposed in [161] learned the features from RGB-D images by training end-to-end a Deep Convolutional Neural Network, while in [121], they trained an encoder-decoder DCNN with HHA features (horizontal disparity, height and, and angle between pixel normals and inferred gravity). In [1], Abelha et al. based affordance estimation on full 3D models of tools, by estimating the suitability of a set of household objects for a set of given tasks by fitting the object's superquadric model to the one of the canonical tool for that task. In all these studies, the affordance labels used for supervised learning were not obtained by robot exploration but instead given by a human experimenter. However, in [121] an actual humanoid robot was used to verify some of the predictions for grasping affordances.

An important aspect regarding features for affordance learning, is whether they should be invariant with respect to the state of the described object, or on the contrary, depend on it. While in the field of navigational affordances it is quite clear that only the second option provides relevant information [178, 57], when considering tool or object affordances, the solution is not so clear. For example, [112, 67, 58] – and naturally all the computer vision based studies – apply features that depend solely on the object's geometry, independent of its position. On the other hand, when features depend on the state of the object, several options arise with respect to the frame of reference. In [179, 182], the object features depend on its position and orientation, with respect to the robot's cameras, which are fixed. In [143, 144] the reference frame is computed relative to the action exerted on the object. By contrast, in our own papers [97, 96, 98], tool features are computed with respect to the reference frame of the hand holding the tool.

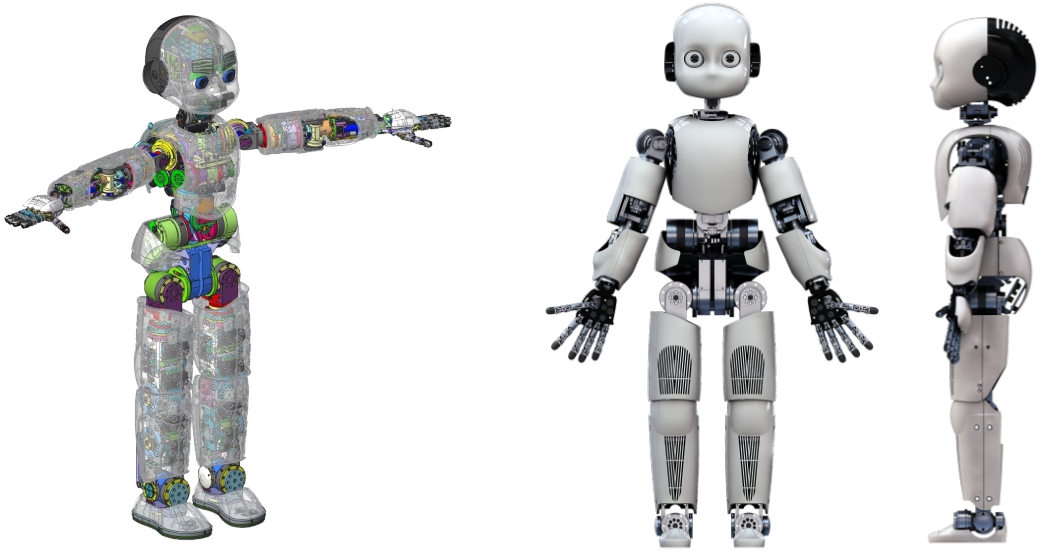
Despite some attempts to compare the prediction performance obtained with robot-relative features against other "invariant" features [179, 143, 144], we believe that this is an ongoing debate which will need many more experiments and paradigms to be satisfactorily resolved.

Besides the cited papers, there are many other proposed features that have not been

tried within an affordance learning scenario, but that have descriptive characteristics that could also provide relevant information for affordance learning. One group of this kind of features would be 2D hierarchical features, given that affordances are generally determined by characteristics of objects or tools at many different scales. Further advantages of hierarchical feature extraction algorithms are that most of the existing implementations provide invariant hierarchies [47, 153, 139, 86] as well as unsupervised or weakly supervised feature learning [139, 117, 87, 86], both very desirable properties. Another group is that formed by global 3D features, which by describing the geometry of objects should correlate properly with their physical affordances. Examples can be the DESIRE feature [195] or the more recent Global Structure Histogram [95]. Other local 3D features, such as the ones proposed in [85, 201, 137] could enable end-to-end 3D feature learning for affordance applications.

3. Robotic Platform

3.1. The iCub robot and its simulator



(a) Detailed CAD model.

(b) iCub front and side.

Figure 3.1.: iCub Humanoid Robot.

All the experiments presented in this Thesis were carried out using the iCub humanoid robot [106, 105, 118] and its simulator [173]. The iCub robot is an open source cognitive humanoid robotic platform for collaborative research in human cognition, human robot interaction, and embodied artificial intelligence. It was the result of the RobotCub European project, which aimed at "testing and developing the embodied cognition paradigm through the creation of a child-like humanoid robot: the iCub, [which] will act in cognitive scenarios, performing tasks useful to learning while interacting with the environment and humans". Two detailed models of the iCub can be observed in Figure 3.1.

In order to better model the characteristics of a learning child, the iCub kinematics

were designed to mimic the body of a three and a half year old child, both in the number of Degrees of Freedom (DoF) and in their location. As a consequence, the robot is 104 cm tall, weights 22-25 kg (depending on the version) and has 53 DoF, most of them (41) in the upper torso. Its 9 DoF hands and 2 DoF wrists, allow for dexterous manipulation and different grasping configurations. Full head movement is provided by 3 DoF in the neck, while tracking and vergence behaviors are supported by the extra 3 DoF controlling the eyes. The torso comprises 3 DoF in order to increase flexibility and enlarge the interaction workspace, while legs have 6 DoF each, and are strong enough to enable crawling, balancing on one leg, and bipedal locomotion. Joints are actuated with brushless and DC motors, controlled by embedded dedicated boards, which are interconnected through a local bus. These boards generate motion from higher level commands, and can perform position control with trajectory interpolation, velocity, and torque control, depending on the selected low-level interface.

From the sensing point of view, the iCub has cameras in the eyes for vision, which also enable depth estimation through disparity computation, microphones for recording sound, and full body tactile skin system (with more than 4000 sensing units) to provide tactile feedback. Proprioception is procured by means of an IMU situated in the head, motor encoders in each of the joints, and 6 F/T sensors in shoulders, hip and ankles. The local "brain" of the iCub consists of a small form factor PC104 mounted inside the head of the robot. This computer acts as a bridge between the local boards and the external computers that perform the heavy computation required to implement robot behaviors, to which it is connected via gigabit Ethernet or Wireless.

The iCub simulator provides an open-source environment in order to reproduce the physics and dynamics of the robot and its environment [173]. The simulated robot model is structured as multiple rigid bodies connected via joint structures, whose design parameters were replicated from the real robot specifications for maximum accuracy. All the sensors available in the real robot are available on the simulated model too, although perfect accuracy cannot be guaranteed on certain scenarios. The simulation of the robot's physical interaction with objects is achieved with the Open Dynamic Engine (ODE), which consists of a high performance library for simulating rigid body dynamics and collision detection. Rendering, on the other hand, is carried out with a combination of OpenGL and SDL.

3.2. YARP middleware

The iCub software is structured as modules that communicate with each other using YARP (Yet Another Robot Platform) middleware. YARP consists of a set of libraries which promote code reuse and interoperability by abstracting software modularity and hardware interface.

To achieve software modularity, YARP implements a set of protocols which address inter-process communication based on the *observer pattern*, a variant of the publish-subscribe paradigm. Specifically, the `Port` and `BufferedPort` protocols decouple producers and consumers, in a synchronous or asynchronous way, respectively. When communication requires reply, it can be accomplished using `RPC -Client` and `-Server` protocols. This way, *modules* (architecture components of the iCub software), are runnable executables which export a certain interface based on `(Buffered)Port` and `RPC` objects. Groups of interconnected modules that execute a specific robot behavior are referred to as *applications*.

The abstraction over the hardware interface is achieved by YARP through the definition of a set of C++ interfaces for classes of devices (sensors, motors, etc), that wrap their native code APIs. This abstraction translates into the possibility to change or update hardware components (which happens often in robotics), by modifying only the related API calls.

Moreover, YARP provides multi-platform support based on the ACE library and CMake, and implements a system of plug-ins that allows users to write custom protocols and interact with other systems. For example, the plug-in System has been used to allow interoperability with ROS, QoS by channel prioritization [127], and the integration of Port Monitors, which allow accessing data passing though a connection from/to a port for monitoring, filtering, and transforming [128].

Code reuse is also facilitated by the `ResourceFinder` class, which enables the existence of different configuration files for the same module, so it can implement different behaviors depending on the scenario.

With these elements, YARP successfully addresses the separation of concerns advised for middleware: Computation, by enabling multi-machine distribution; Communication, by enabling agnostic data exchange among modules; Coordination, through the use of port monitors and `RPC` calls; and Configuration, through the use of `ResourceFinder` to reconfigure modules in function of the application scenario.

3.3. Relevant modules

As we saw, any software architecture built on top of YARP middleware is basically composed of interfaces, modules, and the connections between these modules. Due to the longevity and openness of the iCub platform, a large number of modules and applications are already available on the iCub repositories[1], which serve for a variety of purposes, as outlined below.

The core iCub software infrastructure, named `icub-main`, includes of a set of modules which get installed by default on the iCub setup and serve to control its basic functionalities. Some function most of the time in the background, being transparent to the iCub user, such as the `robotInterface` module, which executes the start-up of the robot, or the modules for calibration and image grabbing from the cameras. Others, however, need to be directly interacted with. In the development of the current PhD the most relevant `icub-main` modules were the following:

- Gaze Controller: By exporting the functionalities implemented in `iKinGazeCtrl` module, the YARP Gaze Controller interface provides an abstract layer to control the iCub gaze in a bio-plausible way, moving the neck and the eyes independently and performing saccades, pursuit, vergence, OCR (oculo-collic reflex), VOR (vestibulo-ocular reflex) and gaze stabilization relying on inertial data [145].
- Cartesian Controller: The YARP Cartesian controller interface exports the functionalities of the `iKin` modules, thus enabling the control of the arms and the legs of the robot directly in the operational space (3D position and orientation of end-effector) rather than in the joint space (configuration for all the joints in the corresponding kinematic chain) [133].
- `actionsRenderingEngine`: This module combines and coordinates multiple libraries and lower-level modules from the iCub repository (such as Gaze and Cartesian controllers) in order to allow the execution of a set of predefined actions (eg. take, push, look, expect, etc), with given parameters or target (coordinates, tracked element, etc).

Moreover, there are some more recent and task-specific modules and applications which are therefore not part of the `icub-main` software core, but that nevertheless have been used and/or modified for the development of this Thesis. All of them can be found

[1] https://github.com/robotology

in the `iCub-Contrib` software repositories.

- Stereo-vision: The stereo-vision modules provide libraries and functionalities which enable depth estimation from the iCub's binocular vision by computing the disparity between the images coming from the cameras in its eyes [131]. The estimated depth perception can be used for tracking, position estimation and, most importantly for the current work, 3D reconstruction.

- KARMA: The KARMA application consists in a set of modules which were used to implement the affordance learning experiment published in [174]. The implementation of tool actions and tooltip extension provided in this application served as the basis for the development of similar functionalities that were applied during the present Thesis.

- Segmentation: The segmentation repository contains a set of modules which perform segmentation of incoming images based on texture analysis [123], graph methods [43], disparity [131], and edge detection [29], respectively. As we will see below, proper segmentation was a crucial step in order to proper characterize tools for affordance learning.

- onTheFlyRecognition: A very recent module which enables "on the fly" learning and recognition of object classes from vision by combining feature extraction from an off-the-shelf Convolutional Neural Network and an online trained linear classifier [130].

Finally, there were some modules that were implemented during the development of this Thesis in order to provide some functionality not yet available that, without being specifically related with the topic of this Thesis, was required to accomplish some of the desired higher order behaviors:

- Disparity based blob detector (`dispBlobber`): This module takes as input a disparity map (grayscale image) and provides as output the closest (brightest) blob, which is generally the object or proto-object of interest in the scene [131]. This simple segmentation algorithm was applied in this Thesis to segment tools in the robot's hand, but has also been applied for object learning scenarios [130]. An illustration of its function can be observed in Figure 3.2.

- segmenation-to-pointcloud (`seg2cloud`): This module combines disparity and segmentation information in order to retrieve partial view 3D pointclouds of desired segmented objects. An instance of this process can is displayed on Figure 3.3. As

Figure 3.2.: Example of the `dispBlobber` module working on the `onTheFlyRecognition` application. The module receives a camera image (left) and the depth estimation based on disparity (center), performs thresholding and filtering, and returns the blob corresponding to the closest object in the scene (right). This blob can be used, as in the example, to crop the image surrounding the object of interest.

Figure 3.3.: Example of pointcloud reconstruction using the `seg2cloud` module. The module receives a camera image (left) and corresponding depth estimation (center). Then, after a region of the scene has been segmented (using either color, depth, graph-based or hand selected regions), the depth information is transformed into 3D coordinates by means of the robot kinematics and the corresponding set of 3D points returned as a pointcloud (right).

we will see in Chapter 4, this partial pointcloud reconstructions were very useful to estimate tool orientation, tooltip, and for automatic tool modelling.

The modules and scripts which implemented functionalities directly related to the Thesis topic, i.e. tool representation and affordance learning, will be detailed when their application is described in the following chapters.

3.4. External libraries

OpenCV: OpenCV is an open source library for computer vision and machine learning with implementations in C, MATLAB, Java, and C++, being the latter the one used in this Thesis. It provides over 2500 optimized algorithms, which allow a wide range

of tasks, such as face recognition, object identification, action classification, tracking, depth estimation from stereo cameras, image stitching, scene recognition, etc [71].

Point Cloud Library (PCL): The Point Cloud Library is an open and large scale project for 2D/3D image and pointcloud processing [148]. The PCL framework implements numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting or segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints or compute descriptors to recognize objects, to name a few.

A *pointcloud* is a data structure commonly used to represent three-dimensional data. In a 3D pointcloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface.

SOM Toolbox: The SOM Toolbox is a software library for MATLAB that implements several functions related directly to the application of Self-Organized Maps. The functions present in this Toolbox have been contributed by a large number of professionals employed in the Laboratory of Information and Computer Science in the Helsinki University of Technology. The SOM Toolbox is free software licensed under GNU General Public License, so it can be freely acquired at the Toolbox's website. A complete manual can be also found at the same website [192, 191].

MATLAB Neural Network Toolbox: The MathWorks, Inc. creators of MATLAB, have also created an all-purpose neural network environment to provide tools for designing, implementing, visualizing, and simulating neural networks. Neural Network Toolbox software provides comprehensive support for many proven network paradigms, as well as graphical user interfaces (GUIs) that enable the user to design and manage the desired networks. This Toolbox, unlike the previous ones, is not freely available.

4. Tool Incorporation

4.1. Overview - Architecture

As stated in the Introduction, we believe that one of the great advantages of building humanoid robots such as the iCub is their potential to take advantage of available tools and devices of human environments. Indeed, robots capable of autonomously achieving a certain level of competence with tool use would be far more versatile than those limited by the design of their effectors or manipulators.

However, a critical problem present in most studies of tool use in robotics is that the actions are performed without considering the geometry or pose of the tool. Instead, most experiments apply standard grasps and assume pre-defined kinematic end-effector extensions that do not take into account the particular pose of the tool in the robot's hand.

Tikhanoff et al. made an important contribution in this sense, by introducing a method with which the robot was able to automatically estimate the position of the tooltip, and use this information to extend the robot kinematics [174]. During the development of the present Thesis, we improved their method qualitatively and quantitatively, by implementing a series of interconnected methods that enable autonomous, fast and reliable estimation of a tool's geometry, reach and pose with respect to the iCub's hand, in order to attach it to the robot's kinematic chain.

We designate this process as *tool incorporation*, in order to refer to the two acceptations of the word *incorporation*: that of including, as in the act of grasping the tool and thus physically adding it to the robot's body; and that of embodiment (literally, *in-corpore*), referred to the process of building a representation of the tool with respect to, and included in, the robot's body representation. Figure 4.1 displays a diagram of the process' workflow, and provides a detailed description of each of the steps.

In brief, the goal of the tool incorporation process is to obtain a pointcloud repre-

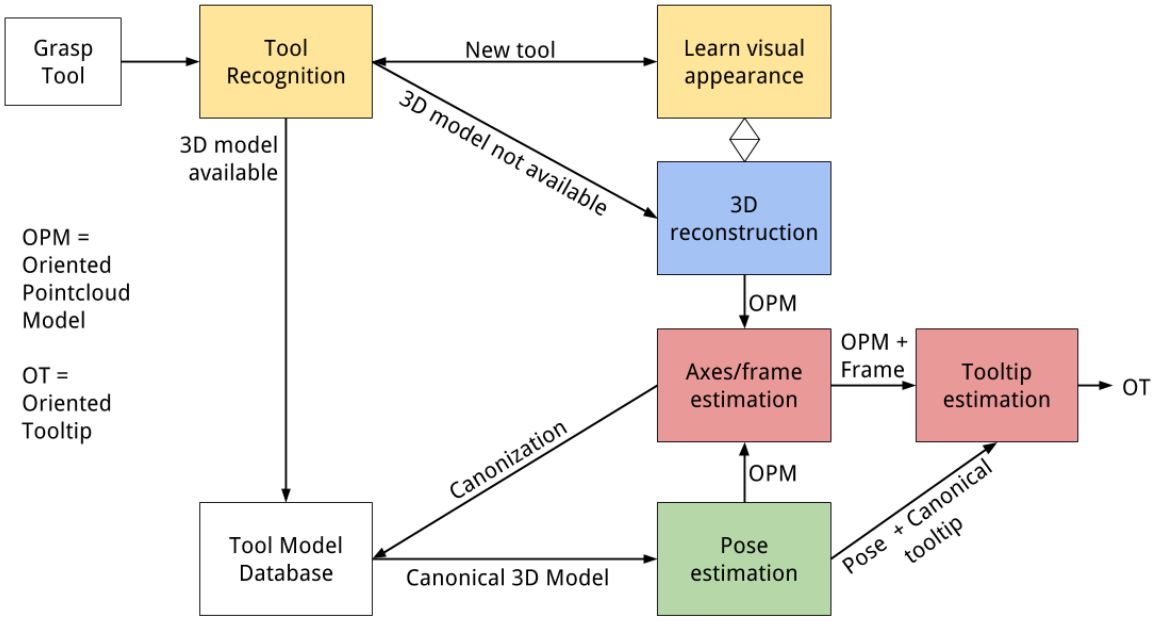


Figure 4.1.: Tool incorporation diagram. The process of tool incorporation begins when a tool is handled to the robot. Once it holds the tool, the first step is to attempt to recognize it, or learn its visual appearance if recognition fails (see Section 4.2, yellow in the diagram). In either case, if there is no available 3D model of the tool, the iCub will perform an autonomous exploration process to reconstruct the tool's 3D geometry making use of its stereo-vision capabilities (see Section 4.3, blue in the diagram). If no representation of the tool is available to the robot, (its visual appearance or its 3D model), the exploration process can be performed to obtain both simultaneously. As a result of the 3D reconstruction process, an oriented pointcloud model will be obtained, that is, the tool's pointcloud representation oriented as the actual tool with respect to the robot's hand reference frame. In this case, if either its pose or tooltip are required explicitly, they can be computed by means of the tool frame estimation method (see Section 4.4, red in the diagram). Once the pose is known, it can be inverted to set the model to its canonical pose (thus called canonization), in order to be saved in the database of tool models. By contrast, if the tool is recognized and its 3D model is available, it will be loaded from the database in its canonical pose. In this case, its pose and tooltip can be estimated by alignment with a partial reconstruction of the actual tool (see Section 4.5, green in the diagram).

sentation of the tool being held by the robot, as well as its position, orientation, and tooltip location. To that end, the first step is to recognize the tool, or learn its visual appearance if it has not been seen before. This process is detailed in Section 4.2, Tool Recognition. In either case, if the tool model is unavailable, the robot will use its stereo-vision capabilities to reconstruct it, as described in Section 4.3, Tool Reconstruction. In this case, the tool tip will be determined through the estimation of the tool's reference frame, following the methods and definitions introduced in Section 4.4, Tool Frame and Tooltip estimation. By contrast, if the model is available, it will be loaded from memory in a canonical pose, and its pose and tooltip estimated by aligning it to a partial reconstruction of the actual tool, as explained in Section 4.5, Tool Pose estimation.

In addition, the development of the constituent modules of this process led to the formalization of some important concepts in order to enable their proper mathematical treatment, namely *tool frame reference*, *tool planes* and *tool axes*; *tool pose* and *canonical pose*; and *tooltip*, which will be described in their corresponding sections.

4.2. Tool recognition

In the scenarios developed in this Thesis for learning tool affordances, we always assume that the robot starts each trial with a given tool in its hand. Therefore, if the tool is known, the iCub should recognize it in order to load its 3D model for further processing. Otherwise, if the tool is unknown, the iCub should explore it in order to learn its visual appearance and 3D geometry, so that they are available for future attempts. In this section we describe the method applied for the first of the tasks, recognition and learning of the tool's visual appearance. The methods for reconstruction of its 3D geometry will be explained in Section 4.3.

In particular, we applied a method based on the techniques for visual object recognition described in [130]. Briefly speaking, in the setup proposed in that paper, in order to train a new object to the system, the experimenter held the object in front of the iCub's cameras and verbally communicated the label of such object to the robot. Then, the robot used a segmentation algorithm based on disparity (`dispBlobber`, described in Section 3.3), which cropped out the closest object (the one to be learned) from the background and fed the stream of cropped images to a pre-trained CNN (AlexNet trained on imageNet [82]). Finally, the output of the last fully connected layer of the CNN was used as the input for a linear classifier, which associated the high-level features pro-

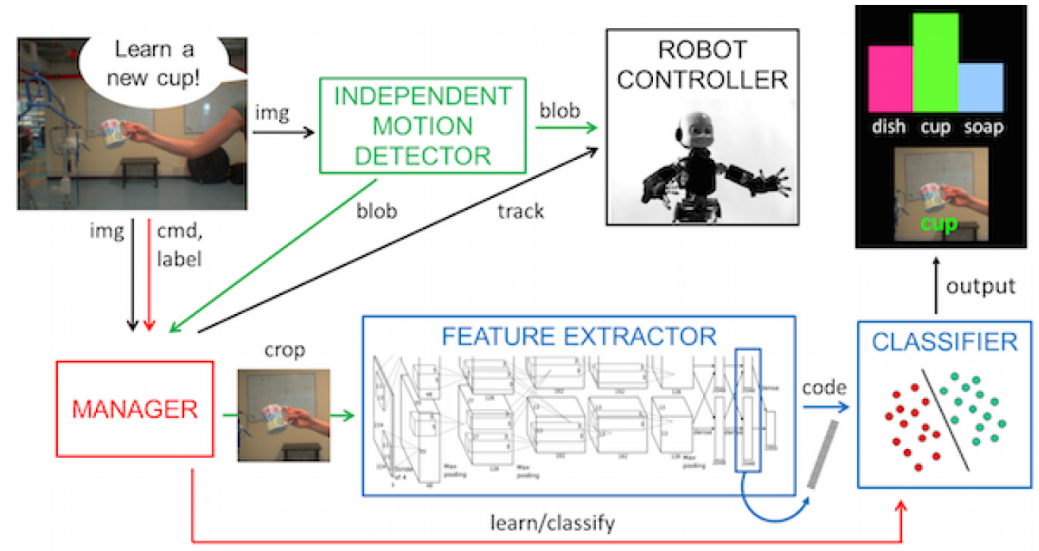


Figure 4.2.: Diagram of the iCub's visual recognition system, from [130]. Details described in the text.

vided by the CNN from the cropped images with the label of that object given by the experimenter. A diagram of this procedure can be observed in Figure 4.2.

On the present Thesis, the methods applied to learn the visual representation of new objects were extended in order to reduce the need of an external teacher. To that end, an autonomous exploration method was implemented, where the only tasks required from the human are to handle the tool to the robot and provide its label.

Once the tool is grasped by the iCub, the exploration process begins. In it, the iCub moves its hand to different poses, so that it can observe the tool from different perspectives. These poses are predefined to utilize the range of iCub's wrist joints to achieve distinct perspectives, but they can be easily modified by the experimenter. On each of the observed perspectives, the tool is roughly segmented from the rest of the field of view, and the resulting cropped image sent to the `onTheFlyRecognition` module, which codes and learns the association between the tool visual appearance and the label provided. In order to obtain the most relevant visual information for latter discrimination, the iCub focuses on the effector, understood as the part of the tool that interacts with the environment.

However, as tools vary in size and geometry, after reaching any of the considered poses for exploration, the tool effector might be partially or totally out of the robot's field of vision. Moreover, at this point the robot has no information about the tool's geometry

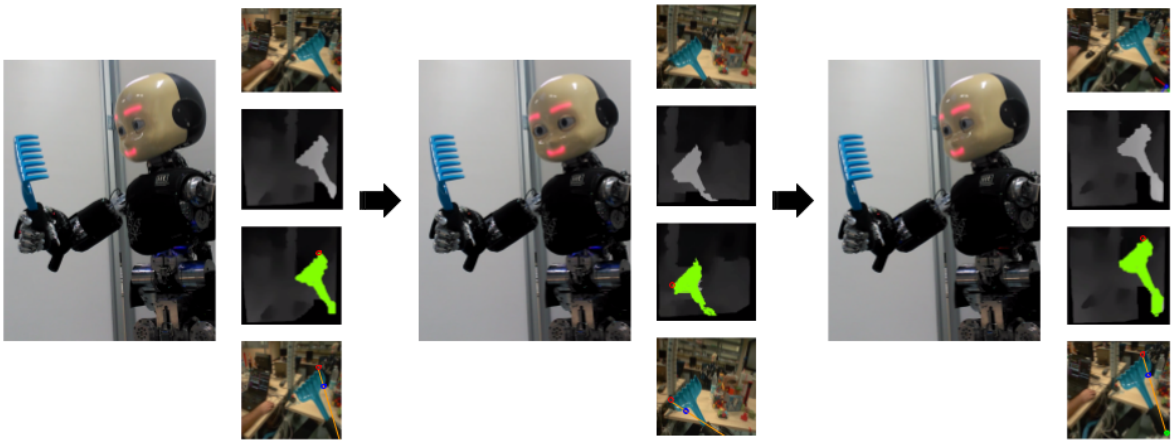


Figure 4.3.: Effector localization procedure. When the tool reaches one of the exploration poses the iCub uses the disparity to segment the tool (in green, third row). Then it locates the point of the blob further away from its hand (small red circle over-imposed on segmentation). Finally, it looks at a point (blue red circle) at 3/4 of the distance between the hand reference frame (green red circle) and the detected 2D tooltip (shown in the last row). This process is repeated until the gaze is stable, or until a certain number of iterations is exceeded.

or pose in order to estimate where the effector might be, so other means have to be applied to locate and gaze at the tool effector. Fortunately, the fact that the tool is in its own hand allows the iCub to have a good estimate of where and in which direction to start looking, since tools are always grasped with a similar radial grasp [1].

Thus, in order to locate the tool, the iCub initially looks just slightly over its hand (10 cm along the X axis and -10 cm along the Y axis of the hand reference frame, that is, along the direction of the extended index and thumb on the palm's plane, respectively). Then, the iCub locates the tooltip through iteratively extracting the tool outline from the disparity map, and looking at the point in the blob further away from the hand reference frame. This process is repeated until the position of the estimated tooltip is stable, or a given number of iterations has been surpassed. An illustration of the described process can be observed in Figure 4.3.

Once the iCub is correctly gazing at the tool effector, a series of images of the tool are obtained by cropping a region around the tool from the whole visual field. This region is determined by bounding box of the closest blob obtained with `dispBlobber`, plus a margin of 10 pixels on each side. Finally, as in the original application, the

[1] Radial grasp refers to a grasp configuration in which the index and middle fingers curl around the tool with the thumb beginning to oppose and press it, and the tool effector extends along the radial side of the hand.

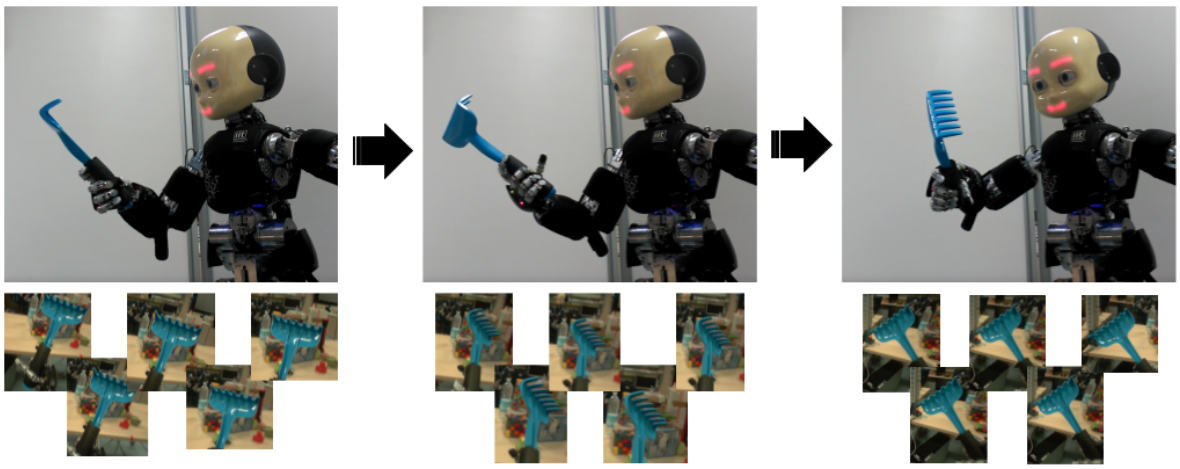


Figure 4.4.: Learning a tool's visual appearance. The iCub moves the tool in its hand to different poses, crops the tool using the methods described in the text, and sends the cropped images to the `onTheFlyRecognition` module, which learns to associate to associate this images with the tool label provided.

cropped images are fed to a CNN whose output feeds in turn a linear classifier which associates them to the user provided tool label. This sequence –tool effector location and subsequent cropping of the tool region to feed the CNN– is repeated for all the exploration poses considered, which provides enough perspectives to recognize the tool in any future pose that it might be grasped in the future. The process and outcome of these steps can be observe in Figure 4.4.

It should be noted that while rendering the process more autonomous, this adaptation from [130] also reduces the number of samples and different perspectives that are captured per object compared to the ones that can be shown by a human experimenter, because of the limited number of poses that the iCub can achieve due to its kinematic restrictions.

It might also be the case, however, that the geometry of the tool is known (a 3D model available from CAD or previous reconstruction, see Section 4.3), but its visual appearance is not. In that case, the same process as described above would be followed to train the system, with the only difference that instead of performing the procedure described to find the tool effector, the tooltip estimation process described below in Section 4.5 would be applied, and the gaze pointed directly at the estimated tooltip.

In either case, after the process of visual appearance learning has been completed by training the classifier with all the desired or available tools, the process of recognition

is simple. After any tool has been handled to the iCub, it observes it in any of the exploratory poses (typically the first one, with the tool oriented vertically with respect to the robot), and uses the same method to crop it from the rest of the image. The cropped image is in turn sent to the trained classifier (CNN + linear classifier), which returns the estimated label of the tool.

4.3. Tool 3D reconstruction

In some scenarios, such as experimental or factory setups, and obviously, in simulations, it can be assumed that 3D CAD or pointcloud models of the tools that the robot might use are available or can be obtained before the robot has to use them for any particular task. Yet, in many occasions, including most real life scenarios, the robot might need to use tools whose 3D representation is not available. In these cases, it should be able to reconstruct the corresponding 3D model of those tools through exploration.

To that end, we implemented an approach that allows the iCub to autonomously generate 3D pointcloud representations of tools, without the need of external intervention by the experimenter (other than eventually handing over the tool to the robot). Basically, it consists of of iterative segmentation, reconstruction and merge of the tool partial views from different perspectives.

Similar techniques have been presented in many different papers in the recent years [72, 202, 140, 136, 207]. However, implementing autonomous full 3D model reconstruction on the iCub platform presents certain specific challenges that need to be considered. In the first place, iCub's stereo vision is achieved through the analysis of the disparity between images from its binocular cameras [42, 129], instead of using structured light or other active methods as in most 3D reconstruction studies. Therefore, the resulting depth estimation is generally noisier than those provided by specific 3D sensors. Also, in order to observe the object to be reconstructed from all angles, most of these studies assume either a fixed camera and an object being moved externally (by the user or on a turning table), which could not be considered autonomous; or a fix scene and a moving camera/robot navigating around it, which is unfeasible on the current iCub setup.

In the present study, thus, we decided to apply a more human-like approach, whereby the iCub starts by holding a tool in its hand, and reconstructs its complete pointcloud representation by obtaining partial view reconstructions from different perspectives and incrementally merging them together. This approach, however, means that neither the

cameras nor the object to be reconstructed stay fixed, hence adding complexity to both the object segmentation and cloud merging tasks. Moreover, aiming at a real life scenario, the whole reconstruction process has to run on-line and on unstructured environments, that is, without markers or otherwise "texturized" surfaces, and independently of the background behind the object to be reconstructed. Finally, the resulting reconstructed model has to be directly applicable for further processing, without the need of human intervention to clean, rotate or otherwise modify it.

While many of the mentioned issues have been tackled separately in the previous literature, we are not aware of any study which considers all of them simultaneously. Yet, in order to overcome the difficulties of such undertaking, we can take advantage of the robot's embodiment and its own involvement in the reconstruction process. Indeed, the process of observing the tool effector in a set of poses for exploration is analogous to the one described in the previous section for learning the tool's visual appearance, which used the hand position to direct the gaze towards the tool. And in fact, both processes can be run simultaneously. In the reconstruction process, however, the steps performed at each exploration pose are the following:

- Segmentation:

  As mentioned above, the process to direct the iCub's gaze towards the tool effector is exactly analogous to the one described in the previous section to learn the tool's visual appearance. However, in this case, after the gaze is properly oriented towards the tool effector, instead of just cropping the image, the tool blob is segmented with `dispBlobber`.

- Reconstruction:

  After the proper segmentation of the visible part (in terms of perspective) of the tool has been achieved, the list of points corresponding to the tool blob returned by `dispBlobber` are sent to the `seg2cloud` module to proceed to their reconstruction. This modules converts the points within the blob into a pointcloud of 3D points in the robot reference frame (as described in Section 3.3).

  A further advantage of the robot embodiment for autonomous tool reconstruction is that irrespective of the perspective of the tool that the iCub is observing, the pose of the tool with respect to the robot's hand is constant. Thus, the hand provides a coherent reference frame for all the partial reconstructions. Therefore, the pointcloud returned by `seg2cloud` is transformed from the robot frame to the

hand's reference frame using the robot's kinematics.

Furthermore, normalization to the hand reference frame facilitates filtering the raw reconstruction provided by `seg2cloud`, as we can safely assume that the tool is connected with the hand, and it does not extend beyond certain boundaries. Therefore, in order to remove any points on the reconstructed pointcloud that might belong to the background or other objects in the scene and not to the tool, the pointcloud is truncated in all three axes of the hand reference frame, removing all the points outside the $(0.0, 0.35)$m range in the X axis, $(-0.3, 0.0)$m range in the Y axis, and $(-0.15, 0.15)$m range in the Z axis. In many cases, the segmentation algorithm is not able to distinguish the tool from the robot's hand, which is therefore also present in the reconstructed pointcloud. As we are only interested in the tool effector (the "superior" half of the tool), the hand is removed by filtering out all the points in the reconstructed pointcloud which are inside a radius of 10 cm from the origin of the hand reference frame. Finally, the pointcloud is smoothed by applying a statistical filter for outlier removal.

- Merging:

  The proposed tool 3D reconstruction method achieves a full representation of the tool by incrementally merging together the partial reconstructions obtained as explained in the previous two points, until the tool has been observed in all the established exploratory poses. Ideally, given that all of them share a coherent reference frame, it would be enough to simply accumulate and downsample the reconstructed pointclouds to end up with a full aligned model. However, there is noise in the kinematic chain estimation, which induces misalignment in among the successive partial reconstructions, and in the depth estimation, which introduces differences in the reconstructed tool surfaces. Therefore, merging is not performed by simple accumulation, but by aligning each new partial reconstruction to the previously reconstructed pointcloud using the Iterative Closest Point algorithm (ICP).

  The roto-translation alignment matrix returned by ICP must represent small values both in translation and rotation, as it should only account for the small misalignment introduced by the sources of noise explained above. Therefore, if the returned translation is bigger than 10 cm in any dimension, or the rotation is bigger than 30° around any axis, the partial reconstruction pointcloud is scaled up or down (alternatively) by 10%, and ICP is run again. If the alignment is not successful

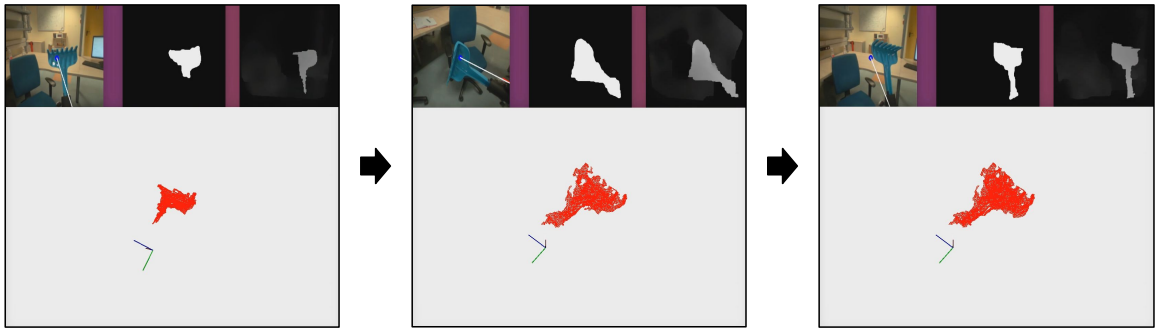


Figure 4.5.: Representation of the reconstruction process at several stages. In order to perform the reconstruction of the tool's geometry autonomously, the iCub iteratively segments it using disparity information, obtains a partial reconstruction, and merges it with the previous one.

after 10 such trials, the pointcloud is rejected and the iCub moves to the next tool exploration pose.

Otherwise, if the alignment step has been successful, the aligned partial reconstruction is combined with the rest of the tool reconstructed so far. Finally, in order to merge overlapping surfaces and reduce noise, the resulting pointcloud is downsampled uniformly using a voxelized grid approach with a leaf side of 2 mm, and the robot moves to the next exploration pose.

After the steps described above have been performed for all the predefined tool exploration poses, the resulting pointcloud is filtered again using a statistical and a neighbors-in-radius filters to further remove outliers and smooth the resulting pointcloud. As a result of the whole process, a pointcloud representation of the tool in the robot's hand has been produced. An example of the outcome of this process at different stages can be observed in Figure 4.5, while Figure 4.6 displays a few examples of reconstructed tool models obtained with the described method, alongside their corresponding original tools.

It is important to remark that because the reconstructions were obtained with respect to the hand reference frame, the resulting reconstructed pointcloud representation reflects the pose with which the real tool is being held by the iCub. In this case, however, the explicit representation of the tool's pose (given by the Pose Matrix P, detailed in Section 4.5) is not available, i.e., the tool's orientation with respect to the hand reference frame is only implicit in its pointcloud representation. In Section 4.4 a method to compute the explicit Pose Matrix from the reconstructed cloud is presented. Moreover, there are cases where the pointcloud model of a tool is already available (from CAD or

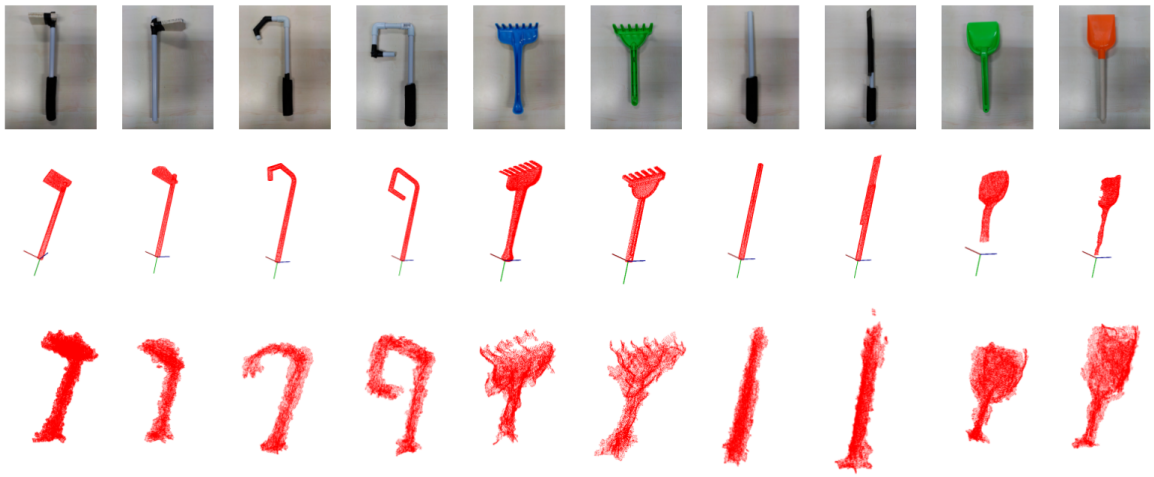


Figure 4.6.: Results of the process of tool reconstruction for a sample of tools. On each example, a picture of the real tool is shown alongside the hand-made CAD model, and the automatically reconstructed model.

a previous reconstruction), but its actual orientation is unknown. For these cases, in Section 4.5 we propose a faster method to determine their pose with respect to the hand reference frame, that avoids having to reconstruct the tool every time.

In order to use consistent terminology in this Chapter and throughout this Thesis, on the remainder, we refer as an **oriented pointcloud model** to the pointcloud representation of a tool that is oriented with respect to the hand reference frame with the same pose as the actual tool being grasped by the robot.

4.4. Tool reference frame and tooltip estimation

The process of tool reconstruction presented in the last section returns an oriented pointcloud model of the tool being held by the robot. However, the information about its orientation is not readily available for the robot, as it is only implicit in the pointcloud representation. In the present section we present a method to make this information explicit, based on the definition and estimation of a reference frame intrinsic to each tool, applicable to the vast majority of man-made tools that could be present in a robotic tool use scenario. This frame of reference, referred to as **tool intrinsic reference frame**, and denoted as $\mathbf{f}$, identifies the effector and handle of the tool, provides its orientation with respect to the hand reference frame, and facilitates the computation of the tooltip's location. The orientation of the tool is useful to determine how to approach an object

with the tool as well as to generalize tool affordances for similar grasps (as we will see in Chapter 6), while the tooltip is required to attach the tool as the robot's new end-effector.

Figure 4.7 illustrates several sets of tools that can be present in a common robotic tool use scenarios, or even in most human ones. If we consider this kind of tools, which we refer to as "radial grasp tools", we can observe that although they vary wildly in shape and function, almost all of them share at least 3 properties:

- The presence of a handle, situated along the longest tool dimension.
- Symmetry on at least one plane which runs along the handle.
- The tool effector, understood as the part of the tool that interacts with the environment, is located at the opposite side of the tool's handle, and its tip lies on the plane of symmetry.

Based on these observations, we can define a reference frame which is coherent for all radial grasp tools, and intrinsic to each tool, that is, dependent only on the tool's geometry. In order to do so, below we identify and define the main planes and axes present in any tool, which characterize the tool's intrinsic frame of reference $\mathbf{f}$. Then, we propose a method to automatically estimate it in the context of tool incorporation, based solely on the tool's pointcloud representation, and the knowledge that it is expressed with respect to the hand reference frame.

4.4.1. Tool reference frame definition

Before presenting the method to estimate the *tool intrinsic reference frame* $\mathbf{f}$, it is required to clarify the meaning of its components, and how they are identified. Although the estimation of $\mathbf{f}$ requires knowledge of the hand reference frame, its definition is general and independent of the tool's pose.

Based on the properties of radial grasp tools stated above, we can define three orthogonal characteristic tool planes, denoted together as a tool's L planes, for any tool of this kind:

- **Symmetry plane** (L_{sym}): It is the plane with respect to which the tool has the maximum symmetry. It runs along the handle and divides the tool into two equal (or almost) longitudinal halves.

Figure 4.7.: Common radial tools that share the qualities described in the text. Above and center: tool datasets applied in other tool affordance studies (taken from [58] and [115], respectively). Bottom left, center and right: common kitchen, workshop and garden tools, respectively.

- **Handle plane** (L_{han}): It is perpendicular to the handle axis, and divides the tool into the effector and the handle sides.
- **Effector plane** (L_{eff}): Orthogonal to the two previous planes, usually divides the "forward" and "back" sides of the tool, forward being the side where the effector is.

Moreover, each plane can also be characterized by a point and a normal vector. If the normalized normal vectors to the L planes are chosen so that they share the same origin, are mutually orthogonal, and one of them can be obtained as the cross-product of the other two, they define a right-handed coordinate system which can be applied as a frame of reference intrinsic to the tool, i.e., $\mathbf{f}$.

However, normal vectors to a plane can be defined in two orientations (on each side of the plane). Therefore, although the direction of $\mathbf{f}$'s axes is determined by the L planes normal vectors, the *orientation* of each axis (the side on which their values become more

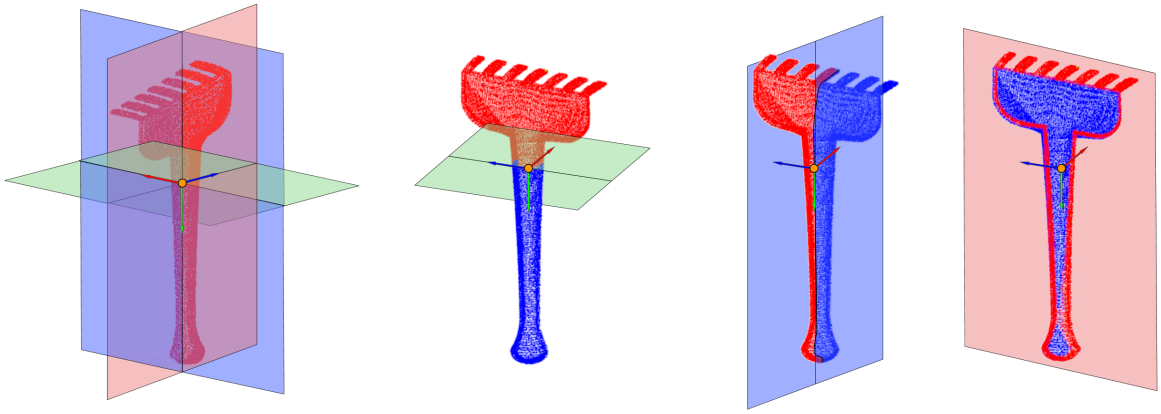


Figure 4.8.: Planes and axes that determine the tool's intrinsic reference frame. Leftmost: Tool model divided by its three characteristic L planes. Center left: Handle plane. Center right: Symmetry plane. Rightmost: Effector plane. The reference frame in all figures shows $\mathbf{f}_{eff}$ in red, $\mathbf{f}_{han}$ in green, and $\mathbf{f}_{sym}$ in blue.

positive) needs to be determined. In order to do so while keeping the orthogonality and right-handedness requirement we define the orientation of the axes in the following way, so that $\mathbf{f}$ is coherent among all tools and it indicates clearly the position and orientation of the different elements of the tool:

- **Effector axis (X)** ($\mathbf{f}_{eff}$): It is positive in the direction of the effector, i.e, towards the "forward" side of the tool.
- **Handle axis (Y)** ($\mathbf{f}_{han}$): Is positive in the direction towards the handle, and negative in the direction towards the effector side of the tool.
- **Symmetry axis (Z)** ($\mathbf{f}_{sym}$): The symmetry basis vector is obtained as the outer product of the other two to ensure orthogonality, so it is positive on the "left" side of the tool, if the effector is looking "forward".

For clarity, Figure 4.8 shows a schematic visualization of the axes and planes defined above, and the divisions that they introduce in an example tool pointcloud.

4.4.2. Tool reference frame estimation

On the previous section we proposed a set of definitions to characterize the elements of the tool intrinsic reference frame $\mathbf{f}$. On the following, we propose a method to automatically estimate the it, relying solely on the pointcloud representation of the tool

expressed with respect to the hand reference frame of the robot.

In the particular application of tool incorporation, we can safely assume that the pointcloud representation of the tool will be expressed in a reference frame which represents the robot's hand, because it is either self-reconstructed by the robot or obtained from an existing CAD model. In the first case, tools are reconstructed with respect to the hand reference frame as described in Section 4.3, so no further modification is needed. In the second case, we can assume that the reference frame can be located at will when constructing or formatting the model. For coherence and convenience, in this study we located the reference frame of the CAD tool models so that it matched the position of the hand reference frame when the tool is being grasped in its canonical position (described in Section 4.5).

Considering the previous information, the method to estimate the intrinsic reference frame of a tool's pointcloud representation W, consists of the following steps:

1. Find the model's main axes: The direction of the tool's intrinsic reference frame $\mathbf{f}$ axes, as well as the location of its origin with respect to the tool itself is estimated by computing the covariance matrix of the pointcloud W, from which its center of mass o, eigenvalues λ and eigenvectors $\mathbf{v}$ are extracted. The first 3 eigenvectors correspond to the pointcloud's main axes, while the eigenvalues correspond to their relative size, and the center of mass to a common reference point for the 3 eigenvectors. Mathematically:

$$C = cov(W), \tag{4.1}$$

$$C\mathbf{v} = \lambda\mathbf{v}, \tag{4.2}$$

$$L[i] \perp \mathbf{v}[i], i \in \{0, 1, 2\}. \tag{4.3}$$

   This set of orthogonal vectors $\mathbf{v}$ provides the direction of the axes of the tool's intrinsic reference frame, whose origin is determined at the center of mass o, but their correspondences to the specific tool axes $\mathbf{f}_{eff}$, $\mathbf{f}_{han}$ and $\mathbf{f}_{sym}$ need to be determined to fully characterize $\mathbf{f}$. The process to identify the planes first, and properly orient the axes later, is detailed on the following steps.

2. Find Handle plane L_{han}: According to the first of the qualities of radial grasp tools stated above, the handle is situated along the longest tool dimension. Thus, the eigenvector with largest eigenvalue indicates the direction of the handle axis,

normal to the Handle plane. That is,

$$\mathbf{v}_{han} = \mathbf{v}[n], \text{ where } n = \underset{i \in \{0,1,2\}}{\arg\max} (\lambda[i]), \tag{4.4}$$

$$\text{accordingly, } L_{han} = L[n] \tag{4.5}$$

3. Find Symmetry plane L_{sym}: The symmetry of the tool's pointcloud representation W with respect to each plane L is computed by measuring the average distance between the points on one side of the plane with the mirrored projection of the points on the other side of the plane. L_{sym} is selected as the plane for which this distance is smaller, that is, the symmetry is larger.

$$L_{sym} = L[m], \text{ where } m = \underset{j \in \{0,1,2\} \neq n}{\arg\max} (sym(L[j]). \tag{4.6}$$

   It should be noted that in order for the result of this step to actually represent the plane running along the handle that separates the tool in two longitudinal halves, the pointcloud representation does not necessarily have to be perfectly symmetrical with respect to L_{sym}, and indeed it rarely is. Even in representations with a high amount of noise, or in tools which are not perfectly symmetrical, as for example some spatulas, L_{sym} still corresponds to the plane that runs along the handle dividing the tool into the two most similar halves.

4. Find Effector plane L_{eff}: The effector plane does not have an intrinsic clear mathematical definition like the previous two. Instead, it is defined and computed in relation to them, as the plane orthogonal to both the Handle and the Symmetry plane:

$$L_{eff} = L[k], \text{ where } k \in 0, 1, 2 \neq n, m \tag{4.7}$$

$$L_{eff} \perp L_{sym} \perp L_{han} \tag{4.8}$$

   After the tool intrinsic planes L have been computed and identified, the orientation of each axis needs to be determined.

5. Find orientation of handle axis $\mathbf{f}_{han}$: As stated above, the orientation of the handle axis $\mathbf{f}_{han}$ is determined such that it is positive in direction towards the handle from the center of the tool. Therefore, the task at this step is to determine which side of the Handle plane L_{han} corresponds to the handle. Contrary to the previous steps, given the large variation in geometry between different handles and effectors, there

is no general assumption on which we can rely for this task that will provide a good solution in at least the large majority of cases.

Nevertheless, when the estimation of the orientation of $\mathbf{f}_{han}$ is part of the process of tool incorporation, we can take advantage of the fact that the pointcloud representation is expressed with respect to the hand reference frame. In this case, under the safe assumption that the robot is holding the tool by the handle while performing its 3D reconstruction, it can be certainly determined that the side of L_{han} where the handle lies is the side that contains the origin of the pointcloud representation, i.e., the hand reference frame. The other side of the tool's pointcloud representation corresponds, by process of elimination, to the effector. Thus, the orientation of $\mathbf{f}_{han}$ is set so that the positive values correspond to the handle side of the Handle plane.

6. Find orientation of effector axis $\mathbf{f}_{eff}$: In order to determine the direction that corresponds with "forward" in a tool, we consider the saliency of features on each side of the effector plane. Specifically, the "forward" side of the pointcloud W is defined as the side where the effector half of the tool (determined in the previous step) contains points further away from the tool's reference frame origin o. Thus, the orientation of the effector axis $\mathbf{f}_{eff}$ (perpendicular to the effector plane) is set such that the positive values take place on this side of the effector plane.

7. Find orientation of symmetry axis $\mathbf{f}_{sym}$: Given that the two main elements of the tool, handle and effector, have been determined in the previous stepsthe orientation of the symmetry axis is defined as the orientation of the cross product between the handle and effector axes basis vectors.

   Given that the orientation of the two other main axes $\mathbf{f}_{eff}$ and $\mathbf{f}_{han}$ has already been determined, and that the goal is to set the orientation of $\mathbf{v}$ so that it corresponds to a right-handed coordinate system that can be used as the tool's reference frame $\mathbf{f}$, the orientation of $\mathbf{f}_{sym}$ is chosen so that the set of axes defined by $\mathbf{v}$ corresponds to a right-handed coordinate system. Thus, it is computed as the cross product between the handle and effector axes basis vectors:

$$\mathbf{f}_{sym} = \mathbf{f}_{han} \times \mathbf{f}_{eff} \tag{4.9}$$

The tool intrinsic reference frame $\mathbf{f}$ is actually expressed on the same frame of reference that the pointcloud reconstruction from which it is estimated, that is, the hand reference frame. Thus, the equations of the frame's axes represent explicitly the orientation of

the tool in any of its three axis. Generally, the orientation of the handle axis represents how tilted the tool is with respect to the hand, while the orientation of the effector axis represents, naturally, where the effector of the tool is pointing with respect to the direction of the hand's palm.

One of the strengths of this approach to estimate the tool's frame of reference **f** is that it relies on very few assumptions to be met in order to work successfully. Indeed, the main assumptions that the method makes are that the tool's handle axis is longer that any other axis, and that the tool has a certain degree of symmetry along a plane that contains that axis. In some situations this premise might not be met, such as when analyzing very wide tools like a wide rakes, or very asymmetrical tools, such as a branch. When analyzing these kinds of tools, the estimation of the Handle plane, and in turn, of the Effector plane, will likely be incorrect, but they correspond to fringe cases, not commonly found among radial grasp tools usually found in robotic scenarios.

Moreover, the method is also very robust to noise in the 3D representation of the tool, since all the computations required throughout the process of determining **f** have a high tolerance to noise. Indeed, as most of the decisions are made in terms of comparison (symmetry between two sides of a plane, longest axis, furthest away point), if noise affects the whole pointcloud similarly, it would not modify their outcome. Outlier points would indeed have more influence in this outcome, but none should be left after the thorough filtering process applied during 3D reconstruction.

4.4.3. Tooltip estimation

As stated above, one of the main applications enabled by the knowledge of the tool reference frame **f** is the precise estimation of the tooltip, required to perform the extension of the robot's kinematic chain to the new end-effector provided by the tool.

To that end, in the first place we need to define what the tooltip is, with respect to the tool's geometry only. Our proposed definition is the following:

Tooltip: Location on the tool represented by the point on the Symmetry plane of the tool, above the Handle plane (i.e. on the effector side), furthest away from the Effector plane, on the positive side of the effector axis.

The main advantage of this definition over other possible ones that define the tooltip with respect to the hand reference frame, is that the present is independent of the tool

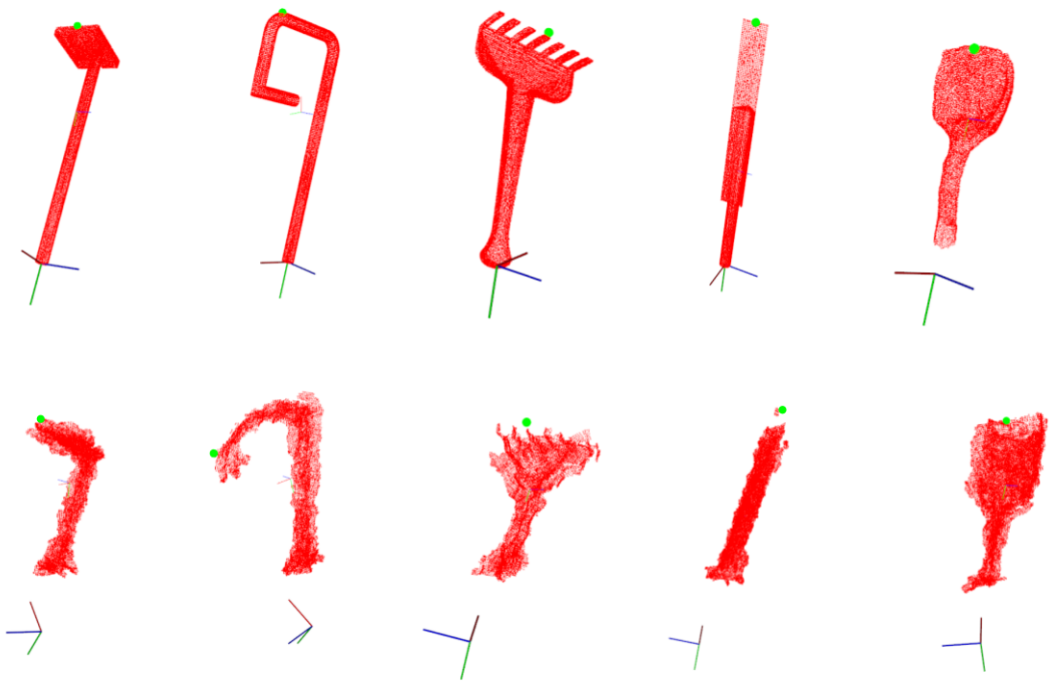


Figure 4.9.: Results of the tooltip estimation process described in the text shown for a few example tool pointclouds, from CAD models (top row), or reconstructed (bottom row).

orientation. Thus, even if the tool is rotated with respect to the robot's hand, the result will be coherent.

Indeed, the application of this definition to many different tool pointcloud representations, obtained either from CAD or through the autonomous reconstruction procedure described in 4.3, returns in the large majority of the cases a location for the tooltip that coincides to what most people would consider to be the tooltip of those tools. This can be observed in Figure 4.9, where the tooltip location is displayed for a large sample of these tools.

Moreover, as it is directly based on information provided by the tool's reference frame $\mathbf{f}$, the accuracy of the tooltip estimation also validates the tool frame estimation process by means of the steps detailed above.

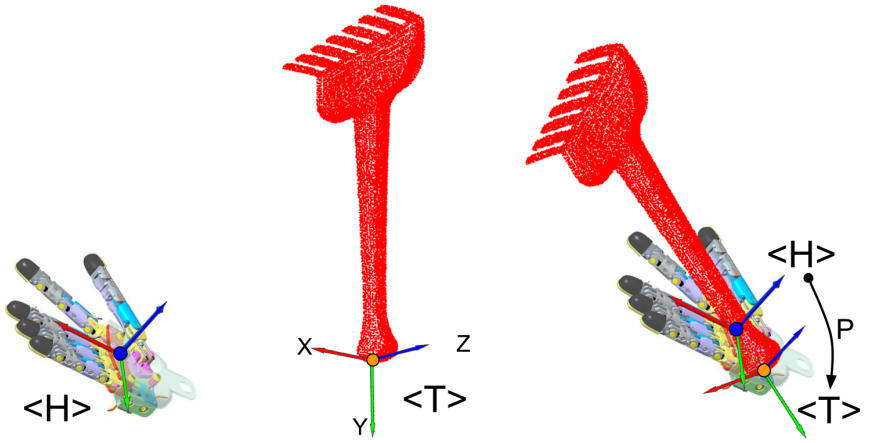


Figure 4.10.: Graphical representation of the definition of the Pose Matrix P. The tool's reference frame ($< T >$, orange origin, center) corresponds to **b**, described in the text. The hand reference frame ($< H >$, blue origin, left) is defined by the robot kinematics. The Pose Matrix P is thus defined as the roto-translation matrix required to align $< H >$ with $< T >$, so that $< T >= P < H >$.

4.5. Tool pose estimation

As stated in this Chapter's introduction, we refer to tool incorporation as the process whereby the iCub obtains a representation of the geometry, pose and reach of a tool which it is grasping. The 3D reconstruction method described in Section 4.3 provides an explicit representation of the geometry, and implicitly, as noted, also the pose of the tool. On Section 4.4, we described how to make the implicit pose explicit, by determining the tool's reference frame **f**.

However, it is a time consuming approach that is not necessary if a 3D pointcloud model of the tool is already available, either from a CAD model or from a previous reconstruction. For these cases, in this section we introduce a fast and reliable method for pose estimation, based on the alignment of the available model with a single partial view reconstruction to the tool in the hand. However, before dwelling into the details of the method to estimate the tool pose, we need to formally define it.

4.5.1. Pose Matrix

Qualitatively, the *tool pose* represents the way in which the tool is being grasped with respect to the hand's reference frame. Numerically, we can express the tool pose in terms of the 4×4 roto-translation **Pose Matrix** P required to transform the hand reference

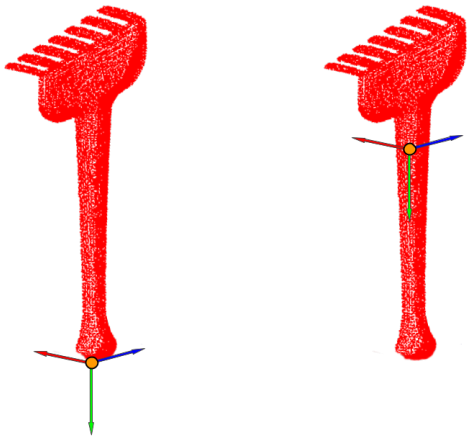


Figure 4.11.: Example of the representation of the tool base reference frame **b** (left) and the tool intrinsic reference frame **f** (right) with respect to a pointcloud representation.

frame $< H >$ frame to any reference frame intrinsic to the tool $< T >$, that is,

$$< T >= P < H > \tag{4.10}$$

The hand reference frame $< H >$ is defined by the robot kinematics. The tool reference frame $< T >$ applied can be arbitrarily chosen, as long as it is coherent among all the tools that can be considered, as the Pose is expressed in relative terms. For clarity, Figure 4.10 displays a graphical description of the elements relevant to the definition of P.

Accordingly, we define a *canonical pose* as the pose represented by the identity matrix, $P^{can} = I$; When the tool is oriented in the canonical pose, the hand reference frame and the tool reference frame are aligned:

$$if\ P = P^{can} = I, \tag{4.11}$$

$$< T >=< H > \tag{4.12}$$

In order to simplify the definition of P, and make its computation more consistent among tools, we define a new reference frame intrinsic to the tool which we refer to as **tool base reference frame** and denote as **b**. The reference frame **b** has the same orientation as **f**, but its origin is not situated on the center of mass of the tool. Instead, it is located on the tool's base, defined as the location of the most positive point along the handle axis (Y). Figure 4.11 illustrates the relationship between the two frames.

The decision to define and apply a new reference frame rather than just using **f** was taken for two main reasons. On the one hand, in order to set the reference frame on CAD models, the center of mass is difficult to estimate, while locating the center of the basis is quite straight-forward. At the same time, translating **f** to the tool's base is also easy when using reconstructed models. Therefore, the definition of **b** made much easier to have a coherent reference frame that is valid for all pointclouds, independently of whether they were obtained from CAD models or reconstructed.

Thus, we assume and enforce that all the saved tool pointcloud representations are expressed with respect to their base reference frame **b**. In the case of CAD models, this is simply done by orienting them correspondingly before transforming them in to pointclouds. In the case of reconstructed pointcloud models, the tool intrinsic reference frame **f** is estimated applying the method described in Section 4.4. Then, **b** is computed by translating **f**, without modifying the orientation, to the coordinates of the most positive point along the handle axis. Finally, the pointcloud is transformed so that it is expressed with respect to **b**. This last step is actually equivalent, assuming that the reconstructed pointcloud orientation corresponded to a grasp on pose P, to perform the inverse transformation from the tool reference frame to the hand reference frame $P^{-1} < H >=< T >$. This step, as it transforms the oriented pointcloud model to its canonical pose, is referred to as *canonization*.

By knowing the geometry and the pose of a tool, represented by the pointcloud representation and the Pose Matrix P, respectively, the robot has precise information of how a tool is being grasped. The model of the tool oriented with respect to the hand reference frame in the same ways as the robot is actually grasping the tool, that is, oriented according to P, corresponds hence to the *oriented pointcloud model*.

4.5.2. Pose estimation by means of alignment

As stated above, the Pose Matrix P is defined as the transformation required to align $< H >$ with $< T >$, i.e. $< T >= P < H >$. Given that when a tool grasped in its canonical position P^{can}, then $< T >=< H >$, if we refer to $< T >$ when $P = P^{can}$ as $< T >^{can}$, we can clearly see that if

$$< T >= P < H > \tag{4.13}$$

when $P = P^{can}$,

$$< T >= P < T >^{can} \tag{4.14}$$

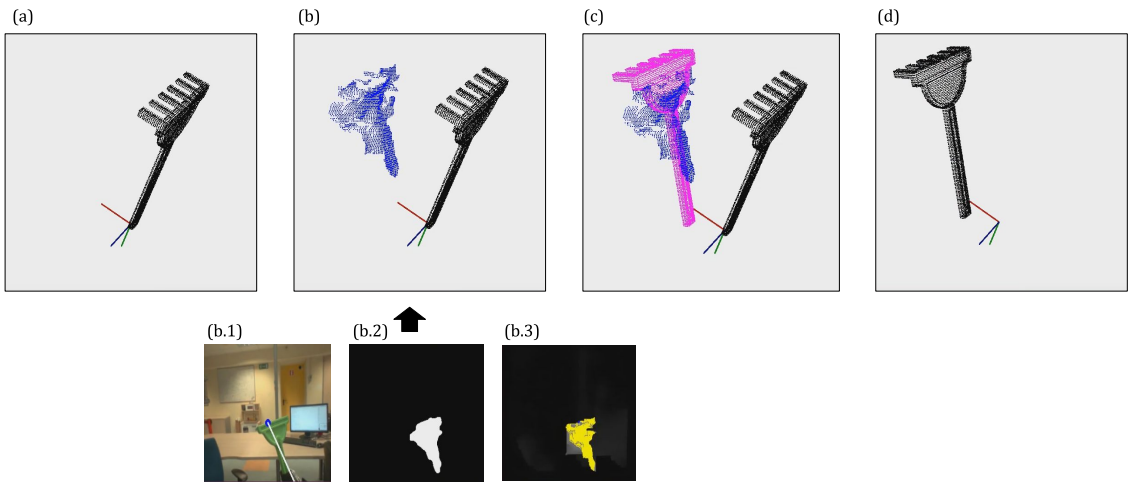


Figure 4.12.: Example of the tool pose estimation through alignment process. (a) Load 3D pointcloud model on canonical pose. (b) Extract partial reconstruction using `seg2cloud` model (segmentation + depth estimation). (c) Find Pose Matrix P by aligning 3D model to partial reconstruction. (d) Obtain oriented pointcloud model by applying P to the 3D model.

In other words, this means that P can also be understood as the required transformation to align a tool 3D model from its canonical pose to its actual pose, given by the oriented pointcloud model.

In development of the present Thesis, we applied the previous corollary in order to implement a method for fast and reliable pose estimation (for the cases where a complete pointcloud model of the tool is already available). For that, we assume (and enforce, as we stated above), that the available pointcloud model is expressed with respect to the base reference frame **b**, that is, in its canonical pose.

If the previous conditions are met, the method for automatic pose estimation consists of the following steps, displayed graphically on Figure 4.12:

1. Load the complete pointcloud model from memory: Applying the methods described in Section 4.2 the tool in hand is recognized and its corresponding full pointcloud model loaded.

2. Extract a partial reconstruction of the observed part of the tool: Once the pointcloud model of the tool in hand has been loaded in its canonical pose, iCub performs the methods described in Sections 4.2 and 4.3 in order to fix the gaze on the tool effector and extract a partial reconstruction of the observed part of the tool. As described in the referred sections, the partial view reconstruction is oriented with

respect to the hand reference frame in the way that the actual tool being held is.

3. Estimate the alignment matrix between the full pointcloud model and the partial reconstruction: ICP algorithm is applied in order to align the pointcloud model loaded from memory to the partial reconstruction just obtained.

4. Check grasp feasibility of the alignment matrix: similar to the case of the merging step of the tool reconstruction process, which was also based on ICP to align partial views together, there are cases where the alignment estimated by ICP does not correspond to a feasible grasp, and thus should be rejected. Specifically, the alignment is rejected if:

   - Translation on any dimension is higher than 10 cm, as it would represent that the base of the tool is further than 10 cm from the hand reference frame, ergo, not in the hand.
   - Rotation on the Z axis $\mathbf{f_{sym}}$ (how tilted the tool is) is outside of the $(0, 90)$ degree range, since it would mean that the tool is pointing out of the fingers or behind the thumb.
   - Rotation on the X axis $\mathbf{f_{eff}}$ is outside the $(-45, 45)$ degree range, as it would mean that the tool is too inclined sideways to actually represent any feasible grasp.

   If the alignment matrix satisfies any of the conditions above, the alignment it is rejected, and the pose estimation process restarted from step 2. If the alignment is unsuccessful more than a given number of times (set to 10 by default), the process is stopped and the iCub announces that it has not been able to estimate the pose. Otherwise, the method proceeds to the next step.

5. If the alignment estimated by ICP corresponds to a feasible grasp, then the returned alignment matrix is assigned to P, and applied to transform the canonical pointcloud model in order to obtain the oriented pointcloud model.

In cases such as the one displayed, where alignment is successful, at the end of the pose estimation process the iCub has explicit information about the precise geometry and pose of the tool in its hand, as well as the corresponding oriented pointcloud model. Therefore, it can apply the method described in Section 4.4.3 to determine the position of the tooltip with respect to the robot's hand reference frame, and hence extend the kinematics of the robot to incorporate the tip of the tool as the new end-effector for

further action execution. Moreover, as in the situation after reconstruction, the oriented pointcloud model can also be used to estimate the affordances of the tool in hand, as described in Chapter 6.

5. Functional Features

5.1. Introduction

Once the robot has incorporated the tool by means of the methods presented in the previous chapter, its reach and pose are known to the robot and hence it is ready to use the tool for interacting with its environment. However, in order to learn the regularities of this interaction that depend on the tool, i.e., the tool affordances, it should be represented in terms of certain descriptors that capture its intrinsic properties relevant to such interaction.

The present work adheres to the notion described in Section 2.3 that affordance detection occurs by means of the perception and analysis of the *functional features* of objects or entities, preceded by a "phase of learning their relationship with the environment and the way in with they are employed", paraphrasing Bogoni [17]. Functional features, thus, represent the properties of objects which correlate with the way in which they behave during interaction, while the learning process should find these correlations and use them to predict the consequences of such interaction.

As stated above, tools are a particular type of object in that they represent objects that an agent can act *with*, rather than act *upon*. In other words, tools are embodied and incorporated by the robot, while other objects remain external to the robot during action execution. This contrast should also be reflected in the way objects are represented when being used as tools, in contrast to when they are not. In fact, there is neuroscientific evidence that different representations of tool and non-tool objects also occur in primate brains [101, 7, 11].

According with this perspective, in this chapter we present two sets of *robot-centric tool functional features*, based on 2D and 3D geometry descriptors (Section 5.2 and 5.3, respectively). The proposed features are robot-centric insofar as they are extracted from tool representations which are normalized with respect to the robot's hand reference

frame. Moreover, they are tool specific because they are devised to describe grasped objects (tools) and encode their pose and orientation. Admittedly, other properties than geometry, such as stiffness, hardness, etc, could influence the behavior of the tool, but geometry is in all cases relevant, and, given the state-of-the-art in computer vision, easier than other sensor modalities to extract and analyse automatically on a robotic setup.

Thus, during the development of this Thesis, we focused on tasks for which the tool's function depended solely on its geometry, or in other words, in tools whose functionality depended only on their geometry. In these cases, it applies that different tools can afford similar functionality if (but not only if) they share some common geometrical features. This is indeed an essential condition to enable the generalization of learned affordances among similar tools based on their functional features. Moreover, the effect that can be achieved with a tool depends as much on the action performed as on the way in which it is grasped. Therefore, we need a way to represent tools which takes into account both aspects, geometry and pose. This approach is also in line with recent neuroscientific studies which suggest that context and pose influence the way in which tools are perceived in the brain, because of the different functionality a tool might offer depending on the way it is grasped [119].

For convenience, we use the term *tool-pose* to specify a particular tool in a given pose, following the nomenclature in [19]. Not to be confused with the tool pose, without the hyphen. A tool-pose is a specific tool in a specific pose, while the pose of a tool, which we denote as P, determines its position and orientation with respect to the hand reference frame. In other words, even if `tool A` and `tool B` share the same pose, they will correspond to 2 different tool-poses. Similarly, the same tool in 2 different poses $P1$ and $P2$ also corresponds to 2 different tool-poses.

5.2. 2D Shape Features

Our first approach towards the characterization of tool functional features for affordance learning was based on 2D geometrical features of the grasped tool. These were extracted from the observed contour from the robot's perspective when looking at the tool, normalized with respect to the robot's hand. An example of the image processing at different stages of the feature extraction pipeline can be observed in Figure 5.1.

In order to begin the feature extraction process, the first step is to determine the

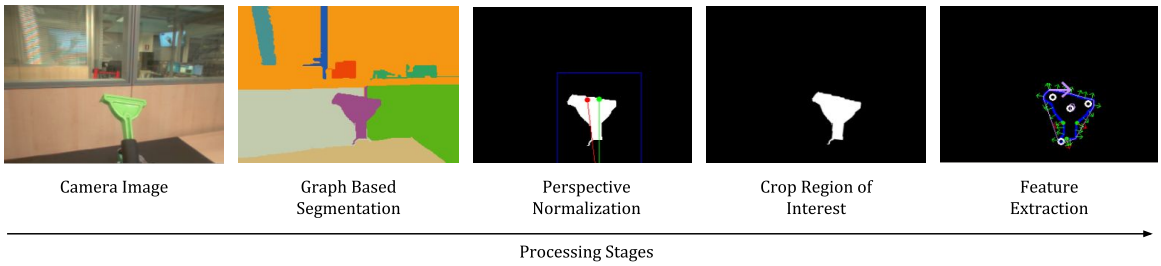


Figure 5.1.: Visual feature extraction processing pipeline. After the robot grasped the tool, it looks at by applying the methods described in Section 4.2. Then, a graph-based segmentation algorithm is used to obtain the clean contour of the tool, whose orientation is normalized w.r.t the hand reference frame, and cropped to retain a region of interest around the tool effector. Finally, the set of 75 geometrical features described in the text is extracted from the contour of the resulting blob.

location of the tool-pose effector in order to look at it, and the tooltip to select the appropriate region for segmentation. To this end, the methods presented in Chapter 4 can be applied if stereo-vision or other methods for depth estimation are available on the robot. However, given that this set of features is based on 2D information only, it can also be implemented in robots without 3D vision systems. In such cases, the method developed in Tikhanoff et al. [174] can be applied to automatically estimate the tooltip position. This method involves a discovery procedure in which the robot swings the grasped tool in different positions, while the generated motion is detected by a variant of the Lucas-Kanade optical flow algorithm extended to compensate for ego-motion [27]. An optimization technique analyzes the motion data to reliably establish the tooltip position based on the minimum error between the predicted position of the tooltip and the observed one. In either case, when the position of the tooltip with respect to the hand is found, it is annexed to the arm's forward kinematics so it can be used as an end-effector and retrieved at any moment.

Once the robot has the tool in its visual field, its precise contour is extracted by means of a graph-based segmentation (GBS) algorithm that splits the images from the robot's left camera into uniform regions [43]. The 2D projection of the tooltip in the image plane is used as the seed for the segmentation to extract the region that corresponds to the tool, i.e., the tool blob.

After the region representing the view of the tool has been isolated from the rest of the image, its orientation is normalized with respect to the hand reference frame. This is accomplished by computing the angle between the tooltip and the hand reference frame (computed from the robot's kinematics), and rotating the tool blob the same amount, but on the opposite direction. This way, the normalized tooltip projection is always

vertical with respect to the hand reference frame.

As it happens with the segmentation methods described in Chapter 4, sometimes the 2D GBS method also includes the hand within the tool region. Thus, in order to represent the tool only in function of its end effector, which is the part that differentiates among tools and poses, a region of interest between the tooltip and about 3/4 of the distance between the tooltip and the hand center is selected, and cropped out from the rest of the image.

Finally, after the segmentation and normalization processes, the following set of geometrical features is extracted from the contour of the resulting blob. References on the list cite previous works that have used them, if any. The number in parenthesis corresponds to the number of values used to represent that feature:

- Based on convex hull:
  - Area of the convex hull (1): Smallest convex area that contains all points in the contour.
  - Solidity (1): Contour area / convex hull area.
  - Depth of the 5 larger convexity defects (5): The 5 biggest areas inside the convex hull, but outside the object's contour, i.e. concave inner corners.
  - Histogram of bisector angles at convexity defects (8): Represents the frequency of "inner corners" in function of their bisecting direction.
- Based on thinning/skeletonization (from [206]):
  - Number of skeleton bifurcations to the left, right, under and above the segmented region's center of mass, respectively (4).
  - Number of skeleton endings to the left, right, under and above the segmented region's center of mass, respectively (4).
- Moments:
  - Normalized central moments *nu11*, *nu02* and *nu02* (3).
- Shape descriptors (inspired by [203] and [204]):
  - Contour Area (1).
  - Contour Perimeter (1).
  - Compactness: ratio of area to perimeter (1).
  - Length: major principal axis (1).

  - Width: minor principal axis (1).
  - Aspect ratio: bounding rectangle width/height (1).
  - Extension to the right w.r.t. the center of mass (1).
  - Extension to the left w.r.t. the center of mass (1).
  - Elongation: length/width (1).
  - Rectangularity: area/(length×width) (1).

- From the angle signature, i.e., the outwards pointing normal angle between each two points in the contour (based on [204]):
  - Bending energy: sum of squares of the angle variation along the contour, divided by the number of points in the contour (1).
  - Angle signature histogram (8).

- Domain transformations from the distance to the centroid signature (from [204]):
  - Fourier coefficients (15).
  - Wavelet coefficients (15).

It should be noted that the due to the wide array of different features that the described set includes, their values comprehend several orders of magnitude. For example, some features such as solidity have a maximum value of 1, while others like area or bending energy can take values of 10^6 or larger. In order to prevent numerical problems on further processing steps that this disparity could cause, the feature values should be normalized so they all span a similar range.

On the experiments conducted as part of the present Thesis where this feature set was applied (described in Section 6.2), the mean and variance of each feature along the training subset of the data were computed. On any incoming test feature vector, each extracted feature was normalized by subtracting the corresponding training data mean and dividing by the training data variance.

The suggested feature vector comprises a large set of contour shape descriptors, larger than any other approach in the literature, to the best of our knowledge. The rationale behind this approach is in line with ideas of Eleanor Gibson, who suggested that agents learned to perceive affordances by being increasingly able to discriminate the information relevant to particular tasks out of the plentiful mass of stimulus properties that emanate from a set of objects [50]. A large set of shape features that describes the tool shape in many different ways is more likely to include relevant features for any task at hand.

However, in principle we do not know which features are relevant for each particular task at hand, or in fact, we should not make use of this knowledge in a developmental approach, as it is for the robot to discover them. In practice, the relevance of each feature for each particular task will be implicitly pondered in the training process of the machine learning algorithm chosen to predict affordances from the given set of features.

5.3. 3D features: Oriented Multi-Scale Extended Gaussian Image

In the context of tool affordance learning, the way in which tools are represented determines the generalization capabilities of the proposed method. For example, a method which represents objects solely by given labels (as in the early work by Fitzpatrick [48] and Stoytchev [168]), will not be able to generalize the learned affordances to any object outside the initial training set. As we saw, some authors tackled this problem by describing objects and tools by means of functional features. However, to the best of our knowledge, all studies on *tool* affordance learning that involved interaction with the environment apply only 2D information [75, 58, 97].

And yet, real world objects functionality depends on, and in many cases emerges directly from, their 3D geometry. Therefore we argue that moving from 2D to 3D features to describe tools in affordance studies is a desirable step. On the one hand, doing so will spare us many of the most common drawbacks of 2D image analysis, such as the object representation dependence on perspective, or occlusion induced errors. On the other, it can also provide much richer information about the actual functionality of tools.

Within the fields of robotics and computer vision, 3D features are mainly applied for object retrieval or recognition/classification [205, 172, 5, 138, 201]. Accordingly, they are usually designed to be similar for similar objects, and also, as opposite of desired in interaction scenarios, independent of the object's pose. This is also true in the few published computer vision approaches for tool affordance learning that apply 3D information [1, 208].

Therefore, on [96] we introduced the **Oriented Multi-Scale Extended Gaussian Image** (OMS-EGI), a holistic descriptor devised to represent grasped radial tools in interaction scenarios. OMS-EGI encapsulates in a compact way the geometrical properties

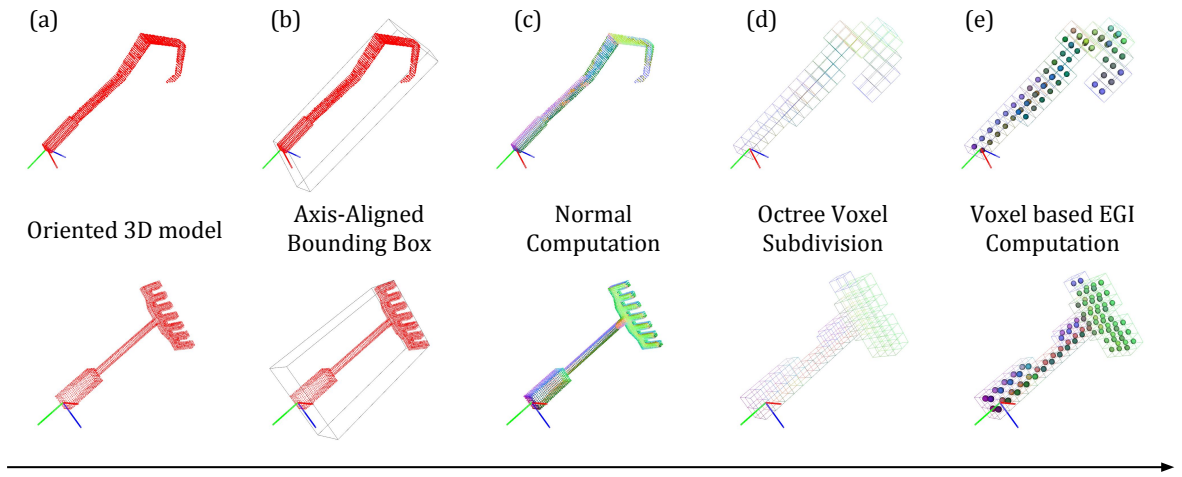


Figure 5.2.: OMS-EGI Computation Steps: First, the 3D pointcloud model of the tool grasped by the robot is loaded from memory and rotated to obtain the oriented pointcloud model (a). Then, the oriented pointcloud model's Axis-Aligned Bounding Box (AABB) is computed w.r.t. the hand reference frame axes (represented in the figure by the axes at the base of the tool pointclouds) (b). At the same time, the orientation of the surface normals is computed for the whole pointcloud (c). On the next step, the volume enclosed by the AABB is iteratively divided into octants D times, generating voxels of different resolution levels l (d). Then, a histogram of the normal orientations of the surface enclosed in each voxel is computed (e). Finally, all histograms are concatenated in order to build the OMS-EGI feature vector. For visualization purposes, only one resolution scale is displayed. Normal values and normal histograms are represented with colors by mapping XYZ angular values to RGB color values, and XYZ histograms to RGB histograms and averaging over color space.

of a tool on a particular grasp configuration as a whole, simultaneously encoding information about its spatial occupation with respect to the hand, and its surface geometry. In a nutshell, the OMS-EGI representation of any tool-pose is obtained by computing voxel-wise normal histograms from iterative octree divisions of its Axis Aligned Bounding Box (AABB). The pseudo code of the process is reported in Algorithm 1, while a visual representation and description of the steps taken is displayed in Figure 5.2.

The OMS-EGI descriptor is a modification and extension of the Extended Gaussian Image (EGI), proposed by Horn in 1984 [69]. The original EGI represents any 3D model in polygon mesh format as a histogram on the spherical space of the normal orientations of the model, weighted by the area of the faces with such normals. The most important variation in the OMS-EGI with respect to the original EGI formulation is that instead of taking a single normal sphere histogram representing the whole model, the OMS-EGI consists of a concatenation of voxel-based EGIs computed at different resolution scales. We refer by *voxel-based EGI* as the normalized normal histograms computed from the

Algorithm 1 OMS-EGI Feature Extraction

omsegi = **compute_omsegi(model_or,** D**,** N**)**, *where*
    model_or = oriented pointcloud model, D = Depth, N = #bins per histogram.
***initialization*:**
    AABB ← *findAABB(model_or)*
    normals ← *compute_normals(model_or)* ▷ Find the normals of the whole model.
    oms-egi ← *[]*
***loop*:**
for *level* = *0* → *D* **do** ▷ Compute sub-EGIs at different scales of granularity
    voxel_list ← *compute_voxel_grid(AABB, level)*
    for all $v \in$ *voxel_list* **do** ▷ Loop through all voxels in Bounding Box
        if *is_empty(v)* **then**
            oms-egi ← *concatenate(oms-egi, zeros(1,*N^3*))*
        else
            norm_hist ← *comp_norm_hist(normals, v, N)* ▷ Compute voxel-based EGI.
            oms-egi ← *concatenate(oms-egi, norm_hist)*
        end if
    end for ▷ End voxel loop
end for ▷ End depth level loop

portion of the model enclosed in a particular voxel, whereas *scale* corresponds to the size and number of voxels from which the voxel-based EGIs are computed.

The other determining aspect of OMS-EGIs is that voxels are computed from octree subdivisions of the pointcloud's axis-aligned bounding box (AABB) with respect to the robot's hand reference frame. Considering that OMS-EGI is computed from a representation of the tool in its actual grasping pose, that is, its oriented pointcloud model, as defined in 4.3, this characteristic ensures that the information on the OMS-EGI is relative to the current tool-pose. Therefore, the same tool in different poses will produce different OMS-EGI vectors. This fact also helps bypassing the necessity of an initial pose estimation, present in all other non-pose invariant 3D features. Nevertheless, if an actual canonical or preferred pose exists for the model, its OMS-EGI can naturally be computed in such pose, which would enable further analysis in absence of any particular grasp.

Moreover, as discussed before, the relevant part of a tool in order to characterize it and determine its functionality is the effector, which we consider as the half along the handle axis further away from the hand when the tool is grasped (Section 4.4).

Therefore, rather than computing the OMS-EGI from the complete AABB, we obtain the voxel-based EGIs only from those voxels contained in the effector half of the AABB. An additional advantage is that the length of the resulting OMS-EGI vector is halved.

When dealing with pointclouds instead of surface meshes, as is the case in the present approach, there are slight approximations to be made with respect to the original EGI formulation. As "faces" do not exists on pointcloud representations, normals can not be computed from them, but rather estimated from the surrounding point neighborhood support of the point (also called k-neighborhood) [147]. Therefore, the voxel-based EGIs can not be weighted by the area of the faces. Instead, under the safe assumption that points in the pointcloud are reasonably uniformly distributed along the model's surface, we normalized each voxel-based EGI by the number of points (each corresponding to one normal) enclosed in the voxel, in such a way that the sum of all the values of each voxel-based EGI histogram is 1.

Furthermore, the information about the tool's pose and geometry is conveyed in terms of surface information and spatial occupation, the former encoded in the normal histograms and the latter represented by which voxels contain histogram data and which are empty. These two sources of information are weighted in function of the values of the following parameters, which also determine the size of the OMS-EGI feature vector:

- N: The number of bins into which the possible values of the angular directions of the surface normal in each dimension X, Y, Z are divided to form the voxel-wise histograms. It reflects the accuracy with which each voxel-based EGI will represent the normals contained in its corresponding voxel. Thus, each voxel-based EGI is represented by N^3 values.
- D: Depth represents how many times the AABB is divided into octants to form voxels. At each resolution level $l = 0, \ldots, D$, the number of voxels resulting from the division of the AABB is 8^l (thus the name "octant"), and hence, $\frac{1}{2}8^l$ belong to the effector half. N represents thus the resolution at which the voxel-based EGIs will be computed, by controlling the number and size of these voxels.

Therefore, the total size S of the OMS-EGI vector is given by the number of voxels in the upper half of the axis aligned bounding box times the size of the histogram on each voxel, plus 1 because at level 0 only one voxel is considered (8^0), which corresponds to the whole aligned bounding box of the tool-pose and thus can not be halved:

$$S = (1 + \frac{1}{2}\sum_{l=1}^{D}(8^l)) \cdot N^3, \text{ where } l\text{: octree level.} \tag{5.1}$$

Therefore, by setting different values to this pair of parameters, the OMS-EGI descriptor can be applied to represent tool-poses in terms of their spatial information (high D and low N), surface information (low D and high N), or a balanced combination of both (similar values to D and N). On the development of this Thesis, the comparison of which kind of information was better suited to represent affordances was conducted by evaluating the following 3 parameter settings:

- **Balanced information (BALAN)**: Setting $N = 2$ and $D = 2$, the feature vector corresponds to a *balanced* OMS-EGI, as applied in [96], where both spatial and surface information are represented. In this case, the length of the resulting feature vector is $S_{BALAN} = 296$.

- **Spatial information (OCCUP)**: If $N = 1$, all normals in each voxel are assigned to the same bin irrespective of their orientation, and therefore each voxel-wise EGI can be subsumed to a single value. On voxels where any point of the oriented pointcloud model is present, this value is 1, while on empty voxels the value is 0. Therefore, setting $N = 1$ transforms the OMS-EGI into a axis aligned binary occupancy grid. In the present study, D is set to 3 so that the total length of the feature vector is similar to the BALAN setting: $S_{OCCUP} = 293$.

- **Surface information (EGI)**: When $D = 0$, the only voxel considered is actually the tool-pose aligned bounding box, without further subdivisions. In this case, which is equivalent to the original formulation of the EGI descriptor [69], the OMS-EGI represents a normal histogram of the tool-pose, provided a certain histogram resolution function of N. In this case, N is set to 6 so that the length of the feature vector is $S_{EGI} = 216$ in a similar range of the other settings.

6. Learning Tool Affordances from Interaction

6.1. Introduction

In previous Chapters we have introduced methods that enable the iCub to autonomously incorporate tools (Chapter 4) and to represent them in terms of robot-centric functional features (Chapter 5). In the current Chapter, we will present a series of learning architectures which, making use of the previously described techniques, allow the iCub to learn tool affordances from interaction.

Together, these architectures represent an incremental development towards more capable methods for tool affordance prediction, in terms of both accuracy and generalization. While some general elements are shared among all these architectures, the focus of each approach varies, as well as the machine learning techniques applied and the way in which the elements of an affordance are characterized and connected. At the same time, the proposed experiments serve not only to evaluate these learning architectures, but also to assess the suitability and potential of the functional features presented in Chapter 5.

In particular, on our first approach, published in [97], affordances are categorized through K-means clustering, and then tool-poses classified in function of the affordance they had generated. Thus, it emphasises the discovery of affordance types, while tool(-poses) are only grouped in terms of the effects they are able to elicit. In this approach, tools are represented by the 2D feature set described in Section 5.2, itself an extension of previous sets of 2D features, but applied here for the first time to consider the tool's pose.

On the second approach, published in [96], the focus is by contrast on discovering similarities among tools based on their functional features, and subsequently learning

an affordance model for each of the discovered tool categories. Tools are represented by the holistic 3D descriptor OMS-EGI detailed in Section 5.3, whose performance is assessed alongside the convenience of representing the tool's pose implicitly or explicitly.

The last approach developed during this Thesis, submitted to [98, 99], improves on the previous ones by introducing a gradual representation of both tools and affordances. This way, it avoids the imposition of categories onto spaces that actually vary in a smooth fashion. At the same time, it investigates what part of the information encapsulated in the OMS-EGI descriptor, surface or spatial, is more relevant for an interaction scenario such as the one considered in this Thesis.

Despite the shift in focus of the presented studies, they share some general aspects, which serve as a common ground to base and compare them. Specifically, they all apply the same formulation of the concept of affordances, and follow a generic common workflow. These elements will be presented next, before moving on to the detailed description of the individual experiments on subsequent sections.

6.1.1. Formalization

According to its most accepted definition [26, 48, 112, 150], affordances are the relationships between the elements of the tuple:

$$\{Action,\ Object,\ Effect\}. \tag{6.1}$$

In the cited studies the robot used its own manipulator to interact with objects, and observed the effect of its actions upon them. However, the manipulator itself was not considered as part of the affordance tuple. Therefore, this definition, while useful and enough for simple interaction scenarios, requires extension when considering any modification to the elements that the agent uses to interact with the environment; depending on the particular embodiment of the agent (including incorporated tools), the effect that an action has on an object, that is, the affordance, will vary. Indeed, the manipulator, and more generally, the body, places a crucial role in determining which affordances an entity provides to an agent. For example, a grasping affordance would never be available for a robot without arms, or the traversability one for a static robot.

Therefore, in recent studies on tool affordances in robotics, the previous formalization has been extended to consider further elements that can influence an agent's interactions with its environment, such as tool use or multiple objects. However, the problem with

defining and learning tool affordances is that different actions with different tools grasped in different ways upon different objects can generate different effects. If all these elements are represented in a single tuple, it would consist of 5 or even more elements, and the search space to prove and test all the possible combinations would be unfeasibly large, specially when dealing with actual robotic implementations, where exploration takes a non-negligible amount of time and physical resources.

Therefore, lacking a formal definition that encompasses all these elements, most extensions have been tailored to the particular requirements of each study. For example, on their work on tool affordance learning [74, 75], Jain & Inamura defined affordances as "The relational instance *{effect, (state, tool-part, action)}* i.e. potential to generate some effect, through application of some action using certain part of the tool given a certain environmental state". In their work, the environmental state refers to the features of a possible target object, but it is omitted during learning for simplicity. In fact, this is a common practice in many tool use studies, as it can be assumed that the object affordances can be learned beforehand as in [174]. In some cases, simplifying the target object representation also allows a feasible exploration of further degrees of freedom on the affordance tuple, such as the grasp, which would mean that the considered affordance elements are those belonging to the tuple *{Tool, Grasp, Action, Effect}* [199].

On the other hand, on [58, 34] tools are represented with the same features as objects, so the formalization could be written as *{Object1, Object2, Action, Effect}*, where the decision of which object corresponds to the tool is discovered through the relational structure, (Bayesian Network in [59], deep AutoEncoder in [34]). Other studies propose implementations where the applied formalization is not clear, such as in [171], where the tool is represented simultaneously as part of the body map (implemented with Multiple Time Recurrent Neural Networks), and on a visual representation which includes the target object and the arm too.

In the present work, as stated in Section 2.2.4, we consider that tools correspond to objects or elements that the action is performed *with* rather than *upon*, which are attached to and extend the robot's own manipulator. Thus, **tool affordances** should not be understood just as the effects that an agent can achieve on a certain object with another object, but as *the functionalities that intermediate embodied objects, through an action on an external entity, enable the agent to achieve.*

According to this definition, and aiming at unifying the formalization applied for tool affordance studies, we propose that a more comprehensive and general definition of affor-

dance should explicitly include the agent's embodiment, thus becoming the relationship of the elements in the tuple:

$$\{Body,\ Action,\ Object(s),\ Effect\}, \tag{6.2}$$

as it is the body which mediates all other possible interactions, and may also be modified. In fact, all studies on tool affordances where these are learned by the robot through interaction (that is, not applying solely computer vision approaches), are implicitly considering the agent's embodiment. Indeed, there is a large number of papers which study how affordances depend on the body, the body scale, etc, [197, 198, 26]. However, almost surprisingly, embodiment was never, to the best of our knowledge, explicitly stated on a formalization of affordances.

The proposed definition allows to clearly differentiate the role and formalization of tools from that of other objects. In this formalization, the *Objects* element represents the external entities to which the action is directed *upon*, while tools are comprised in the *Body* element, which encapsulates the elements that the action is performed *with*. It also makes the implicit assumption that tools are incorporated into the body schema, which is supported by a large body of studies in neuroscience [100, 44, 119, 11], as well as in many other robotic studies [116, 103, 122, 171].

Furthermore, if we observe this general definition closer, we can realize the intrinsic relationship between the two pairs of elements *{body, action}* and *{object, effect}*:

$$action = f(body) \tag{6.3a}$$

$$effect = g(object), \tag{6.3b}$$

where $f()$ and $g()$ represent generic functions that imply a change on the state of their argument. Indeed, actions are changes in the state of the body, the same way effects represent changes in the state of the object, as perceived by the agent. Thus, the previous formalization can be further compacted in the following tuple:

$$\{body,\ f(body),\ object,\ g(object)\}, \tag{6.4}$$

This succinct definition reduces the number of independent elements in the definition of affordances, while simultaneously being more general that the prevalent *{Object, Action, Effect}* definition. Moreover, this new definition automatically takes into account all the possible modifications and extensions to the agent's body, including the inclusion of extra parts or tools, and the parameters relevant to these, such as the grasp.

6.1.2. Affordance vectors

In all the experiments conducted during the development of the present Thesis, the elements of the affordance tuple were represented in the following way:

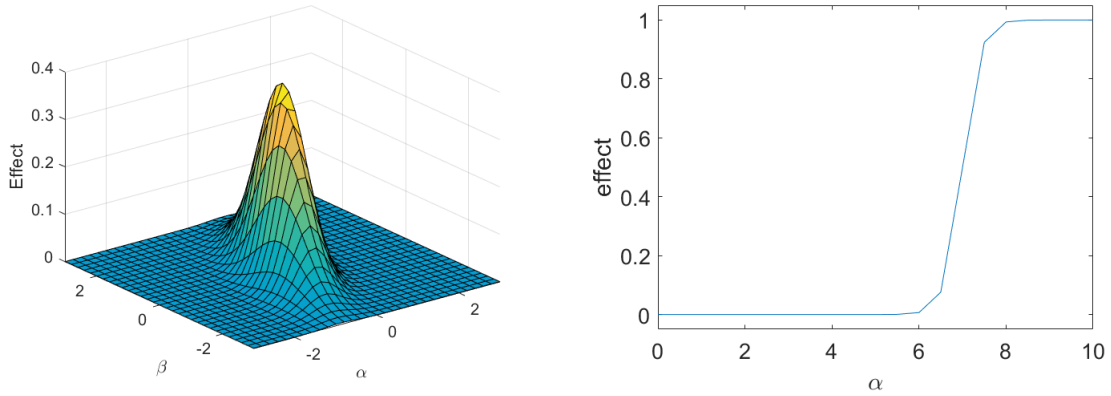
- **Body:** The embodiment is represented in terms of the tool-pose functional features described in Chapter 5.
- **Action:** Actions are implemented as set of parametrized motor primitives, and thus represented by the values given to the controlling parameters.
- **Object:** For simplicity, in terms of reducing the search space resulting from the combination of all the possible actions, tools, poses, objects and effects, we always apply the same target object and limit its representation to its location with respect to robot's reference frame, disregarding any properties such as geometry or material. We acknowledge, as stated above, that these properties do influence the effect of actions on the object, that is, the affordance, but as in the previous literature, we assume that it can or has been learned in previous stages of the robot development.
- **Effect**: Given that the target object is represented in terms of its location, the effect is measured in terms of the displacement that the tool action achieves on it.

Thus, in the practical implementation of the affordance formalization described in the previous section, the representation of the embodiment corresponds to the functional features that characterize the tool-pose, and the object element can be removed, given that it is constant for each experiment and hence does not influence the relationship among the rest of the terms. Accordingly, the actual implementation of affordance knowledge during the development of this Thesis was as the relationship between the representation of the terms

$$\{\textit{tool-pose, action, effect}\}. \tag{6.5}$$

Yet, in a interactive approach to tool affordance learning such as the present one, for every tool-pose grasped by the robot, a map which relates action and effect could be obtained by exploring the action repertoire. This map, in essence, represents the expected effect that that tool-pose can achieve in function of the action parameters (assuming that the target object is constant), or in other words, the affordance for that particular tool-pose for the given action repertoire.

In the general case, affordance maps can be applied when the effect can be repre-



(a) Example of 2D affordance map.

(b) Example of 1D affordance map.

Figure 6.1.1.: Generic examples of affordance maps, where the number of parameters N controlling the action is 2 in (a) and 1 in (b).

sented as one value per action, and the action described by N parameters that can be discretized. Then, an affordance map takes the form of a N-dimensional matrix where the discretized values of each of the parameters are represented by the indices of the matrix on each of the N dimensions, and the value at each combination of indices is given by the effect achieved by performing the action with the parameters corresponding to those indices. On Figure 6.1.1a, we can see an example where the action is represented by parameters α and β discretizable in values α_i and β_j. Thus, the affordance map corresponds to a 2D matrix where the axes indices correspond to α_i and β_j, and the value on each coordinate $e_{i,j}$ to the effect achieved by the action performed with those parameters: $e_{i,j} = f(action(\alpha_i, \beta_j))$.

In particular, in the present Thesis the action repertoire of all the experiments carried out is determined by one parameter, thus $N = 1$, and the map is therefore a vector, such as the example illustrated in Figure 6.1.1b. Thus, in the remainder we refer by **Affordance Vector** to the representation of the effect that can be achieved with a tool-pose in function of the applied action. This representation was applied in Tikhanoff et al. [174], where each considered object and tool was identified by a label and had an associated affordance vector obtained from interaction. However, assuming that similar tools have similar affordances (as discussed in Section 5.1), this approach can be extended to simplify the implementation of affordances, transforming the generic relationship between the elements *{tool-pose, action, effect}* to the specific mapping between the space of functional features (which represent tool and pose) and the space

of affordance vectors (which represent action and effect):

$$affordance\ vector = f(functional\ features) \tag{6.6}$$

This approach is advantageous for the actual implementation of tool affordance learning systems, given that most machine learning techniques available function as input-output, and only a few exist that can seamlessly learn relationships between 3 or more elements (as for example Bayesian Networks, applied in [112]), but hthey have other drawbacks. Moreover, it does not suppose any loss of generality, as affordance exploration is actually carried by a discrete set of tools (or tool-poses) so each of them can be associated with a prototype affordance vector from which the affordances of similar tools can be generalized.

6.1.3. Workflow

Despite their differences in focus and machine learning techniques applied, the general workflow is common among all the learning architectures that will be detailed in the following sections. A diagram describing this workflow can be observed on Figure 6.1.2.

In particular, all the performed experiments consists of three phases, data gathering, training and testing. Each trial of the data gathering phase starts with the robot grasping a tool in a particular pose. Then, the resulting tool-pose is observed and incorporated following the methods explained in Chapter 4 and its functional features extracted, as described in Chapter 5. Next, the robot localizes the target object and executes an action upon it using the tool, and after it is completed, observes the achieved effect on the object. For each tool-pose, the action is repeated with all the considered values of the action parameter, so that the full *Affordance Vector* corresponding to the tool-pose being grasped can be recorded.

Ergo, each trial of the data gathering phase consists in the execution of all the actions in the repertoire (determined by the possible values of the action parameter) for a given tool-pose, whereby the functional features of the given tool-pose and its corresponding affordance vector are recorded. In order to gather enough information to enable learning and generalization among different tool-poses, several trials are recorded for each tool-pose, and the process is repeated for a large number of different tool-poses. On the next phase, once all the data has been gathered, the affordance models are trained using these data in order to learn the general function that maps functional features to affordance

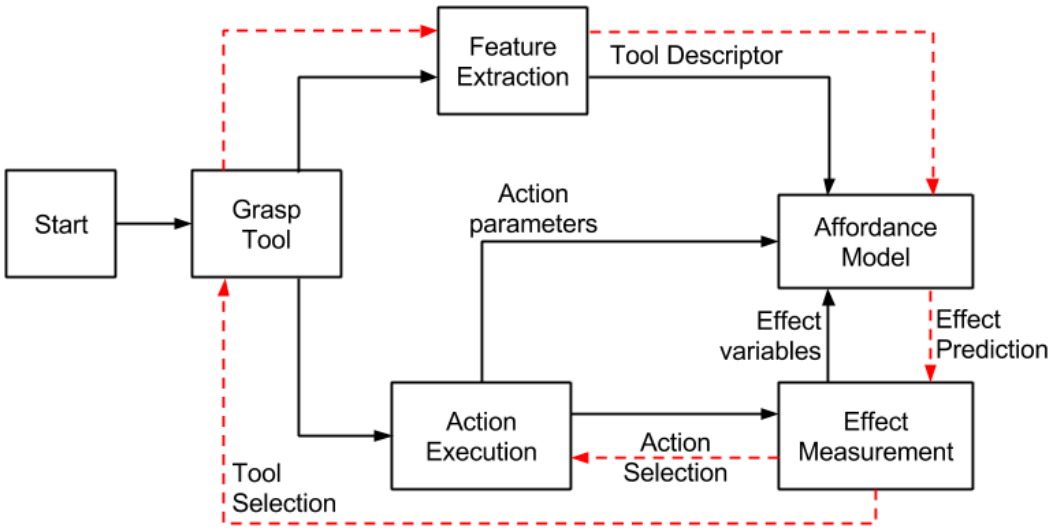


Figure 6.1.2.: Common workflow of proposed tool affordance learning architectures. During the data gathering phase (black arrows), the robot grasps the tools in the training set a number of poses, and for each tool-pose performs the actions in its repertoire, observing the resulting effect after each action. This way, for each tool-pose, its functional features and associated affordance vector are recorded. These data is then used to train an affordance model. On the test phase (red arrows), the model is queried to predict, given the functional features extracted from a tool-pose, the effect of all the actions in the repertoire. This prediction can be applied to test the learning and generalization capabilities of the system, for action selection, or to select another tool if no available action is adequate enough.

vectors, i.e. $f(\textit{functional features}) = \textit{affordance vector}$, and be thereby able to predict, given a tool-pose, the effect of all the actions in the repertoire.

On the test phase, the generalization capabilities of the trained system are evaluated by means of two possible procedures. On the first one, employed to assess the general prediction capabilities of the system, the gathered data is separated into training and test sets. Data from the former set is used to train the model, while from the second, the functional features are fed to the system to obtain predictions of the corresponding affordance vectors. In this testing procedure, the prediction performance is assessed by comparing the predictions returned by the model to the previously recorded affordance vectors.

The second evaluation procedure serves to evaluate the suitability of the proposed system for action selection. It is achieved by having the robot grasp a tool that has not been used for training the system, and incorporate it and extract its functional features as in the data gathering phase. In this case, though, instead of performing all

the actions, the extracted functional features are fed to the system, which returns the predicted affordance vector. Then, given a certain task, the robot either performs the action with the best predicted effect, or, if none is adequate enough, selects a different tool.

On the following sections we will describe in detail the different specific implementations and experiments that were carried out to test the paradigms presented in this Section. Each of them corresponds to a published paper; Section 6.2 to [97], Section 6.3 to [96] and Section 6.4 to [98] (and [99], accepted).

6.2. Discovering effect categories

6.2.1. Introduction

In the current Section we will describe the work presented in the the article "Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot" [97]. The proposed approach represents the first of the incremental steps at tool affordance learning developed during this Thesis.

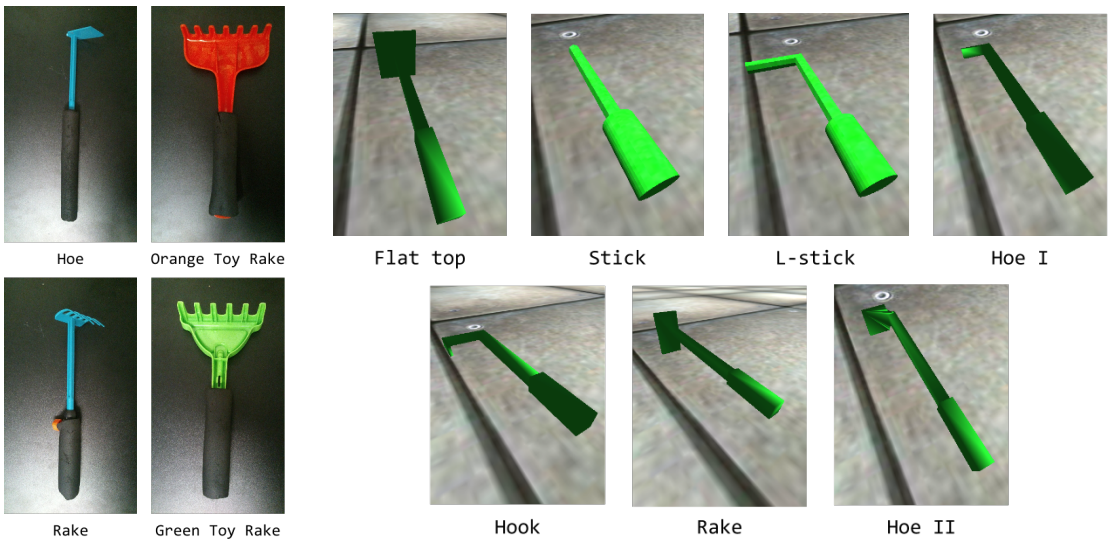
In this study tools were represented by the extended set of 2D functional features described in Section 5.2, and affordance vectors were applied to represent the effect of tool use. Together, they implicitly accounted for all the elements of the affordance tuple, as we saw in the Section 6.1. In particular, the pull-affordance of several tools was learned, taking also into account the way in which their effector was oriented.

Learning happened in a self-supervised manner, where first, the robot autonomously discovered the affordance categories of the tools by clustering the effect of their usage. These categories were subsequently used as a teaching signal to associate the 2D functional features to the expected tool's affordance. The advantage of this approach is that the tools are categorized visually depending on their effect, i.e. on the corresponding affordance vector. On the following, we describe the specific details of the proposed method, and the experiments conducted to evaluate it.

6.2.2. Experimental setup

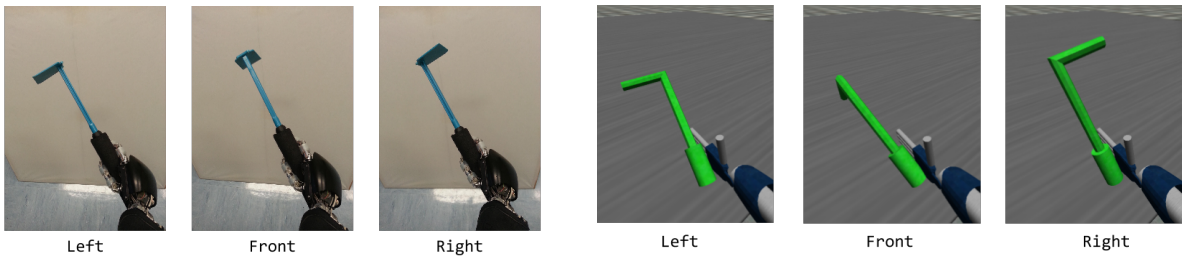
The experiments carried out in this study were performed using the iCub humanoid robot as well as its simulator, both described in Chapter 3. All modules concerning robot motor actions as well as sensory extraction and processing were written in C++, with extensive use of the OpenCV library for feature extraction. The data analysis and learning processes were programmed in MATLAB, making use of the third party libSVM library [23] to implement SVM learning algorithms.

4 different tools were used for the experiments on the real robot, and 7 tools for those on the simulator, which can be observed in Figures 6.2.1a and 6.2.1b, respectively. In order to study how the way in which a tool is grasped affects what it affords, 3 different tool-poses were considered for each tool: either to the front, to the right, or to the left, as shown on Figure 6.2.2. Therefore, 12 tool-poses were considered in the real robot and



(a) Real Tools. **(b)** Simulated Tools.

Figure 6.2.1.: Tools used on the experiment on the real robot (a) and simulation (b).



(a) Real robot. **(b)** Simulation.

Figure 6.2.2.: Visualization of the considered tool orientations on the real robot (a) and the simulator (b): left, center and right.

21 in the simulator. Actions with those tool-poses were performed upon a small cube of around 6 cm in side, placed on a table in front of the robot.

6.2.3. Discovering and learning pull affordances

The method used in the present study divides the affordance learning process in the following three stages:

1. Gather affordance data through interaction and observation of the effects.
2. Discover affordances by clustering the observed effects.

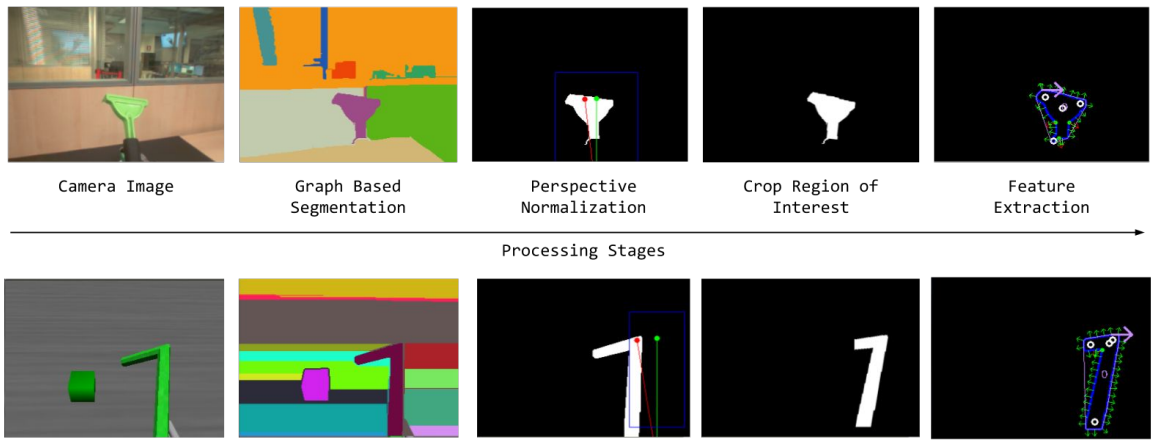


Figure 6.2.3.: Examples of the feature extraction process at different stages on the real (above) and simulated (below) setups. The process is described in detail in 5.2.

3. Mapping functional features to the discovered affordances.

As discussed in Section 6.1.3, each trial on the data gathering stage consists of three consecutive steps, feature extraction, action execution, and effect observation. In this study, feature extraction is performed as described in Section 5.2, using only 2D information from a single camera. This process also returns the estimated tooltip, which is attached to the robot kinematics so that it can be applied as the new end-effector. It should be noted that while the pose that the robot adopts in order to observe the tool and extract its features is always such that the whole tool is within its visual field, the specific position and orientation of the hand are slightly randomized so that no two sets of extracted features will be the same, even if the robot is holding the same tool in the same orientation in several observations. Examples of this process, on the real and simulated scenario, can be observed in Figure 6.2.3.

After the features have been extracted, in order to observe the pull affordance of the tool-pose being held, the robot performs a series of executions of a pull action upon the target object. An instance of one of these executions at different stages can be observed in Figure 6.2.4. These pull action is parametrized by α, which represents how much on top of the object or to either side of it the action is done, on integer centimeter values. Specifically, each training trial of the present study consists on 11 pull executions, corresponding to values of α from -5 to 5 cm to either side of the object. When reaching for the object, the tooltip is not directed exactly to the coordinates where the object is detected, but 3 cm behind, so that the end-effector is able, if the approach parameter is adequate, to grab the object during the pull action, which has a constant length of

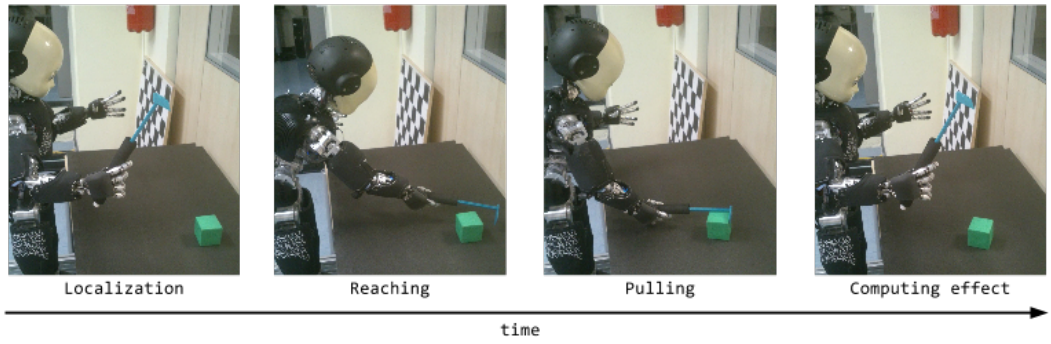


Figure 6.2.4.: Example of an instance of a single pull action at different stages.

20 cm. A diagram of the pull action with respect to the approach parameter can be observed in Figure 6.2.5.

In order to perform this task, a tracker module is applied to constantly locate the target object within the field of view, by making use of a particle filter which is initialized at the beginning of the experiment with a user provided template of the desired target object [68]. This way, it is able to reliably provide at any time the position of the pixel in the robot camera where the center of the target object is located. The transformation from the 2D coordinates in the image to the 3D coordinates in the robot's coordinate frame is achieved by the Gaze Controller module [132, 145], which takes advantage of the robot's kinematics to determine the direction of the gaze and project the point on the plane of the table, whose height with respect to the robot is known. Therefore, in order to execute a pull, the only required parameter is the approach position with respect to the object, as the object position is tracked automatically and the actions required to reach and pull are estimated and performed by the iCub's Cartesian Controller module.

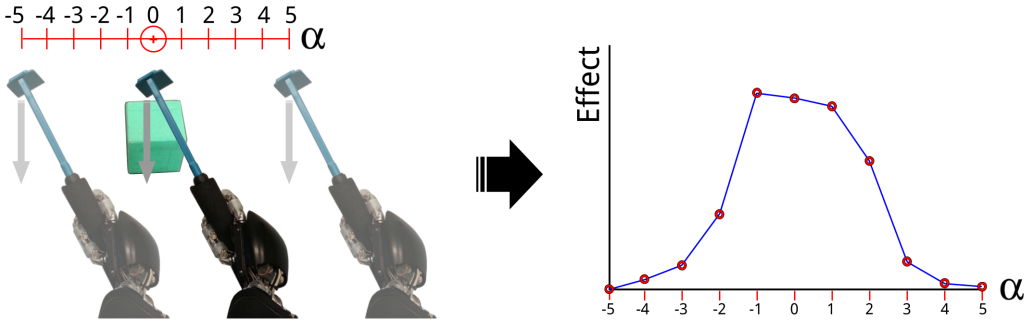


Figure 6.2.5.: Pull action sequence: The iCub performs pull actions parametrized by α, the horizontal distance of the approach to the center of the object, from -5 to 5 cm. On the right side a possible resulting affordance vector is displayed.

For each pull execution, the effect of the action is computed as the distance between the position of the object before and after the action, as provided by the tracker. If the object has been displaced, it is put back after the computation of the effect on approximately the same position as it was before the action. The 11 *{action, effect}* value pairs obtained by this process represent the affordance vector which describes how well a particular tool-pose affords pulling as a function of the approach position with respect to the object, as the example illustrated in Figure 6.2.5.

On the second stage of learning, the aim is to enable the iCub to autonomously discover how the individual affordance vectors recorded in the previous stage are distributed in certain categories, and to observe whether these might be commonly shared among different tool-poses.

To that end, we performed a series of K-means clustering runs among all the affordance vectors, regardless of the tool-pose that generated them. Because K-means does not yield an optimal number of clusters, we evaluated the clustering performance with a range of number of clusters K, ranging from the minimum ($K = 2$) to half of the number of the considered tool-poses, that is, to $K = 11$ for simulation data and $K = 6$ for real robot data. For each run, the quality of the clustering is assessed with the Davies-Bouldin clustering index [32], which measures how well separated the clusters are as the ratio between the average of the mean scatter of the points in each cluster and the average distance among their centroids. Thus, the number of clusters that better separated the affordance vectors could be determined as the K which produced the lowest DB-index, K_{best}.

This clustering procedure provided two valuable results for the next learning stage. On one hand, the cluster index given to each affordance vector served as a class label to classify the feature vectors. On the other, the centroid vector of each cluster also represented the prototype vector of the corresponding affordance category, which could be later used to evaluate the learning process as well as to compute the best action parameter.

The third learning stage deals with learning the mapping between the feature vectors extracted from the tool-pose contour and the affordance cluster to which their corresponding affordance vectors belong. To this end we applied Support Vector Machine classifiers, due to their good performance and readily available implementations [190].

In all the training schemes that have been carried out, the SVMs are batch trained offline using the full normalized feature matrix as input and the corresponding cluster

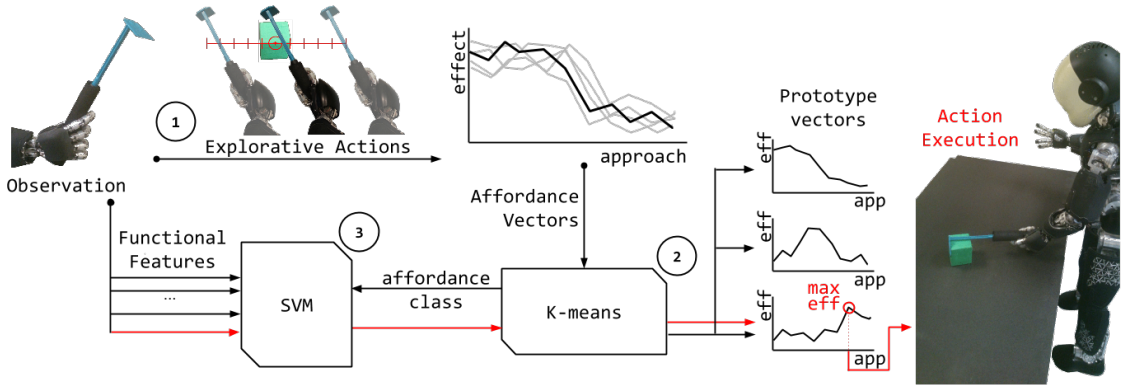


Figure 6.2.6.: Diagram of the proposed affordance learning and prediction method. Black and red arrows represent the flow of information in the training and prediction phases, respectively. The three stages of the training procedure are indicated by the circled numbers as follow: (1) Gathering of tool use data through explorative actions and observation of the effects; (2) Discovering affordance types by clustering the observed effects; (3) Mapping functional features to the discovered affordance classes. On the test phase (red arrows), after the functional features of a new tool-pose are extracted, they are fed to the trained SVM, which returns its predicted affordance class. The centroid of the predicted class is retrieved from the K-means, and the action for which the predicted effect is larger is executed.

indexes determined in the previous stage as target. As SVMs are always binary classifiers, K_{best} SVM classifiers are trained on *one-versus-all* discrimination. The SVM c and γ parameters are estimated using recursive grid search based on cross-validation results on a training subset. The parameters for which the average cross-validation accuracy is highest are used to train the SVM model. When evaluating the model, a fraction of the data is kept apart only for testing, while the rest is used to carry out the training processes just described.

A schematic diagram representing the flow of information and the modules involved in the process of discovering and learning affordances described above is shown on Figure 6.2.6 with black arrows.

6.2.4. Affordance prediction and action selection

After the training process, the robot has learned the mapping between the functional features of the different tool-poses and the category of the affordance that they are likely to achieve. Thereby, when holding a new tool-pose, the robot is able to compute the expected effect for each of the possible values of the approach parameter α, by retrieving the prototype vector of the predicted affordance category.

Based on this information, the parameter value that is expected to maximize the displacement of the object is chosen and the respective pulling action subsequently performed. This process is shown on Figure 6.2.6 superimposed on red to the diagram of the training process, on which it relies. The effect caused on the object by the robot's action is measured as in the training trials, by retrieving the coordinates of the object before and after the action has been performed, and measuring the euclidean distance between them. A video presenting a condensed version of the whole experiment can be watched at: https://www.youtube.com/watch?v=neiX_eP4qq4.

6.2.5. Results

6.2.5.a. Discovering affordances

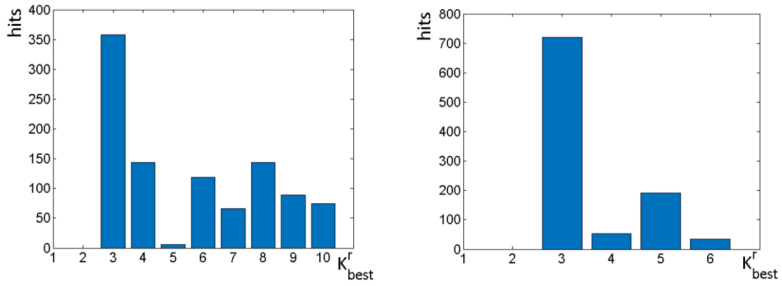
In the current study, between 20 and 25 affordance vectors were recorded for each of the tool-poses considered in simulation, and around 10 vectors for each of the tool-poses considered on the real robot, making up a total of 567 vectors (6237 pulls) on simulation and 138 vectors (1518 pulls) on the real robot.

After the data gathering stage had concluded, the existing affordance categories were discovered by clustering all the recorded affordance vectors by means of K-means algorithm, run with increasing Ks. However, due to the random initialization of K-means, the K_{best} number of clusters which minimizes the DB-index can vary significantly from run to run. In order to cope with this inconsistency, we ran 1000 times the procedure to find K_{best} described above, whose results are displayed on Figure 6.2.7. The prototype vectors of each affordance cluster, as represented by their centroids, are shown on Figure 6.2.8.

6.2.5.b. Affordance prediction and generalization

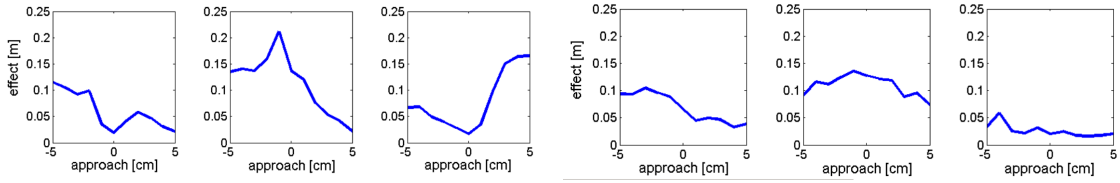
The assessment of the prediction performance of our system has been carried out in two different ways. On one hand, we measured how well the robot was able to predict the affordance from tool-poses which it had already experienced, while on the other we wanted to evaluate the generalization capability of this method to predict the affordance of previously unseen tool-poses, based only on their functional features.

Accordingly, on the first test the training subset is built by randomly gathering 3/4 of the vectors obtained with each tool-pose, while the test subset contains the remaining



(a) Simulated data **(b)** Robot data

Figure 6.2.7.: The histogram shows the number of times that each possible K was K_{best} based on the Davies-Bouldin index (left: simulated data, right: real robot).



(a) Simulated data **(b)** Robot data

Figure 6.2.8.: Prototype Vectors from the affordances discovered by K-means.

1/4 of the vectors. The training dataset is subsequently fed to the SVM classifier to learn the mapping between the set of tool-pose functional features of each training trial and the affordance category to which their corresponding affordance vectors had been clustered. For testing, the SVM classifier is presented only the feature vectors of the testing subset, for which it returns the indices of the predicted affordance categories.

By comparing the indices predicted by the SVM classifier with the ones that the recorded affordance vectors were assigned to, we can quantify the prediction accuracy. Moreover, by confronting the prototype affordance of the predicted cluster with the real affordance vector recorded during the experiment, we can graphically assess if the predicted affordance resembles the recorded one, and compute the mean square error from the distance between them. Table 6.2.1 shows the resulting values of accuracy and rooted mean square error (rMSE) obtained on this test with simulated data as well as with data from the real robot.

The second test determines how well the system is able to predict the affordance of previously unseen tool-poses. To this end, we performed a leave-one-out test, whereby data from one tool-pose at a time is used to test an SVM classifier which has been trained with all the data of all the remaining tool-poses. Therefore, this test involves

Test	Env.	Class. Acc. (%)	rMSE [cm]
Prediction	Sim.	81.9 %	6.4
Prediction	Robot	64.1 %	5.1
Leave-One-Out	Sim.	56.9 %	7.7
Leave-One-Out	Robot	53.9 %	5.4

Table 6.2.1.: Prediction Performance for known (prediction) and unknown (Leave-One-Out) tools.

training and evaluating as many SVM classifiers as tool-poses are being considered, so the overall performance displayed in Table 6.2.1 is measured by averaging across them.

6.2.5.c. Affordance based action execution

The last experiment that was carried out to evaluate our method consisted in applying everything that the iCub had learned so far in order to steer its behavior online. The procedure involved the iCub holding a tool in its hand and visually extracting its geometrical features, using them to perform online prediction of the affordance category of the tool, selecting the best action for the aim of displacing the object, and finally performing the action.

In this scenario there is no ground truth or labels against which to compute the performance accuracy, so instead we used two measures to assess the performance: On one hand, the amount by which the goal of displacing the object is achieved is measured by the percentage of trials in which the object was actually displaced (i.e. the distance measured by the tracker is more than 5 cm, to account for tracking error). On the other hand, the error in prediction performance was computed as the average of the absolute difference between the predicted effect of the action and the measured object displacement after the robot's action was performed.

In order to carry out this evaluation, 6 trials were performed with each tool-pose on simulation, while the real robot performed 3 trials per tool-pose, thus having a total of 126 trials in simulation and 36 trials on the real robot. Results can be observed in Table 6.2.2.

Environment	Goal Acc. (%)	Avg. Diff [cm]
Simulation	86.51 %	6.4
Robot	86.11 %	5.6

Table 6.2.2.: Action Selection Results

6.2.6. Discussion

This study presented a novel approach for the study of tool affordances in robotics, introducing several contributions to the state of the art. In it, we expanded the application of functional features for tool representation beyond the limited sets used before. We applied a comprehensive set of geometrical features extracted from vision which is not task-specific and whose features might relate to a number of different functions, as well as enable generalization among geometrically similar tools, while simultaneously taking into account how the tool is being grasped. Furthermore, tool-pose affordances were discovered by evaluating the effects of the robot actions, considering a set of effects generated by a parametrized action as a single affordance entity. This allowed the robot to predict not only a single effect for a particular action, but a whole set of them which can be used then to select the best action. Moreover, the proposed system learns tool affordances without the need for external labeling or supervision for classification, based only on the observation of the effects of tool use.

In particular, the results from the affordance discovery, Section 6.2.5.a, show that the discovered affordance categories resemble the different effects that one would expect to find. Indeed, if we look at the results in Figure 6.2.7 we can observe that the number of clusters which was more likely to be K_{best} was 3. This means that the effects that the different tool-poses achieve on the object naturally grouped into 3 affordance categories, both in real and simulated experiments. Moreover, as can be seen in Figure 6.2.8, the prototype vectors of each affordance cluster, as represented by their centroids, display large effect when the approach is either on the right, the center or the left of the object, respectively, coinciding with the three tool orientations that were considered.

In is also noteworthy that the prototype vectors from simulated data represent more distinct behaviors than those from robot data. This is due to the larger noise in the effect of real robot actions, which leads to smeared out centroids, and the fact that due to the angle of approach, the object was only seldom displaced when the approach position was to its right, even if the tool end-effector was oriented to the left.

Moreover, results on Section 6.2.5.b show that the method allows the iCub to predict quite reliably the affordance category of tool-poses that have been previously observed, based only on its visual features. On Table 6.2.1, we can observe that results for prediction of the affordance of known tools obtained for simulation are almost as high as we could expect, given the self-supervised nature of the method, which already introduces inaccuracies on the target signal. On data from the real robot, however, while accuracy is still almost double over chance, it is considerably lower than on the simulator. This decrease in performance can be explained by the fact that both affordance vectors and segmented tool blobs on the real robot experiments are noisier than those on simulation, as well as by the smaller amount of data available to train the classifier.

Results yielded by the leave-one-out test are slightly worse, but still about 20% over chance, which means that although there is plenty of room for improving the method, it already allows the robot to generalize its knowledge to previously unseen tools. One of the issues that drove down the performance of the classifiers, other than inaccuracies in the blob extraction and clustering procedures, is class imbalance, meaning that one of the affordance clusters usually dominated over the remaining. This led the classifiers to disproportionally classify feature vectors as corresponding to the dominating affordance category.

Nevertheless, when applying the learned methods in order to steer the robot based on its affordance knowledge, results in Table 6.2.2 show that for a large majority of trials, the robot achieves its goal of moving the object with the tool, and does so with an average error very similar to the one achieved in the off-line experiment for affordance prediction on known tools. It is noteworthy that the accuracy in achieving the displacement of the object is higher than the affordance prediction accuracy. This can be supported by the fact that, sometimes, even if the affordance category is wrong, it is usually close enough to the correct one that the maximum values of its prototype vector leads to an action that still achieves the goal.

Overall, results demonstrate that although the approach has some shortcomings that should be addressed, it is possible to predict the affordances of radial grasp tools from their 2D geometry taking into account the way in which they are grasped.

6.3. Discovering tool categories

6.3.1. Introduction

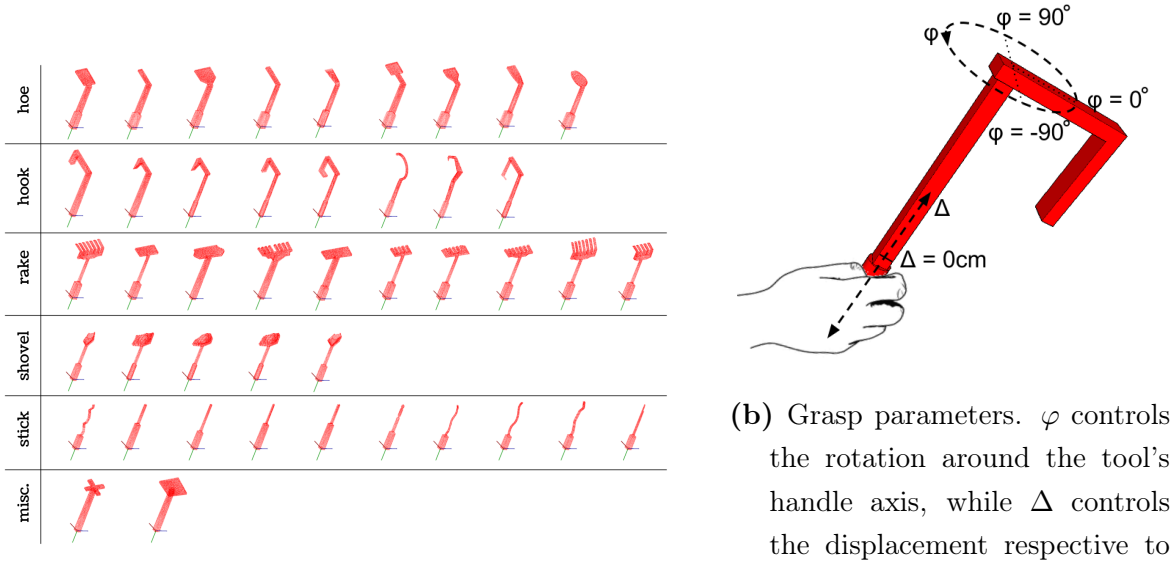
In this Section we will present the work carried out for the article "Multi-model approach based on 3D functional features for tool affordance learning in robotics" [96]. The main aim of this paper is to investigate the effect of geometrical features on the tool affordance. To that end, we introduced substantial changes with respect to the approach detailed in the previous section, and switched the focus towards studying the prediction of affordances based on the similarity among tools in terms of their geometrical features, rather than only effect categories.

In particular, we introduced the Oriented Multi-Scale Extended Gaussian Image (OMS-EGI), a set of 3D features devised to describe tools in interaction scenarios, able to encapsulate in a general and compact way the geometrical properties of a tool relative to the way in which it is grasped, described in detail in Section 5.3. In order to provide more insight on the possible applications of the OMS-EGI features, we compared their performance when the tool was represented in a canonical pose against the case where the tool was represented by the oriented pointcloud model.

Then, we proposed an approach to learn and predict tool affordances whereby the robot first discovered the available tool-pose categories of a set of radial tools, based on their OMS-EGI features, and then learned a distinct affordance model for each of the discovered tool-pose categories. We argued that in order to ease learning and enable more precise affordance predictions for tool use scenarios, instead of learning a single model that tries to relate all the possible variables in an affordance, the robot should learn a separate affordance model for each set of tool-poses sharing common functionality.

6.3.2. Experimental setup

The experiments carried out in the present study were performed using only the iCub simulator, fully described in Section 3. The processing of 3D models and feature extraction was implemented using the Point Cloud Library [148], while experimental data analysis including learning and visualization was implemented in MATLAB, relying on the third party `SOM_Toolbox` [192] for Self-Organized Map analysis and on the built-in Neural Network Toolbox for regression analysis.



(a) Tool Dataset

(b) Grasp parameters. φ controls the rotation around the tool's handle axis, while Δ controls the displacement respective to the hand on the same axis.

Figure 6.3.1.: (a) Dataset of tools used in the current study. (b) Visualization of parameters controlling the pose of each tool on the experimental trials.

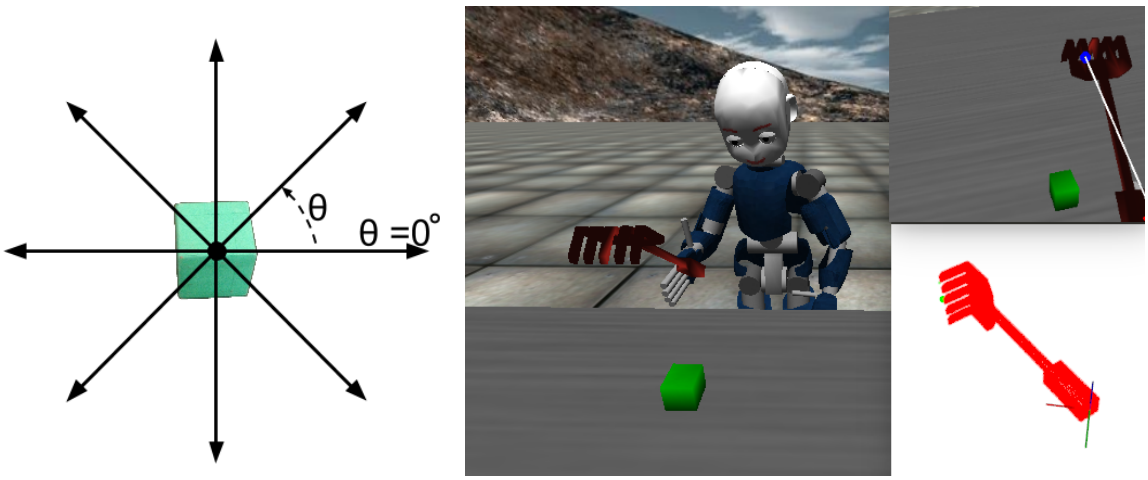
As for the experiment itself, we have used 44 different virtual tools which roughly correspond to 6 different categories, as can be observed in Figure 6.3.1a. The pose with which each tool was grasped for each trial was determined by the following parameters:

- *Grasp orientation (φ)*: Controls the angle with which the end effector is rotated around the tool's longitudinal axis. At $\varphi = 0$, the end effector is looking to the front with respect to the hand, i.e. in the direction of the extended index finger (or along the X axis in the iCub's hand coordinate frame). $\varphi = 90$ corresponds to the end effector oriented to the left (along the Z axis of the hand reference frame), while $\varphi = -90$ represents the end effector oriented to the right (-Z axis hand reference frame).

- *Grasp displacement* (Δ): Controls the position of the grasp along the handle, in cm. At $\Delta = 0$, the base of the tool is at the height of the little finger, which corresponds to the Y axis origin of the iCub hand reference frame. Positive values for Δ correspond to the tool being in the direction of the extended thumb (-Y axis hand reference frame in the simulator), while negative Δ values mean that the tool is grasped from a position in the handle closer to the effector end.

The meaning of these parameters can also be observed graphically in Figure 6.3.1b.

The set of actions that the robot could perform in the current experiment was limited



(a) Diagram of the drag action. **(b)** iCub Simulator Setup

Figure 6.3.2.: (a) Diagram of the drag action for different values of *theta*. (b) iCub Simulator setup for learning tool use through interaction, example when using a rake.

to a drag action parametrized by $\theta \in \{0, 360\}$. θ corresponds to the angle (in degrees), along which the robot tries to drag the object, as displayed in Figure 6.3.2a. The tooltip is initially placed slightly behind the object, and then the tool is displaced 15 cm along the radial direction given by θ. We implemented this set of actions so that it is simple to represent (defined by a small number of parameters) while still being able to discriminate between different tools and different grasps. In other words, so that different tool-poses would generate different effects. This allowed us to focus on the study of the suitability of the features and tool type discovery strategy, which were the main aim of the study. Also, this kind of drag action resembles the one commonly used in psychological experiments to assess the ability of macaques or babies to perform tool use [73, 193, 189].

The target object was a cube whose initial position before each drag action was fixed at 40 cm in front of the iCub and 10 cm to the right side of the robot's sagittal plane on a virtual table of known height, so that the iCub will always use the right arm holding the tool to perform the action. The setup can be observed in Figure 6.3.2b.

6.3.3. Learning architecture

6.3.3.a. Functional tool-pose clustering

The main goal of this paper is to study how could a robot benefit from the fact that similar tool-poses have similar affordances. To that end, we applied here the OMS-EGI features (detailed in Section 5.3), which were devised as a means to encapsulate the geometry and pose of a handled tool. Moreover, in this study we wanted to assess whether affordance prediction was more accurate when the tool pose information was provided explicitly as an input to the affordance model, or rather implicit as part of the tool representation.

Therefore, for every tool-pose observed on the data gathering part of the experiment (following the workflow described in Section 6.1.3), two different sets of OMS-EGI features were extracted. In the first schema, which we refer to as *Oriented features*, the OMS-EGI descriptor was computed from the oriented pointcloud model, that is, the tool representation in the actual pose it was being held by the robot to interact with the environment. Thus, the *Oriented Features* OMS-EGI implicitly encode grasp information.

In the second schema, referred to as *Canonical features*, OMS-EGI features were extracted from the pointcloud model of the tool being oriented in its canonical pose (described in Section 4.5), therefore not matching the pose of the actual tool in the simulator. The grasp information is thus not encoded by the *Canonical Features* OMS-EGI, since it is constant for each considered tool, independently of how it is being grasped by the robot.

In order to make sense of these features, however, the robot needs to analyse a large set of tool-poses and find out the eventual common characteristics among them, thus discovering the available tool-pose categories. To that end, unsupervised clustering was applied on both OMS-EGI datasets. The method chosen to do so was Self-Organized Maps based K-Means (SOM K-means), due to the relative high-dimensionality of the feature vector when compared with the available number of samples, which cause simple K-means to yield very irregular and unbalanced results.

SOMs provide a lower dimensional representation of the input data based on an iterative method of vector quantization [81], on which K-means can be performed without the issues appearing when applying it directly on the higher dimensional data. Still, K-means is very sensitive to the initialization conditions, and does not provide an auto-

matic way of selecting K. In this study, we improved the way of selecting K by applying a cluster quality index Q, which combines the Davies-Bouldin index applied in the previous study with another term to promote cluster balance. Specifically, the cluster quality index Q is defined as

$$Q = d + 2b \tag{6.7}$$

, where d corresponds to the Davies-Bouldin index [32], which represents the ratio of of within-cluster and between-cluster distances:

$$d = 1/k \sum_{i=1}^{k} (max_{i \neq j}\{D_{i,j}\}),\ where \tag{6.8}$$

$$D_{i,j} = \frac{intraDist_i + intraDist_i}{interDist_{i,j}} \tag{6.9}$$

The term to promote cluster balance, b, corresponds to the standard deviation of the histogram of the cluster indices from the K-means runs, normalized by dividing by the maximum value of the histogram. That is, if $\mathbf{c} = (c_1, ..., c_i, ..., c_n)$, the clustering indices returned by K-means, where c_i is the cluster index assigned to tool-pose i, then

$$b = std(hist(\mathbf{c})/max(\mathbf{c})) \tag{6.10}$$

$$\tag{6.11}$$

Finally, b was weighted by a constant that determined its influence over the Davies-Bouldin index, which we set to 2.

6.3.3.b. Tool-pose category dependent affordance models

Once the set of available tool-pose categories has been discovered by clustering the OMS-EGI features with the methods described above, the robot should learn what the common affordances of each tool-pose category are. To the best of our knowledge, all previous affordance studies apply a single-model approach, where only one model aims at learning the relationships between the terms in the affordance tuple for all their possible values.

Based on the assumption that tool-poses clustered together share common geometrical properties and hence also similar functionality, we propose instead a multi-model approach, where a distinct affordance model is learned for each discovered tool-pose category. In order to do so, each tool-pose is assigned a category c_i, among the discovered

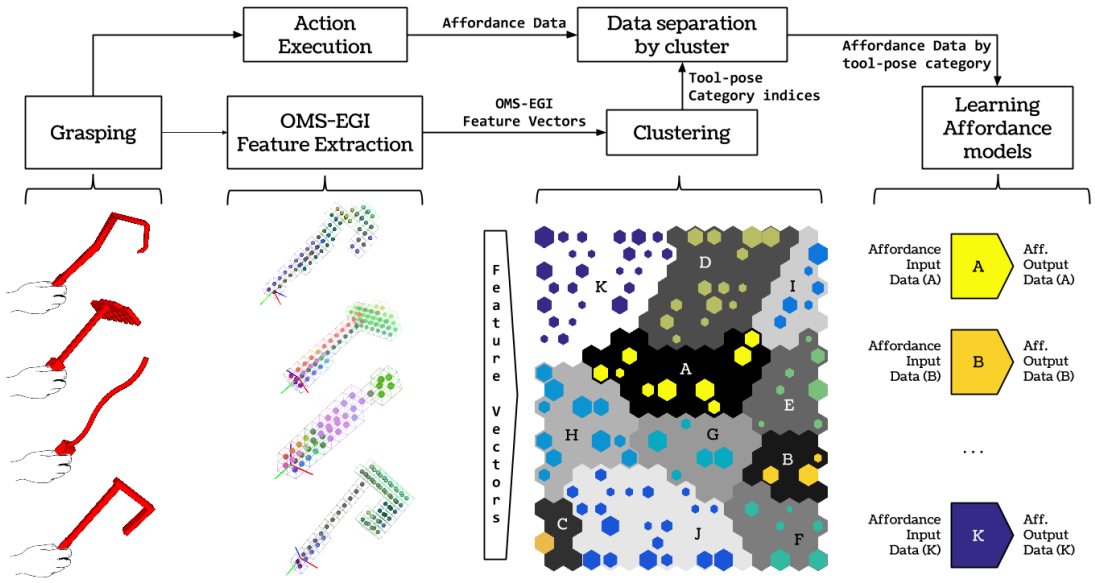


Figure 6.3.3.: Diagram of the proposed multi-model approach for tool affordance learning. From left to right: OMS-EGI features are extracted from the tools 3D models, and subsequently clustered using SOM K-means. Then, recorded affordance data is divided into subsets according to the cluster c_i to which the corresponding tool-poses belong. Finally, each affordance data subset is used to train a separate affordance model, so that each of them models the affordances of a particular tool-pose category.

ones. Then, the affordance data gathered through interaction is divided into subsets according to the categories of the tool-poses that generated it. Finally, a separate affordance model is trained with the data in each of the data subsets. An explanatory diagram of the proposed approach can be observed on Figure 6.3.3.

Affordance data refers here to the data gathered during the exploration phase, except the OMS-EGI functional features which are used to discover the tool-pose categories. In particular, these data consisted of the values representing the following terms:

- *Grasp:* Represents the way in which the tool is being grasped, in terms of the two grasp parameters φ and Δ. These parameters were provided to the affordance models only in the case where they represented the *Canonical features* categories, given that on the *Oriented features* scheme the tool pose is implicitly represented on the OMS-EGI, and thus the tool-pose categories already contain information about the tool's pose.
- *Action:* Represents the angle of execution of the drag action, represented by θ.

- *Effect:* The effect of the robot's tool use was measured as the euclidean distance between the object's position before and after the action execution.

Thus, the data required in order to train the affordance models depends on the feature schema that is being used. On the one hand, the *Oriented features* represent grasp information implicitly, so the distribution of the resulting tool-pose categories reflects this information. Consequently, when using the *Oriented features* grasp information does not have to be provided explicitly to the affordance models, which map directly *action* $\rightarrow$ *effect*.

On the other hand, the *Canonical features* represent tools always on their canonical pose, independent of how they are actually being grasped. Therefore, no grasp information is encoded in the distribution of tool-pose categories, so it has to be provided explicitly. Thus, when using the *Oriented features* the resulting affordance models perform the mapping *{grasp, action}* $\rightarrow$ *effect*.

In other words, when using the *Oriented features* schema, categories represent tool and pose implicitly, so each model needs to know only the desired action to predict the effect. By contrast, the categories obtained when clustering the *Canonical features* represent only the geometry of the tools. Therefore, in order to predict an effect, grasp and action parameters are required.

Independently of the feature schema, all affordance model inputs and outputs are real values, so the learning problem becomes one of regression: $\hat{e} = f_{tp}(I)$ where $\hat{e} \in \mathbb{R}$ is the predicted effect, f_{c_i} the affordance function to learn for each tool-pose category, and I is the affordance input. For *Oriented features*, where the only input is the action parameter, $I = \theta \in \mathbb{R}$. For *Canonical features*, where the grasp parameters are also fed to the affordance models, $I = \{\theta, \varphi, \Delta\} \in \mathbb{R}^3$. Given that the elements present in the tuple are relatively low dimensional, the regression models do not need to be utterly complex. In the present work, we use generalized regression neural networks (GRNN), a modification of radial basis networks which is able to approximate arbitrary functions and avoid local minima with 1-pass training [160]. This kind of networks have a single hyper-parameter σ which serves a regularization parameter by controlling the spread of its radial basis function. In order to find the optimal σ for each affordance model, we performed recursive line search based on cross-validation results on the training subset. The parameter σ for which the average cross-validation accuracy is highest is used to train the final model for each tool-pose category.

6.3.4. Results

6.3.4.a. Experimental data collection and separation

In the current experiment, 44 virtual tools represented by between 1500 and 4500 points each (see Figure 6.3.1a) were used by the iCub in simulation to gather interaction data. Each tool was grasped in 9 different poses, corresponding to the combinations of 3 different grasp orientations ($\varphi = \{-90, 0, 90\}$) and 3 different grasp displacements ($\Delta = \{-2, 0, 2\}$), adding up to a total of 396 tool-poses. For each tool-pose, two OMS-EGI feature vectors were computed and recorded: *Oriented* and *Canonical*, as described in Section 6.3.3.a. With each considered tool-pose, the iCub performed the drag action described in Section 6.3.2 in 8 directions, corresponding to angles θ from 0 to 315 degrees in intervals of 45 degrees ($\theta = \{0, 45, 90, 135, 225, 270, 315\}$), thus executing a total of 3168 actions.

For each of these actions, all the affordance data values ($\{grasp, action, effect\}$) were recorded, and linked with the tool-pose that generated them and its two OMS-EGI descritpors. Before any further processing, these data were separated in training and tests sets, which remained constant throughout the experiment and the off-line data analysis. 75% of all the tool-poses were selected randomly, and all the data associated with them used for training. The data corresponding to the remaining 25% of the tool-poses were used for testing. Thus, the training set consisted of the *Oriented* and *Canonical* OMS-EGI features of 297 tool-poses, and the 2376 affordance data vectors associated with those tool-poses, while the test set was formed by the OMS-EGI vectors of the remaining 99 tool-poses, and their corresponding 792 affordance data vectors.

Furthermore, given that there is no ground truth for the clustering process, and that interaction data is itself very noisy, errors might appear even if both processing steps worked perfectly. Hence, we needed to set a performance baseline against which we could compare the performance of our approach. To that end, we carried out an additional data processing run for each feature schema whereby after performing clustering, all the tool-pose category indices were shuffled. Therefore, the affordance data lost its correspondence with the tool-poses that generated it. We refer to these data as *Oriented-shuffled* and *Canonical-shuffled*, for the *Oriented* and *Canonical* feature schemes, respectively.

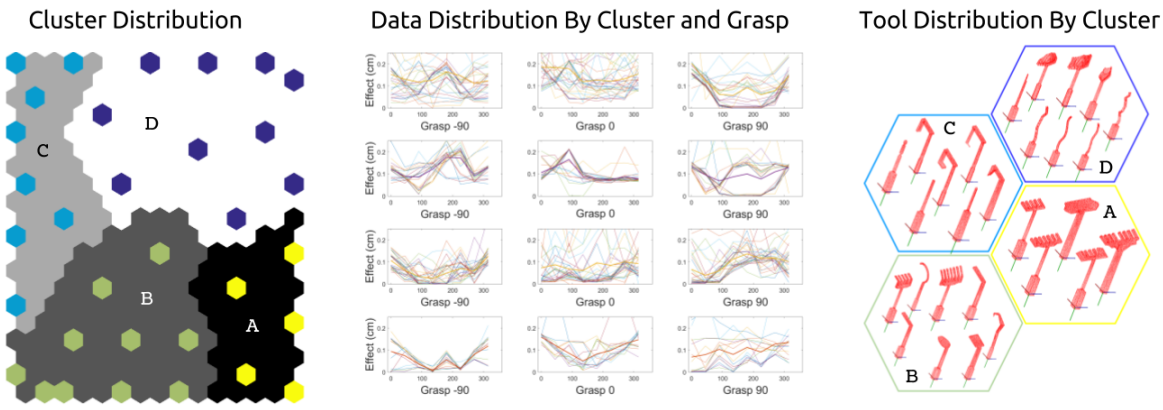


Figure 6.3.4.: Example of clustering results on training data for *Canonical features*. Left side: cluster distribution on the SOM with superimposed best-matching units of the clustered samples. Center: Train affordance vectors separated by cluster (rows) and grasp (columns). Right: Tool distribution by cluster, illustrated with a subset of the models in each cluster surrounded by a hexagon of the corresponding color.
The tools in each cluster, represented as `name of tool + [tool indices]`, are the following:
A: rake [2, 3, 5, 6, 7],
B: hook [5, 7]; rake [8, 9]; hoe [0, 1, 3, 4, 8, 9],
C: stick [3, 2, 5]; hook [1, 3, 4, 8],
D: shovel [4, 5, 6, 9]; stick [0, 6, 7, 8, 9] ; star.

6.3.4.b. Discovery of tool-pose categories

Once the data had been sorted out, the first step in our approach to model tool affordances was to discover the available tool-pose categories by clustering the OMS-EGI features belonging to the training dataset. In this study, we applied the BALAN parameters setting for the OMS-EGI, that is, $D = 2$ and $N = 2$ (see Section 5.3). Thus, the total length of the OMS-EGI feature vector is $S = 296$ features. When performing the analysis of *Oriented features*, each tool-pose produced a distinct OMS-EGI vector, which thus add up to a total 297 samples clustered. In the case of *Canonical features*, as the resulting OMS-EGI vector was always the same for each tool, only 33 distinct OMS-EGI vectors were extracted and clustered. The clustering results for *Canonical* and *Oriented* schemes can be observed on Figures 6.3.4 and 6.3.5, respectively. It must be noted that given the random nature of the train/test data separation as well as the K-means initialization, the displayed figures correspond to representative runs.

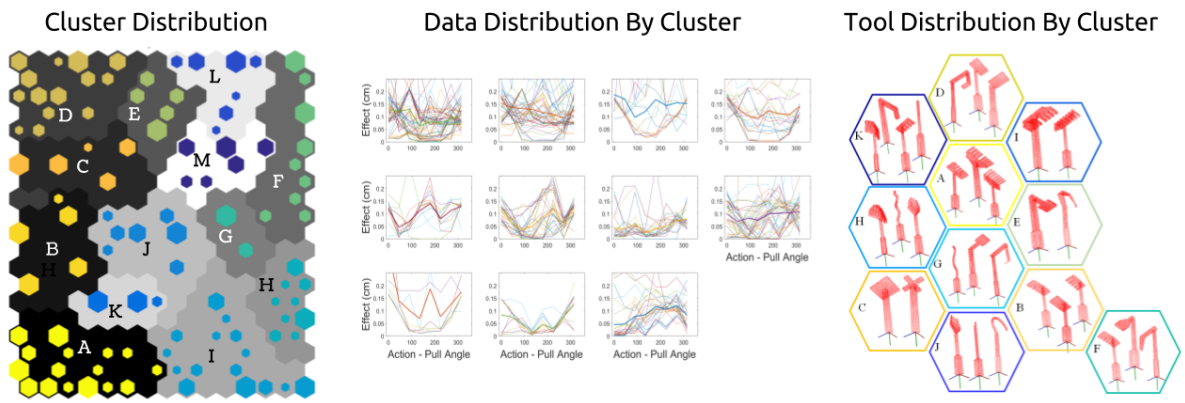


Figure 6.3.5.: Example of clustering results on training data for *Oriented features*. Left side: cluster distribution on the SOM with superimposed best-matching units of the clustered samples. Center: Train affordance vectors separated by cluster. Right: Tool-pose distribution by cluster, illustrated with a subset of the models in each cluster surrounded by a hexagon of the corresponding color.
The tool-poses in each cluster, represented as `name of tool + [tool indices] + (grasp orientation)`, are the following:
A: rake[0,1,2,3,4,5,6,7,8,9]:(90),
B: rake[1,2,3,4,5,6,7](-90),
C: flatTop:(-90, 0, 90); star(-90, 0, 90),
D: rake[0,1,2,3,7,6,9](90); hoe[0,1,2,3,4,5,7](90); hook[0,2,3,4,8](90); rake[0,6,1,5,7,3](-90); hoe[1,2,5, 0,4,7,8](-90); hook[4,2,0,7,5](-90),
E: hoe[0,1,2,9,5, 3, 4,7,8](0); hook[5,7](0),
F: rake[0,6,8,9](-90); hoe[0,1,3,4,5,7,8](-90); hook[0,1,2,3,4,5,7,8],
G: hoe[5](90); hook[0,1,2,3,4,5,7,8](90); stick[7](90),
H: stick[0,8](0); stick[0,6,8,9](90); shovel[4,5,6,7,9](-90); shovel[4,6,9](0);
I: rake[0,1,2,3,6,7,8,9](0),
J: stick[1,2,3,4,5,6,7,9](-90); stick[2,4,6,7,9](0); stick[2,5,7](90); shovel[5,7](0); shovel[4,5,6,7,9](90); hook[1,2,3,4,8,](0),
K: shovel[4,5,7,9](-90) / [4,6,7](0) / [4,5,6,7,9](90); hook[0,1,2,3,4,5,7,8,9](0); flatTop(0) star(-90); (0); (90); stick[1,6,7](-90)/[2,3,4,5,6,7,8,9](0)/[1,2,6,8,9](90); hoe[0,1,2,3,5](0); rake[0,1,2,3,4,5,6,9](0).

6.3.4.c. Prediction of tool-pose affordances

Through the clustering procedure, 11 tool-pose categories were discovered for the *Oriented features* and 4 for the *Canonical features*. In each case, an affordance model was instantiated per discovered category, and trained with the affordance data generated by the tool-poses belonging to the category it represented. As described in Section 6.3.3.b, each affordance model was implemented using a GRNN whose σ parameter was determined by recursive line search based on cross-validation.

Then, in order to evaluate the validity of our approach, we assessed the predicted effect values obtained with the data from the tool-poses belonging to the test set, which had not been used either on the clustering procedure or for training the affordance models. To that end, the first step was to classify the test OMS-EGI features into the previously discovered categories. This was done by finding the best matching unit (BMU) and corresponding cluster on the trained SOM of each test OMS-EGI feature vector. Then, the affordance input I data associated with the test tool-poses was fed to the corresponding GRNN affordance models, which returned the predicted effect for each of the test tool-poses. Finally, we computed the Mean Absolute Error (MAE) between the average predicted effect $\hat{e}$ and the average recorded effect e.

$$MAE = mean(abs(e - \hat{e})) \tag{6.12}$$

This process was repeated with the Shuffled datasets in order to obtain the Baseline performance. The prediction MAE achieved in this case is referred to as MAE_{BL}. By comparing the prediction performance in both cases, we measured the Percentage of Improvement (PI), which provides a measure of of the increase in prediction performance due to the learning process.

$$PI = \frac{MAE_{BL} - MAE}{MAE_{BL}} \tag{6.13}$$

Moreover, given that the train/test separation and the K-means initialization for clustering were random, every process was run 100 times in order to have reliable results. The average results are shown in Table 6.3.1.

Figure 6.3.6 shows the predicted affordance vectors, compared with the recorded ones, on a representative run. On the results shown for the *Oriented features*, each graph corresponds to a tool-pose category. On the ones for *Canonical features*, however, the data was divided according to the grasp that generated it (left, center, right) for visualization and evaluation.

	MAE	MAE_{BL}	$PI(\%)$
Canonical	5.37 ± 1.01	5.88 ± 0.88	8.67
Oriented	5.10 ± 0.13	5.78 ± 0.15	11.8

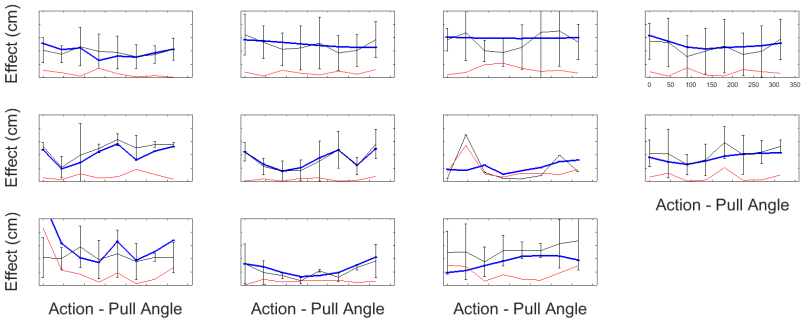
Table 6.3.1.: Prediction Error for *Oriented* and *Canonical Features* and baseline for comparison and the corresponding percentage of improvement.

Observing Figure 6.3.6a, we can see that for *Oriented features*, the affordance model predictions match the average recorded effect of their corresponding tool-pose categories quite well in most cases, even in some of the categories with larger variance among tool-pose individual effects. These results mean that the clustering step was successful in discovering and partitioning the oriented tool-poses into functionally similar categories, which in turn means that the OMS-EGI extracted from oriented tool models provided enough information to do so. On Figure 6.3.6b we can notice that the distance between the predictions for *Canonical features* and the recorded effects are larger than those for Oriented features. This observation is also supported by the numerical results on Table 6.3.1, which show that the *Oriented features* schema achieves higher accuracy on the affordance prediction.

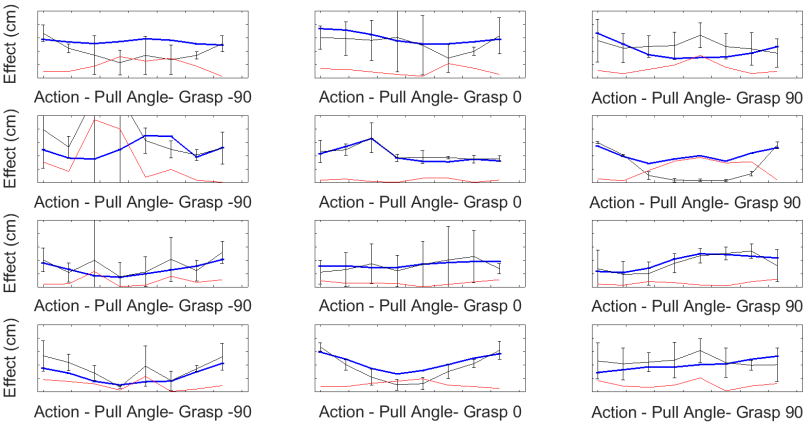
6.3.5. Discussion

In the present study we tackled the question of how could robots take advantage of the fact that similar tool-poses have similar affordances. In doing so, we presented two novel contributions to the field of tool affordance learning in robotics. On the one hand, we introduced OMS-EGI, a set of 3D features devised specifically for tool use scenarios. This set of features encapsulates the geometry of a tool and the way in which it is grasped. It has been shown that thse features relate nicely with the functionality of the tool and allow to predict it. We have also determined that these features provide more information about the tool functionality when extracted from the oriented 3D models than when extracted from a canonical pose, even if combined with explicit grasp information.

On the other hand, we proposed a multi-model approach where instead of a single model aiming at generalizing all possible cases, a different affordance model is learned for each tool-pose category, where categories are found by clustering the tool poses based on their geometrical properties. Results show that the combination of OMS-



(a) Oriented features.



(b) Canonical features.

Figure 6.3.6.: Prediction results on the test data by (a) tool-pose category for *Oriented features* and (b) considered grasp for each tool-pose category for *Canonical features*. The blue line represents the average affordance model's prediction for each possible action. The black line represents the average recorded effect for all the tool-poses determined to belong to the corresponding category, where the vertical errorbars represent its standard deviation. The red line represents the absolute error between recorded data and prediction. Effect axis in all graphs spans from 0 to 20 cm, while Action-Pull ranges from 0 360 degrees.

EGI 3D features and multi-model learning approach is able to produce quite accurate predictions of the effect that an action performed with a tool grasped on a particular way will have.

6.4. Learning affordances without categories

6.4.1. Introduction

The current section introduces the work presented in the articles "Self-supervised learning of tool affordances from 3D tool representation through parallel SOM mapping" and "What can I do with this tool? Self-supervised learning of tool affordances from their 3D geometry" [98, 99] (both accepted for publication). These studies presents a novel approach towards endowing humanoid robots with the skill to autonomously learn tool affordances and generalize the knowledge to similar tools, building upon the work done on the studies presented in Sections 6.2 and 6.3. Its contribution is twofold: On the one hand, we further investigate the suitability of robot-centric 3D features to describe tool-pose affordances, by comparing the performance among different variants of the OMS-EGI descriptor. On the other hand, we present a novel learning architecture which combines and improves the work presented in Sections 6.2 and 6.3 by linking the representation of tools based on their 3D geometry with the discovered affordances, which, unlike the work in Section 6.3, allows to group together tools that have different geometrical features but similar affordance. Moreover, it does so by applying a fine grained (gradual) representation of tools and effects that avoids the need to categorize either, resulting in a much higher predictive performance.

The assessment of the applications and potential of the OMS-EGI descriptor is performed by comparing the three parameter settings described in Section 5.3, namely EGI, OCCUP and BALAN. These feature set variants describe the tool-pose based on its surface normal histograms, its occupation of the space within its bounding box, and on a combination of both, respectively. Therefore, they allow us to study whether affordances are better learned based on the tool-pose's surface information, its spatial information, or a balanced blend of both. Additionally, we compare their performance against state-of-the-art computer vision features used for tool recognition (see Section 4.2), namely, the feature vector extracted from the last convolutional layer of AlexNet Deep Convolutional Neural Network (DCNN) [82], when the robot is looking at the tool-pose in its hand.

As stated in Section 6.1.3, affordance knowledge can be implemented as the mapping between the space of tool-pose features, which represent tool and grasp, and the space of affordance vectors, which represent action and effect, that is: $\textit{affordance vector} = f(\textit{functional features})$. On the previous works, in order to ease learning, this mapping

was applied by clustering one of the elements and classifying the functional features into the discovered categories (of affordance vectors in 6.2 and of tool-poses in 6.3). In contrast, the architecture proposed in this work is capable of learning the mapping between these two spaces in a gradual manner, without the need to set categorical boundaries in any of them. This is achieved by means of a 2-step process whereby functional features and affordance vectors are mapped first onto respective Self-Organizing Maps (SOM) to achieve dimensionality reduction, and then a regressor model is trained to learn the mapping between the coordinates of the tool-pose features and those of the corresponding affordance vectors on their respective SOMs. A diagram of the proposed learning architecture can be observed in Figure 6.4.1, whose components will be detailed in Section 6.4.2.b.

These methods were evaluated on a scenario similar to the one applied in [96], but performed additionally on the real robot. While the presented methods could be applied in more complex scenarios, we chose this one as a proof of concept, because of the feasibility of the actions and the automatic effect computation.

6.4.2. Materials and Methods

6.4.2.a. Experimental setup

As stated above, all the experiments in this study were carried out using the iCub humanoid robot and its simulator, both described in detail in Chapter 3. All the tool 3D models that were used for feature extraction in the experiments were modeled using Trimble's SketchUp software [175]. 3D processing and visualization was performed with help of the Point Cloud Library [148]. Experimental data analysis, including visualization and learning, was implemented in MATLAB, employing the third party SOM_Toolbox for dimensionality reduction and data visualization [192], and the built-in Neural Network library for learning regression models from the data. In order to use the models learned in MATLAB to guide the robot actions, the available YARP bindings for MATLAB were applied.

All the code used in the present study is available at
www.github.com/robotology/affordances.

The experimental scenario was devised so that with a relatively simple repertoire of actions, different tool-poses achieved distinct effects. For that end, each data gathering

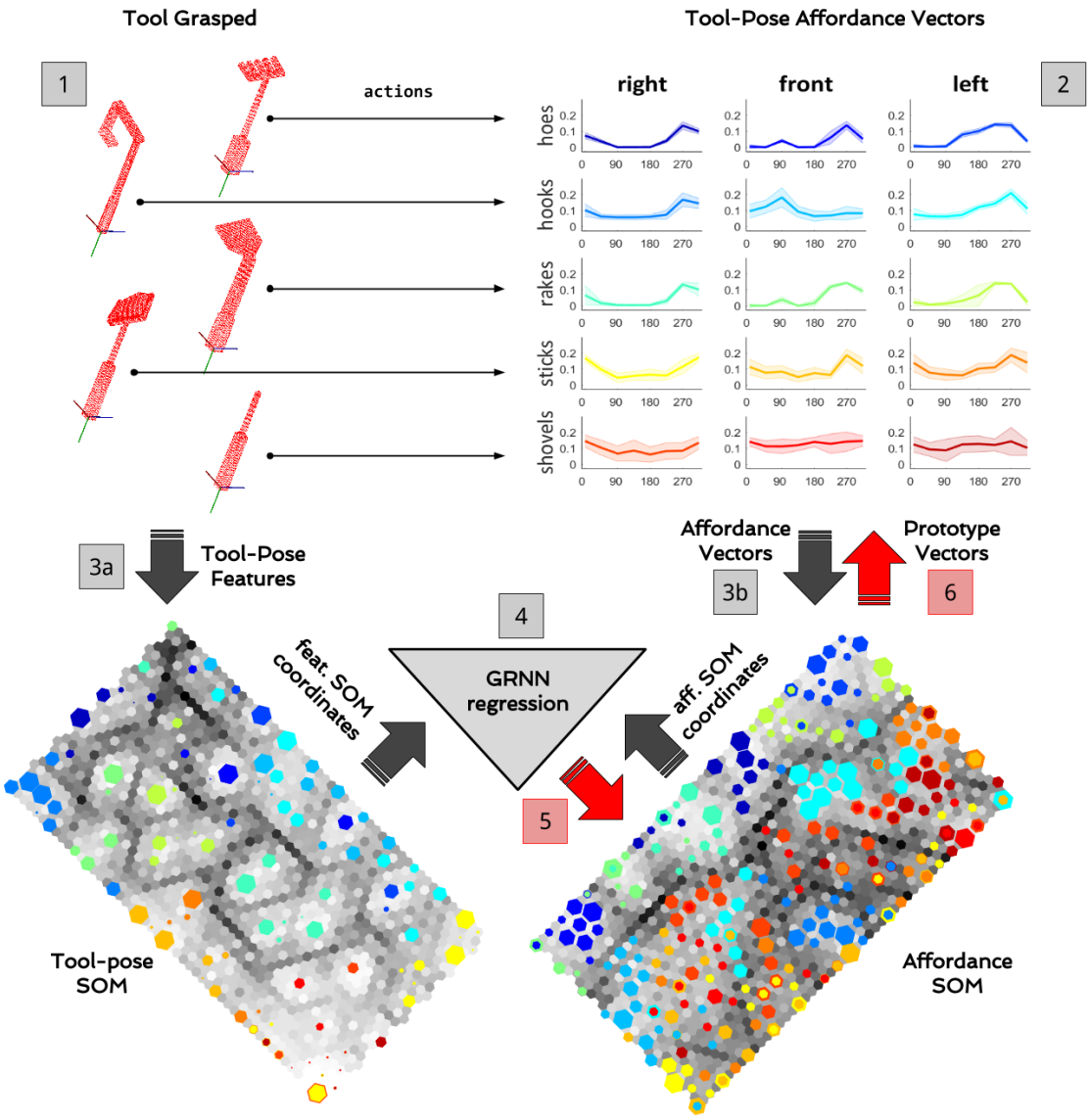
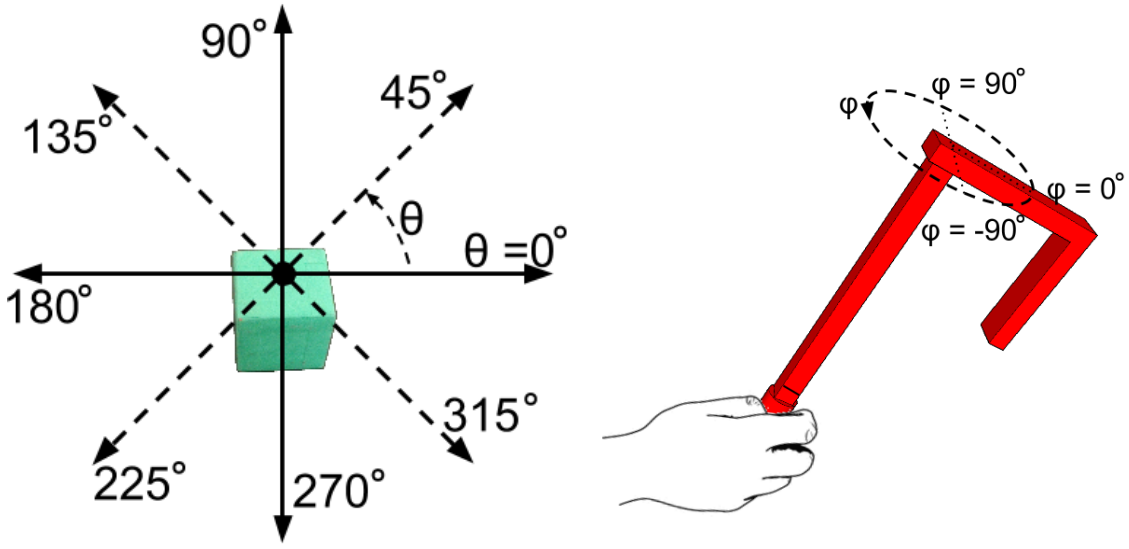


Figure 6.4.1.: Diagram of the proposed approach to discover, learn and predict tool-pose affordances. On the training phase (black arrows), a set of tool-pose features [1], and their corresponding affordance vectors [2] are available to the system from previous recording. Keeping the correspondence, tool-pose features and affordance vectors are mapped into respective SOMs for dimensionality reduction [3a and 3b]. Finally, a GRNN regressor model is trained to learn the mapping between the coordinates of the tool features in the **tool-pose SOM**, and those of the corresponding affordance vectors on the **affordance SOM** [4]. On the prediction phase (red arrows), the tool-pose SOM coordinates of the tool-pose being held are computed from the tool's features, and fed to the regressor to get an estimate of its coordinates on the affordance SOM [5]. The prototype vector of the closest neuron to the estimated coordinates is considered the predicted affordance vector for that given tool-pose [6]. For easier interpretation, each color corresponds to data generated by a particular tool type hoe, rake, etc) in a particular pose (right, front, left), assigned following the affordance vector graphs on the right of the diagram. This information was not used for training.



(a) Diagram of the drag action cycle. The tooltip is initially placed slightly behind the object, and then the tool is displaced 15 cm along the radial direction given by θ. In the real setup, only the actions displayed with full arrows are performed.

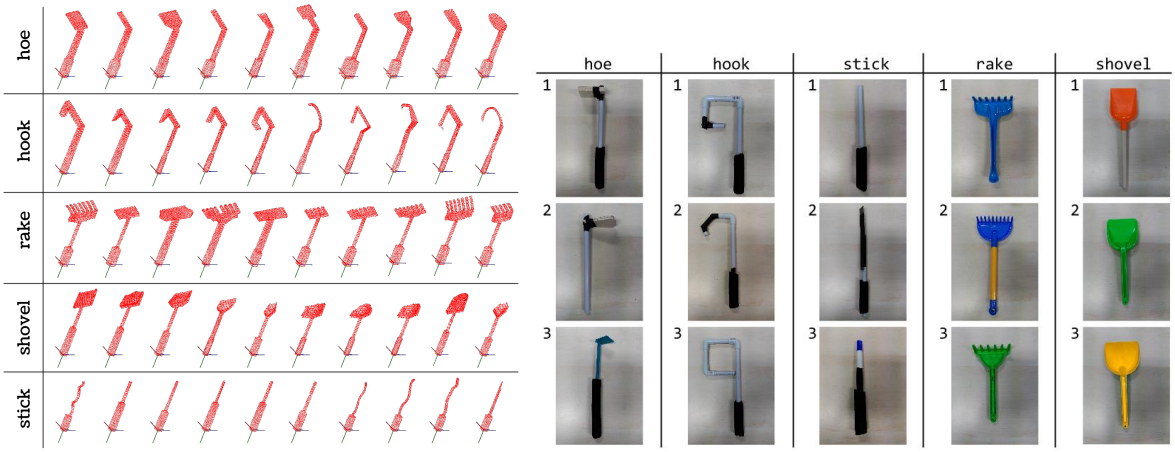
(b) Grasp parameter φ controls the rotation around the tool's handle axis. Therefore, `right` orientation corresponds to $\varphi = -90°$, `front` to $\varphi = 0°$, and `left` to $\varphi = 90°$.

Figure 6.4.2.: Parameters controlling tool-pose and interaction: (a) Action and (b) Grasp.

trial consisted in a series of 15 cm drags performed upon a small target object along directions at intervals of 90 degrees on the real robot and 45 degrees in simulation, thus 4 actions per trial in the real robot and 8 in simulation (see Figure 6.4.2a). After each action execution, its effect was computed as the Euclidean distance that the object had been displaced on the table plane, and the object relocated to its starting position. Thereby, for each performed trial, an *affordance vector* was recorded, representing the drag affordance of the tool-pose with which the trial was performed.

The target object was a small cube of 6 cm in side placed at a spot on a table randomly chosen at 40 ± 4 cm in front of the iCub and 10 ± 4 cm to its right ($x \approx -0.4$, $y \approx 0.1$, $z \approx -0.13$ in the iCub's reference frame). The object was tracked by a segmentation algorithm based on Ojala's Local Binary Pattern technique (implemented from [123]), so the specific starting position could be modified, as long as the working space in each direction allowed the robot to perform the dragging action without colliding with itself (when pulling) or going out of reach limits (when dragging away).

The described experimental trials were performed with 50 different tools in simulation



(a) Tools in simulation. **(b)** Tools in real setup.

Figure 6.4.3.: Set of tools used in this study, on the (a) simulated setup and (b) real setup. Individual names of each tool are formed by just adding the tool index after its type, eg `hook2`, `hoe3`, etc.

and 15 in the real setup, divided in 5 categories for visualization and clarity: rake, hoe, hook, shovel, stick (displayed in Figure 6.4.3). CAD models of all the tools were created using Sketchup, and transformed to the pointcloud format using PCL. For the experiments, each tool was grasped by the robot in 3 possible orientations: `right`, `front`, `left`, as described in Figure 6.4.2b. Thus, experiments were carried out with 150 distinct tool-poses in simulation and 45 on the real robot. An image of the setup with the iCub performing the drag action with some example tool-poses can be observed in Figure 6.4.4.

For each tool-pose grasped by the robot, its oriented pointcloud model was obtained from the available canonical model applying the methods described in Section 4.5, and subsequently used for two purposes. On the one hand, it was sent to the processing modules in order to extract the OMS-EGI features in its three variants, BALAN, OCCUP and EGI (detailed in Section 5.3). On the other, it was used to determine the position of the tooltip with respect to the robot hand reference frame, required to extend the kinematics of the robot to incorporate the tip of the tool as the new end-effector for further action execution, as described in Chapter 4.

6.4.2.b. Parallel mapping from tool-pose features to affordances

When considering tools whose affordances depend solely on their geometry, it can be assumed that in general, tools with similar geometry will offer similar affordances.

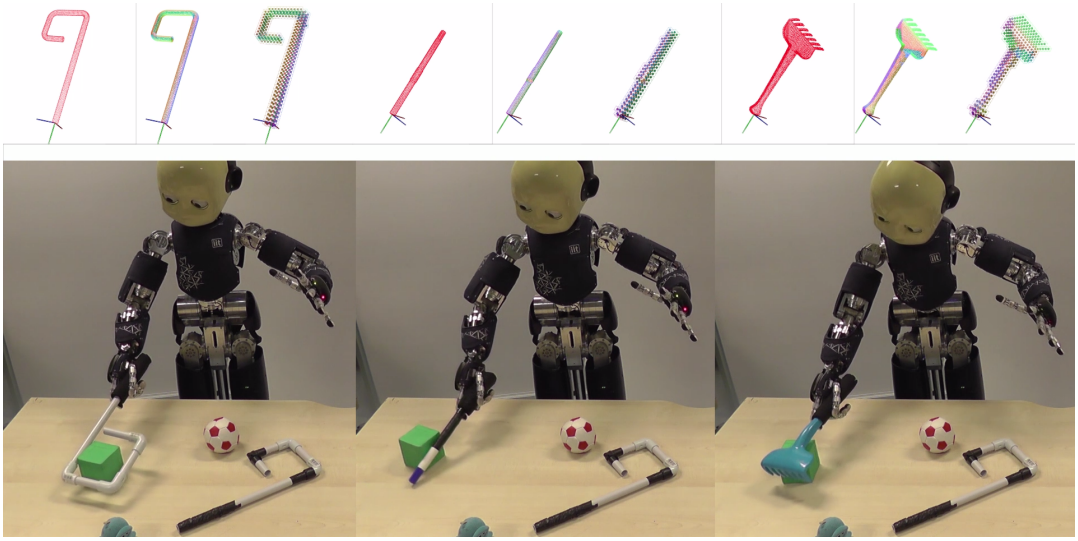


Figure 6.4.4.: Example of the Experimental setup. On each image, the iCub performs a drag action with a given tool-pose, namely HOE3-left, STICK3-front and RAKE2-front. The images on top of the iCub pictures illustrates the process of OMS-EGI acquisition, where the leftmost image in each triplet is the oriented pointcloud mode, the one in the center shows its normals, and the rightmost the voxel-wise normal histograms, represented as in Section 5.3.

That is why most affordance studies only consider a limited number of possible outcomes of robot actions, either by performing automatic clustering of the perceived effect ([158, 180, 141, 97]), or by predefining effect categories to which the observed effects are assigned [113, 58]. Similarly, it is a common practice in studies where objects or tools are represented by features to cluster them before further processing [124, 96]. However, it is frequently the case that these discretization steps are imposed on a data space (of measured effects, or object features) which is relatively homogeneously distributed, often leading to thresholds or boundaries separating similar data points. Moreover, the within-cluster differences that may be present in these measurements or features are subsumed into the cluster label and ignored when learning the objects or tools affordances.

In the present study, tool affordances are represented by the mapping between tool-pose features $X \in \mathbb{R}^{S_J}$, which describe the tool and grasp pose applied, and affordance vectors $Y \in \mathbb{R}^K$, which determine effect in function of the action for any given tool-pose. S_J is the length of the OMS-EGI feature vector for variant J; OCCUP, EGI, or BALAN and K the number of different directions of the drag action considered, which determines the length of the affordance vector (8 in simulation, 4 on the real robot). The proposed architecture enables learning the mapping between these two spaces, $f : \mathbb{R}^{S_J} \rightarrow \mathbb{R}^K$, without the need to set categorical boundaries in any of them.

To that end, both spaces were mapped first onto respective 2-dimensional Self - Organized Maps, referred henceforth as *tool-pose SOM* and *affordance SOM*. Then, a regressor model was trained to learn the mapping from the coordinates of the tool-pose features on the tool-pose SOM to the coordinates of the corresponding affordance vectors on the affordance SOM. Applying unsupervised dimensionality reduction on the original spaces limits the complexity of the subsequent supervised regression problem, by reducing the original problem of finding $f : \mathbb{R}^{S_J} \rightarrow \mathbb{R}^K$ to $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, being $K > 2$ and $S_J >> 2$.

SOMs were chosen over other dimensionality reduction methods because they allow to map new data points incrementally (unlike, for example, the popular t-SNE [188]), which is required for predicting the affordances of tool-poses not seen in training, and they maintain to a large extent the topology of the data in the original high-dimensional space [81]. Moreover, each neuron in a SOM has an associated prototype vector which, after training, approximates the values of the input vectors mapped to that neuron and can thus be used for inverse mapping. Prototype vectors are analogous to cluster centroids used in our previous study [97], but unlike them, which are by definition distinct from each other, prototype vectors can vary very gradually from neuron to neuron, given an enough amount of neurons to represent the original space. Therefore, they provide a fine grained representation of the original space, and avoid the need to predefine any number of clusters into which to divide it. Similar techniques involving 2 parallel SOMs have been used in [157] for multimodal object learning, and [141, 142] for object affordance learning, but applying very different data modalities, and learning and prediction methods.

In the present scenario, applying first a dimensionality reduction step based on SOMs has further advantages: Primarily, whereas training the regressor directly would have required matching pairs of tool-pose features and affordance vectors, SOMs are trained separately with data from their respective spaces, i.e., tool-pose features and affordance vectors. While the affordance SOM has to be mapped with the data gathered from observing the effects of the experiments performed, the tool-pose features can be generated by extracting them from slight rotations of the oriented pointcloud models recorded in the experiment. Hence, we applied this "data augmentation" technique to generate more tool-pose feature samples than the number for which we have corresponding affordance vectors, and use this extended dataset to train the tool-pose SOM. Another important advantage is that the prototype vectors of the affordance SOM provide a mechanism to retrieve predictions in the original affordance vector space from the lower dimensionality

regression results.

After dimensionality reduction, the second step of the proposed tool affordance learning method consisted in learning a regression model between the low dimensional representation of the tool-pose features and of their corresponding affordance vectors. The regressor was implemented using Generalized Regression Neural Networks (GRNN), a modification of radial basis networks which is able to approximate arbitrary functions and avoid local minima in 1-pass training [160]. Support Vector Regression methods weer also tested, but they yielded very similar results while taking longer to train, so GRNN was selected for further processing. These networks depend on the regularization parameter σ, which controls the spread of the radial basis functions. The best value of σ for each model was found by performing recursive line search[1]. In order to train the GRNN model, first we computed the best matching units (BMUs) of all the train tool-pose features and affordance vectors on their corresponding SOMs, and obtained their coordinates. We refer to the set of coordinates of the BMUs corresponding to the tool-pose features as $X_{SOM}, \in \mathbb{R}^2$, and to the set of those corresponding to the affordance vectors as $Y_{SOM}, \in \mathbb{R}^2$. Finally, the GRNN model was trained by feeding X_{SOM} as input and Y_{SOM} as target, so the desired mapping function $f(X_{SOM}) \rightarrow Y_{SOM}$ was learned.

As mentioned above, one advantage of the proposed method is that the SOM prototype vectors can be applied to yield predictions in the original space of affordance vectors. In order to achieve this, the first step was to extract the tool-pose feature vector $x \in X$ that represents the tool-pose from which we want to predict the affordance vector. The obtained feature vector x was then mapped to the trained tool-pose SOM, and the coordinates of its BMU, $x_{SOM} \in X_{SOM}$, computed. These coordinates were subsequently fed to the trained GRNN model, which in turn predicted the coordinates $\hat{y}_{SOM}$ on the affordance SOM. Finally, the predicted affordance vector $\hat{y}$ was given by the prototype vector of the closest neuron to the predicted coordinates $\hat{y}_{SOM}$.

6.4.2.c. Prediction based action selection

The methods described above enable the robot to predict the affordance vector of any radial tool in any pose, i.e., the expected effect for any of the possible actions. Therefore,

[1]Recursive linesearch was conducted by evaluating the accuracy of the regressor at equally spaced values of σ with 5-fold cross validation, and iteratively exploring values at smaller distances centered around the value with the best accuracy on the previous iteration, until the accuracy improvement among consecutive iterations was under a certain threshold.

if the predictions are accurate, this knowledge would allow the robot to select the action with the best expected effect for any desired task achievable through its action repertoire. In order to evaluate the extent to which the proposed methods can lead to successful action selection for a certain task, we devised a secondary test experiment, which we refer to as the Action Selection experiment.

In this experiment, the robot applied the affordance knowledge learned by means of the methods described above in order to predict the affordance vector of a tool given to it by the user, and used this prediction to select the best action for the task of maximally displacing the target object. In order to provide a good assessment of the generalization capabilities of the proposed learning architecture, this test was performed with affordance models that had not been trained with the tool being used, in any of its poses.

Hence, on each test trial, the predicted affordance vector for the considered tool-pose was computed as described in Section 6.4.2.b. Based on this vector, the action with the maximum predicted effect was selected, and its parameters sent to the robot to be performed. After the action execution, the actual achieved effect was measured in order to evaluate the success of the given task.

6.4.3. Results

6.4.3.a. Experimental data collection and separation

Experiments in the present study were carried out in simulation as well as on the real iCub Humanoid robot. In simulation, the tool set consisted of 50 tools, while in the real setup, 15 tools were used instead (see Figure 6.4.3). Each tool was grasped by the robot in 3 different orientations, `right`, `front` and `left`. Therefore, the total amount of considered tool-poses was of $3 \times 15 = 45$ on the real robot and $3 \times 50 = 150$ in simulation. In the simulated setup, the orientation of the grasp was achieved by setting the orientation parameter φ to 3 possible values: $-90°, 0°$ and $90°$, respectively. On the real robot, the tool was handed to the robot by the experimenter with the effector pointing in the desired orientation, and the real parameter φ was estimated by matching the tool's 3D model to a partial reconstruction of the tool obtained with the robot's stereo-vision, as described in Section 4.5.

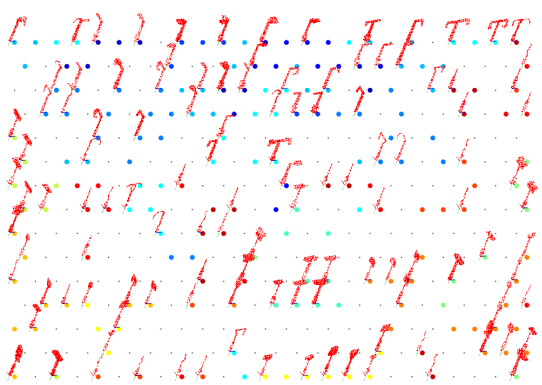
Interaction data was gathered by performing 4 experimental trials with each tool-

pose, where each trial consisted in the execution of drag actions in 4 directions in the real robot and 8 directions in simulation, performed as described in Figure 6.4.2a. As a result, 720 actions corresponding to 180 trials were performed on the real setup, and 4800 actions corresponding to 600 trials in simulation. On each of these trials, the recorded data consisted of the affordance vector representing the recorded effects of the actions performed in that trial, and the OMS-EGI feature vectors used to describe the tool-pose with which the actions were performed: EGI, BALAN, OCCUP. On the real setup, the features extracted from the deep neural network when the robot was looking as the tool (as described in Section 4.2) were also recorded. Moreover, for each tool-pose used to perform the experiment, 30 extra OMS-EGI feature vectors of each OMS-EGI variant were gathered by means of the data augmentation method described in 6.4.2.b. This number was selected in order to have a number of OMS-EGI samples considerably larger than its dimension S_J to perform the unsupervised training of the tool-pose SOM. Meanwhile, the deep learned features were extracted from around 25 different observations of each tool-pose, to have a similar number of samples.

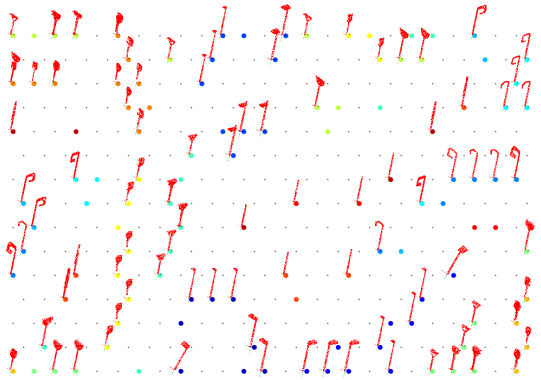
On every evaluation scenario (simulation or real robot setup, with EGI, BALAN or OCCUP feature set), the data gathered was divided into training and testing sets to evaluate the presented methods. In order to provide a more complete assessment of their performance, we applied two different data separation schemes. The first separation scheme served to evaluate the general predictive performance of the proposed method, and was achieved applying 4-fold cross validation, that is, iteratively selecting at random the data corresponding to 1/4 of the trials for testing, and keeping the rest for training. We refer to this separation scheme as RAND. The second separation scheme assessed the capability of the proposed methods to generalize the learned affordances to previously unseen tools. For that end, we performed tool-wise 1-out separation where on each run, the data from all the trials corresponding to a given tool (in all its poses) were used for testing, and the data from the rest of the tools used for training. This scheme is referred to as 1OUT.

6.4.3.b. SOM-based unsupervised dimensionality reduction

As described in Section 6.4.2.b, the first step in the proposed method for affordance learning is to map the spaces of tool-pose features and affordance vectors onto corresponding SOMs. In the current study, both SOMs were chosen to have a hexagonal lattice of 15×20 units, which provided a good compromise between representation resolution and training time required. The tool-pose SOM was trained using all the OMS-EGI

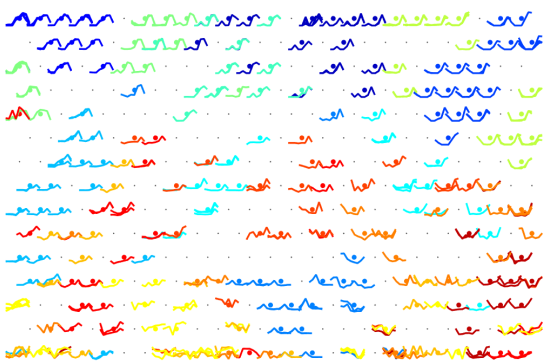


(a) Trained **tool-pose SOM**, from **simulation** data.

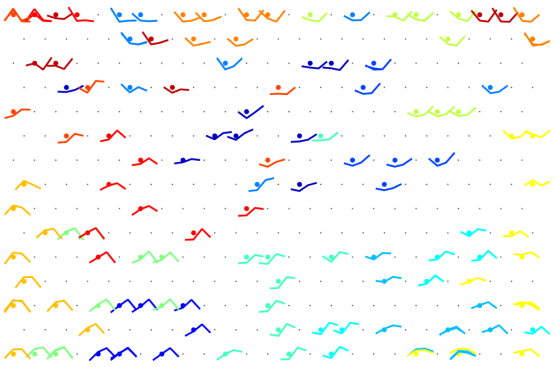


(b) Trained **tool-pose SOM**, from **real setup** data.

Figure 6.4.5.: Trained tool-pose SOM topologies from features of the tool-poses used in simulation (a) and real setup (b). Tool-pose models are displayed onto the BMUs of their corresponding feature vectors.



(a) Trained **Affordance SOM**, from **simulation** data.



(b) Trained **Affordance SOM**, from **real setup** data.

Figure 6.4.6.: Trained affordance SOM topologies from data gathered in simulation (a) and on the real setup (b). In order to understand better how the high-dimensional data is mapped onto the SOMs, sample affordance vectors from the training set are plotted onto their corresponding BMUs. Colors, as in Figure 6.4.1, represent the tool type and pose that generated the data, but this information was not used for training.

vectors recording during data gathering that belonged to the train set, as well as the ones provided by the data augmentation technique. The results of this mapping process can be observed on Figure 6.4.5. The affordance SOM, on the other hand, was trained with affordance vectors from the training set, all of which had corresponding tool-pose vectors. Results are displayed in Figure 6.4.6.

6.4.3.c. Prediction of tool-pose affordances

The performance of the proposed method was evaluated by comparing the affordance vectors predicted for the test set of tool-poses, $\hat{Y}$, with the affordance vectors previously recorded for those tool-poses, Y, for all the evaluation scenarios (different setups, data separation schemes, and OMS-EGI parameter settings). In each case, a baseline performance was also computed in order to compare the prediction results achieved by the trained system against the results obtained in the absence of learning. The baseline was defined as the prediction performance achieved when the prediction models were trained with data in which the correspondence between tool-poses and affordance vectors was broken, which was achieved by shuffling the indices of X_{SOM} and Y_{SOM} before being fed to the GRNN for training.

Prediction performance was measured in terms of the Mean Absolute Error (MAE), which represents the average absolute distance between the predicted affordance vectors $\hat{Y}$ and the recorded ones Y. Learning performance was assessed by means of the percentage of improvement (PI), which indicates how much better the trained system performs when compared to the baseline one, so that if nothing was learned, hence error was not improved, PI would be 0% while if error was reduced to 0, $PI = 100\%$.

Formally:

$$MAE = \frac{1}{N} \sum^{N} abs(Y - \hat{Y}) \tag{6.14}$$

where N is the number of test trials, and

$$PI = \frac{MAE_{BL} - MAE}{MAE_{BL}} \tag{6.15}$$

Table 6.4.1 displays the prediction error in each of the evaluation scenarios with the proposed OMS-EGI settings, expressed in terms of the MAE computed as the average from 50 runs in RAND data separation mode and as many runs as tools were considered

SIM		**BALAN**			**EGI**			**OCCUP**	
	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$
RAND	2.77	5.00	44.6	2.85	4.99	42.9	2.66	5.02	47.0
1OUT	3.51	5.12	31.4	3.28	5.09	35.6	3.26	5.09	35.9
REAL		**BALAN**			**EGI**			**OCCUP**	
	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$
RAND	2.44	5.71	57.2	2.36	5.72	58.7	2.27	5.73	60.4
1OUT	3.78	5.67	34.7	3.91	5.63	30.5	4.86	5.89	17.0

Table 6.4.1.: Mean Absolute Error (MAE, in cm), Baseline (MAE_{BL}, in cm), and Percentage of Improvement (PI, in %) average for each evaluation scenario.

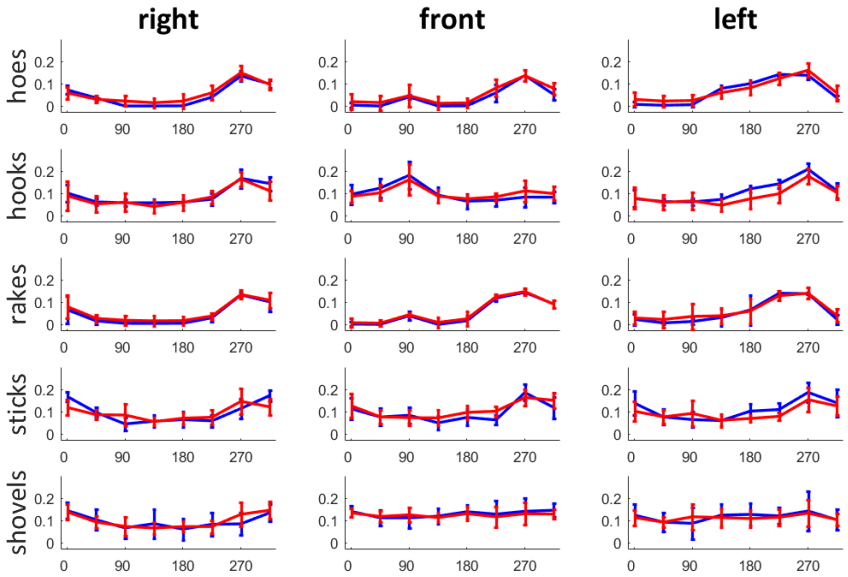
REAL	**CNN features**		
	MAE	MAE_{BL}	$PI(\%)$
RAND	3.18	5.71	45.6
1OUT	5.53	5.76	4.01

Table 6.4.2.: Prediction results obtained with AlexNet's FP7 features.

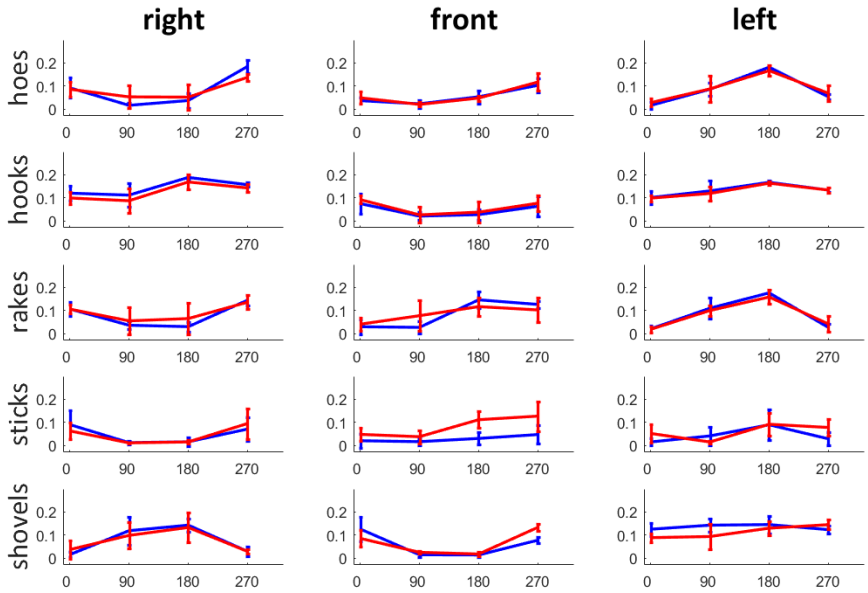
in the 1OUT data separation mode (15 in the real setup and 50 in simulation). For comparison, Table 6.4.2 displays the results obtained when, instead of the proposed 3D features, the affordance models were trained and tested with the deep learned features extracted from layer FP7 of the off-the-shelf AlexNet CNN used to recognize the tools (only applicable in the real setup, see Section 4.2). In Figure 6.4.7 the comparison between the recorded affordance vectors and those predicted by the model trained with 1OUT data separation scheme using BALAN features (which was chosen for the Action Selection experiment) can be observed graphically.

6.4.3.d. Action selection

The last evaluation step consisted in applying the learned models to select, given a tool-pose, the best action for the task of achieving maximum displacement of the target object, as explained in Section 6.4.2.c. Therefore, the actions executed by the robot depend on which of the learned models we apply. In order to ensure fair evaluation, this test was carried out using always models trained with the 1OUT data separation scheme, so that the data corresponding to the tested tool had never been used to train

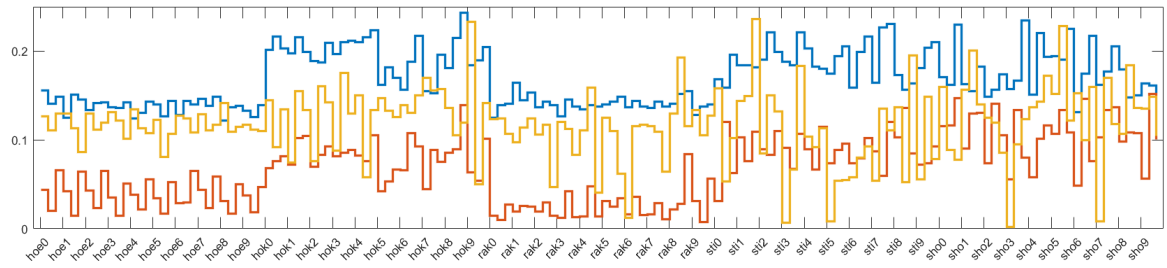


(a) Simulation

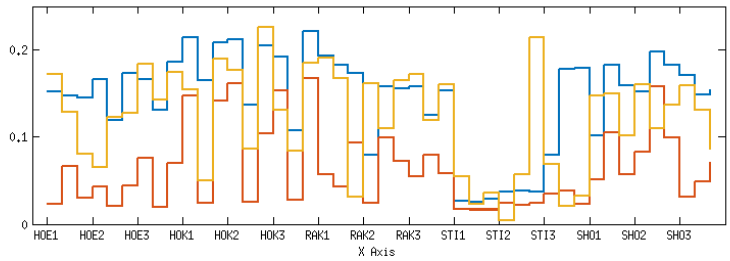


(b) Real

Figure 6.4.7.: Predicted effect (red) against recorded effect (blue) with variance (vertical lines), for (a) Simulation and (b) Real setup, using the 1-OUT data separation scheme and BALAN parameter settings. Each row of subplots corresponds to the aggregated data of all tools in a tool category (hoe, rake, etc), separated by pose (on columns). In all graphs, X axis corresponds to the angle of the drag action, from $\theta = 0°$ to $315°$, and the Y axis to the displacement (predicted or measured) in meters.



(a) Action selection experiment results, in simulation.



(b) Action selection experiment results, in the real setup.

Figure 6.4.8.: Results of the Action Selection experiment by tool-pose (a) simulation and (b) real setup. The average of the 2 effects measured from execution of best expected action for each test tool-pose (orange) is displayed against the baseline for that tool-pose (red), and its maximum effect achieved during the data gathering phase (dark-blue). All tool-poses are considered, but only tool names are added on the X axis in order to prevent clutter.

the models used to predict its affordances. Concerning the OMS-EGI feature parameter setting used to trained the models, it can be observed in Table 6.4.1 that in simulation the models learned using OCCUP parameter settings perform better than the rest for the 1OUT data separation scheme. However, in the real setup the performance of the models trained with OCCUP decreases considerably, when compared with the performance of the models trained with the other tool-pose feature parameter settings. Therefore, we chose to perform the action selection test applying the models trained with BALAN parameter settings, because, although they perform slightly worse than OCCUP on simulation, the PI they achieve on the real setup is two times higher. For each test tool-pose, thus, the robot obtained a prediction of its affordance vector from the model trained using 1OUT and BALAN settings, and executed the action with maximum expected displacement, as described in Section 6.4.2.c. This procedure was run twice for each tool-pose, yielding the results that can be observed in Figure 6.4.8.

Based on these results, the degree of accomplishment of the given task was assessed by comparing the achieved effects against an action selection baseline, which was defined for

SIM	hoes	hooks	rakes	sticks	shovels	**Total**
$S(\%)$	100	90	96.7	50	66.7	80.67
G	100	63.3	86.7	0	40	56.7
REAL	hoes	hooks	rakes	sticks	shovels	**Total**
$S(\%)$	100	88.9	88.9	77.8	88.9	88.9
G	100	77.8	77.8	44.4	77.8	75.6

Table 6.4.3.: Action selection performance results, in simulation and the real setup.

each tool-pose as the median effect among all effects recorded for that tool-pose during the data gathering part of the experiment. In principle, if actions had been selected at random, the achieved effect would be over this baseline 50% of the times. Figure 6.4.8 shows the displacement that the robot achieved using the best action for each of the tested tool-poses, alongside the corresponding baseline and the maximum effect of any actions observed during the data gathering phase. The overall degree of task success was measured in two ways similar to [58], in order to allow for comparison. On the one hand, we measure the Success Rate S as the percentage of times that the effect achieved by the selected action was higher than the baseline. On the other, we measured the Gambling Score G, which is computed by subtracting the number of unsuccessful trials (UT) times the number of possible unsuccessful options (UO) (in this case one, effect below the baseline) to the number of successful trials (ST), divided by the total number of trials (T), so that a random action selector would lead to $G = 0\%$, and a perfect one to $G = 100\%$, that is:

$$G = (ST - (UT \cdot UO))/T \tag{6.16}$$

The results for simulation and real setup, separated by tool type, can be observed in Table 6.4.3.

6.4.4. Discussion

In the current study we presented a set of methods to allow a robot to learn tool affordances through interaction with its environment, considering also the pose in which the tools are grasped. These methods were tested on the iCub robot by means of comparing recorded effects of tool use with predicted ones, as well as using the prediction to select actions with previously unseen tools. The results show that the proposed methods enable

the iCub to learn and accurately predict tool affordances based on their geometry, and hence successfully use this knowledge to select valid actions to achieve a displacement task on the target object.

The prediction results, displayed in Figure 6.4.7 and Table 6.4.1, evince that even in the presence of unknown tools, the proposed methods are able to successfully generalize the knowledge learned for similar tools, and apply it to correctly predict the effect the tool will generate for any considered action. This prediction, however, is heavily influenced by the type of features used to describe the tool-poses. Thus, in this study we compared the performance of 3 different implementations of the OMS-EGI descriptor; EGI, BALAN and OCCUP, described in Section 5.3, as well as that obtained with features from a pre-trained CNN.

Interestingly, the OCCUP setting yields the best results (although by a small amount) in simulation, while in the real setup, this feature setting led to considerably lower performance on the generalization scenario (with the 1OUT data separation scheme). This can be explained by the fact that the small variations performed in the tool-poses in order to obtain the data augmentation samples do not modify in essence the occupancy of these oriented pointclouds within their bounding box, at least at the resolution that the actual OCCUP parameters are able to measure. Therefore, all OCCUP samples from each tool-pose get mapped to the one or at most a few BMUs. On the real setup, where the number of tool-poses is small, this concentration of training samples onto a few prototypes implies that the SOM is not able to provide a gradual representation of the tool-pose features, and therefore might not represent well new tools that had not been used to train this map, leading to poor predicting performance. On simulation, although data augmentation samples from OCCUP settings still tend to coalesce in one or few BMUs per tool-pose, the much larger number of tool-poses considered makes the concentration effect less relevant, as a gradual variation of tool-pose representation along the SOM is provided nevertheless.

With BALAN parameters (as well as EGI), on the other hand, small rotations do vary the surface normals considerably, and thus the resulting feature vector. Thus, activations from different samples of the same tool-pose tend to be more spread on the SOM, which in turn results in a more gradual representation of tool-pose features, more capable to accommodate features from new tool-poses.

In general, the better performance of OCCUP features (when enough data is provided to produce the smooth representation) could be explained because in an interaction

scenario such as the present one, the occupancy grid that OCCUP features represent provides coarse spatial information of the tool's extension and position with respect to the hand, which is more relevant to explain its use than the surface information provided by the EGI and BALAN features.

The results obtained with the off-the-shelf deep learned features show that in the RAND scenario, the performance of the off-the-shelf deep learned features is almost as good as that obtained with the proposed 3D features. In this scenario, the prediction problem is akin to a discrimination task, where similar instances of previously seen classes (tool-poses) have to be associated with their corresponding outputs (affordance vectors). However, when the task consists in predicting the output for instances of classes that had not previously been seen by the system, such as in the 1OUT scenario, the performance of deep learned features dropped to almost chance levels. These results suggest that tool representations based on their pose and 3D geometry correlate much better with the affordances that those tools can offer than those based on invariant 2D properties. In general, this indicates that despite the unquestionable performance of deep learned features in traditional computer vision scenarios such as detection and classification, other kind of features such as the proposed ones can be more suited in interactive scenarios like the present one where the physical properties and position of the object matter.

On the results from the action selection phase of the experiment, displayed in Figure 6.4.8 and Table 6.4.3, we can observe that the predictions yielded by the proposed method enabled the iCub to select the desired action for a given task with a high degree of success. Yet, in some cases the estimated affordance differs from the real one, and the selected action does not produce a large displacement of the object as expected. By careful examination of the recorded affordance vectors for each tool-pose, we observed that the trials where the selected action failed to achieve maximum effect usually corresponded with situations in which a given tool-pose, or tool-poses with similar geometries, did not offer consistent affordances, that is, achieved different effects for the same action. This effect was more pronounced on simulation, where sometimes small contacts between the tool and the target object generated unpredictable effects. In particular, we observed that due to errors in collision calculations, when a tool pushed the target object down against the table there were some cases where the object would "jump" a few centimeters away in an unpredictable direction. This situation happened with tool-poses where the effector was situated in the same vertical axis as the handle, namely sticks, hooks oriented to the front, and some shovels oriented to either side, and

explains the poor results obtained with these tool-poses.

This effect is also reflected in the high variance for recorded data corresponding to sticks and shovels that can be observed in Figure 6.4.7, and in how spread the BMUs corresponding to stick affordance vectors are on the SOM representations (yellow vectors on Figure 6.4.6).

On the other hand, for those tool-poses which offered consistent affordances, such as hoes, rakes, hooks oriented to the side and most shovels in simulation, and all tools on the real setup, the selected actions led to successful effects with a high degree of accuracy (up to 100% in the case of rakes). These results indicate that the proposed methods were able to accurately predict the tool-pose affordances from their geometry, and apply this knowledge to select suitable actions to achieve the given task, even with previously unseen tools.

In order to assess the achieved results in the context of the state-of-the-art, we compared our results with the ones by Gonçalvez et al. [58], as it is the only study, to the best of our knowledge, performed in a similar setup and with comparable actions and effects. In our study, the Gambling score G in simulation gets seriously penalized by the inaccuracies on the prediction of the sticks affordances, which indeed perform as if predicted at random, and therefore is on average lower than on their study. On the other hand, the overall accuracy is nevertheless around 6.5% higher in our study. On the real setup, however, it can be observed that our method provides a substantially higher score, with over a 10% increase in both measurements. An important factor to take into account, moreover, is that in our study we consider 8 possible directions and 150 different tool-poses in simulation and 45 tool-poses in the real robot, while in [58], only 4 tools and 4 push directions are considered in simulation, and 1 tool and 1 action on the real robot.

However, our focus on the tools came at the expense of simplicity in the representation of all the other elements present in the interaction which influence the affordance. Namely, the action repertoire, as well as target object and effect representations have been kept purposefully simple in order to limit the search space of all possible combinations, which otherwise would render its exploration unfeasible in a reasonable amount of time.

Concerning the target object, we have only considered its location, disregarding any properties such as geometry or material. We acknowledge that these properties do influence the effect of actions on the object, but as in other studies in the previous

literature, such as [174], we assume that it can or has been learned in previous stages of the robot development.

We also acknowledge that the action repertoire and possible grasp orientations, as well as the way in which the effect is measured, are quite limited. While the presented learning methods could cope with higher dimensionality in inputs and outputs, increasing the complexity of these elements, specially the action repertoire, would easily lead to search spaces impossible to explore sufficiently on a robotic setup, even in simulation, unless other constraints are in place.

Yet, we recognize that the available actions, as well as the representation of the effect, directly determine the kind of affordances that can be discovered, and therefore, learned. For example, the applied effect representation is unable to measure, and thus identify, actions such as hammering, pouring, or cutting. Figuring out a representation that could encompass all these possibilities, and moreover, be automatically computed, is a complex task out of the scope of this study, but nevertheless worth tackling in the future.

Another limitation of our approach is the current lack of support for tool selection, given that every trial starts with the tool already being held by the robot. However, if combined with some method for tool recognition, such as the one presented in Section 4.2, the present architecture could be applied as a forward model, which would predict the affordances of a number of "common" grasps of the available tools, and use the best prediction to select tool *and* pose. Of course, most effects can be achieved by many different tool-poses, so extra constraints and evaluations would have to be put in place to limit the selection process.

7. Integration

7.1. Integration

Most of the work carried out during the development of this Thesis was framed within the research pursued for the `Xperience` and `Poeticon++` projects, whose objectives were, among others:

- **Xperience:**
  - Implement, adapt, and extend a complete robot system for automating introspective, predictive, and interactive understanding of actions and dynamic situations.
  - Evaluate, and benchmark this approach on existing state-of-the-art humanoid robots, integrating the different components into a complete system.
- **POETICON++:**
  - Develop a number of basic technologies for cognitive artificial agents, which enable them to generalise their behaviours and cope with uncertainty and unexpected situations in real life environments, based on:
    * A self-exploration model that integrates motoric skills, multisensory perception skills (visual and tactile) and verbal labelling of self-acquired sensorimotor experiences for artificial agents.
    * Improved grasping skills for a humanoid via learning and affordances.

Thus, the presented methods were not only employed for the (sufficient on the other hand) scientific endeavour of proposing and testing novel methods for tool affordance learning, but also had to be applicable for "real life" demos required by these projects.

In fact, both projects had been running before the beginning of this Thesis, and a

preliminary demo was already available for the Xperience project, on top of which our contribution was built [128]. In particular, the goal of the iCub in that demo was to clean a table by removing all the objects on it, placing them in a bucket located alongside the table.

Objects could be in one of three possible ranges: reachable, which would be directly grasped and put into the bucket, reachable by tool, on which the iCub would grasp a tool and use it to pull them into the reachable range, and out of range, in which case the robot would ask a human to help. However, the only tool available was a rake located at a known position on a rack which would be grasped in a predefined configuration.

That work was mostly focused on presenting a novel approach to enhance components reusability by extending their functionalities using plug-ins at the level of the connection points (ports) [126]. Plug-in platforms, in general, extend a core system with new features implemented as components that are plugged into the core at run time and integrate seamlessly with it. When an application supports plug-ins, it enables customization, thus, provides a promising approach for building software systems which are extensible and customizable to the particular needs. Indeed, the demo was completely built using modules from the iCub software repository.

On our approach, we focus instead on the tool affordance aspect of the demo. To that end, we devised an analogous table cleaning scenario, but in this case, depending on the position of the object, several actions can be applied. Moreover, the decision of whether to perform one action or another is determined by the affordance of the tool-pose that is given to the robot. A graphical representation of this scenario as well as an execution example, can be observed in Figure 7.1.

The implementation of this demo has taken advantage of the evolution of the methods described in Chapters 4 and 6, which enabled increasing complexity and robustness. Initially, the demo relied on the techniques presented by Tikhanoff et al. in [174] for tool use learning and end-effector extension.

In our first approach [97], described in Section 6.2, a set of 2D functional features was introduced to represent tools based on their shape. Moreover, the grasp configurations were also represented in the descriptor, accounting for the fact that a tool grasped in different ways can offer very different affordances. Later, as outlined in Section 6.3, 2D features were substituted by more robust 3D features, which were used to categorize tools based on their geometry and pose, and predict the tool's affordance based on these categories [96]. On the last iteration, described in Section 6.4, the architecture was

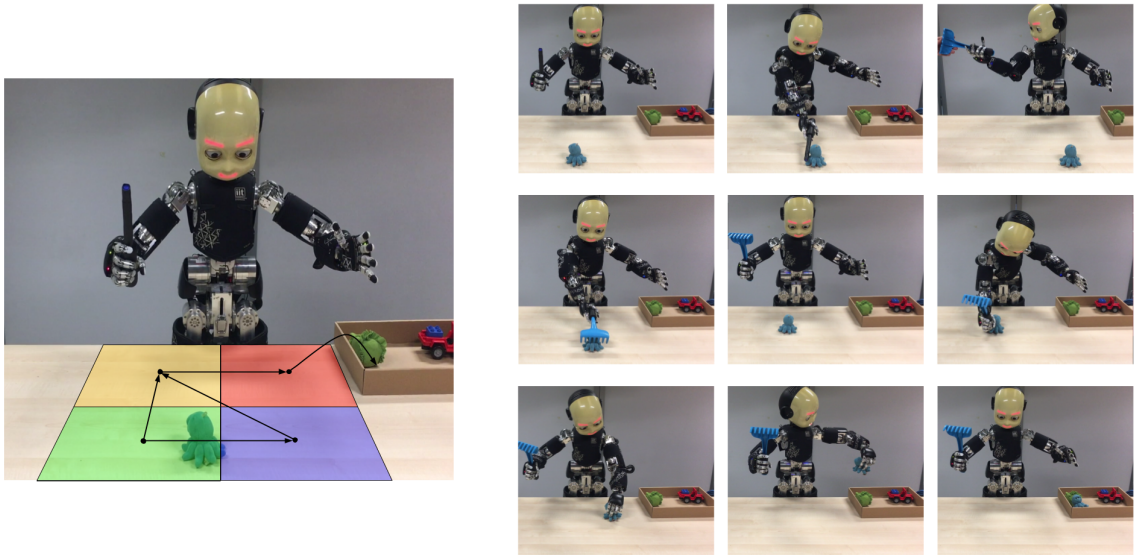


Figure 7.1.: Table Cleaning scenario. On the left side, the different areas of the workspace are indicated by color, and the arrows represent the possible actions in each area. The green and blue areas are where the object can be reached with a tool but not with the hand, so the available actions are aimed at bringing the object closer. If the object is in the yellow area, it will be pushed towards the red one, on which it will be picked up and put in the box. On the right side, a sequence of snapshots of the execution of the demo can be observed. Initially the iCub is given a stick, so it is incapable of pulling the object. Thus, it executes other possible action, slide left, and asks for another tool. The rake given then by the experimenter affords pulling, so the iCub pulls the object closer to himself, drags it to the reachable area, and finally picks it and places it in the box.

modified so that tool-poses and affordances were represented in a gradual way, which allowed to increase the accuracy of the prediction. Moreover, importantly, the tool incorporation methods described in Chapter 4 were developed too, which enabled the iCub to automatically estimate the pose and tooltip of any tool in its hand. This set of incremental steps can be observed graphically on Figure 7.2. The tool dataset available for the demo corresponded thus for the tools on which the iCub had previously trained an affordance model. These affordance models provide a prediction of the effect that will be accomplished for each of the actions in the repertoire, depending on which tool has been grasped and its orientation in the hand, which are therefore used to choose the most advantageous action.

These methods enable the iCub to achieve the goal of cleaning the table as the original demo, but they extend it by allowing a larger set of tools to be used (potentially any radial tool), as well as being adaptive to the grasp in which each tool is held, rather than allowing a single tool in a single grasp as the original demo. Specifically, the behaviour

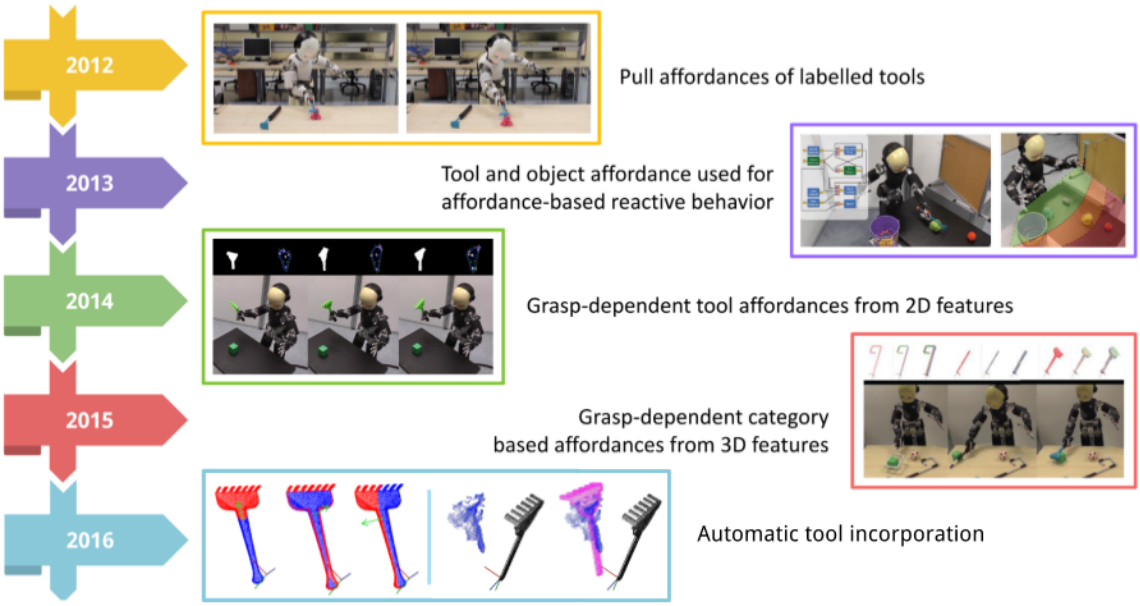


Figure 7.2.: Story of affordances: This diagram displays chronologically in a graphical way the main contributions of the late work carried out at IIT on affordances, from simple object affordance learning to the generalizing tool affordance methods presented along this Thesis.

based system allows the iCub to detect the position of objects in the table and determine the required task to remove them, in a continuous and reactive fashion, while the tool affordance learning architecture enables it to find out the pose and tooltip of the tool it is holding, load the corresponding affordance model, and select and perform the action that will best contribute to achieving its goal (removing all objects), or otherwise ask for another tool.

8. Conclusion

8.1. Contribution

In the previous chapters, we have presented the work done during the development of this Thesis, concerning the topic of tool affordance learning from interaction with the environment, evaluated on the iCub robot. We have proposed and evaluated novel methods to: incorporate tools, i.e., estimate their shape, position and tooltip and attach it to the robot's kinematics (Chapter 4); represent tools in a way that correlates with their functionality, taking into account their geometry and pose (Chapter 5); and learn the relationship between the represented tool-poses and their functionality in function of the performed action, that is, the tool-pose affordances (Chapter 6). Moreover, we have shown how all these methods can be integrated in a coherent and robust behavior, which has served as a live demo (Chapter 7). We believe that the contributions to the state-of-the-art have been manifold, both theoretic and technical, as outlined below:

Theoretical Contributions:

- Introduction, definition and mathematical formalization of the concepts of *tool reference frame*, as well as the related concepts of *tool planes* and *tool axes*, *canonical pose* and *tooltip*. These constitute a general framework under which is possible to represent the tool's pose and its main characteristics, in a way that can be applied to the vast majority of radial tools.
- Introduction of the OMS-EGI descriptor, as the first holistic, robot-centric, functional feature specifically devised to represent a tool's geometry and pose in interaction scenarios.
- Reformulation of the formalization of affordances, so that it includes explicitly the embodiment of the robot and thus accounts for its possible modification (via tools, for example), while reducing the number of independent elements considered.

- Formalization of the concept of the affordance vector, as a representation of the effect that can be achieved with a tool(-pose), or on an object, in function of the applied action.

Technical Contributions:

- Introduction of methods for robust and automatic tool incorporation, which take advantage of the involvement of the robot in the action of holding and observing the tool.
- Implementation and assessment of the potential of the OMS-EGI feature with respect to canonical representations and to the parameter setting applied.
- Introduction, implementation and evaluation of methods of tool affordance prediction with increasing generalization capabilities and prediction performance.
- Integration of the proposed methods with reactive behaviors into a working live demo.

8.2. Challenges and future work

In the current Thesis we have presented a set of methods that allow a robot to incorporate a tool, represent it with respect to its own body and learn its affordances through interaction with the environment, considering also the pose in which the tool is grasped. However, we acknowledge that the scenarios devised to evaluate these methods are relatively simple. The application of the methods developed in the Thesis to more complex scenarios opens up new challenges for future research.

For example, one of the things that we have purposefully kept simple during the development of the current Thesis is the action repertoire available to the robot to interact with its environment. Larger action spaces would have increased the size of the search space, rendering its exploration unfeasible in a reasonable amount of time. A possible way to tackle this problem could be to define the actions in terms motor primitives which can get composed in order to form complex actions. This would enable the iCub to perform complex exploratory actions, while their representation is kept to a few parameters. Furthermore, simple actions could also be concatenated in order to perform complex behaviors by means of planning algorithms, allowing for the identification and completion of goals achievable through the robot's action repertoire.

Directly tied with the action, the representation of the target object and of the effect of acting on it have been also limited to its position, and the generated displacement, respectively. As explained before, this decision was made not only to reduce the search space on the object element of the affordance tuple so we could focus on the representation of tools and their poses, but also so that the effect of the action could be easily measured automatically. However, we recognise that the representation of the effect directly determines the kind of affordances that can be discovered, and therefore, learned. For example, with such a representation, the effect of hammering, or pouring, or cutting, would not be measurable and thus, not identifiable. Figuring out a representation of the effect that could encompass all these possibilities, and moreover, be automatically computed, is indeed a complex task out of the scope of this thesis, but nevertheless an interesting research question.

Another shortcoming of the present approach in its current state is that in all experiments, trials start with a tool already being held by the robot, that is, it does not support tool selection. The main reason for this is that although some of the presented architectures allow inverse mapping, which could predict the desired tool-pose features for a desired task, these features are dependent on the pose, hence, on how the tool is being grasped. Thus, there is no way to describe a tool with our proposed features without fixing a grasping pose, as the grasp obviously happens only after a tool has been selected. This conundrum prevents associating the features with "ungrasped" tools, and thus makes selecting among a set of available "ungrasped" tools unfeasible by inverse mapping. Yet, other approaches could be applied to enable tool selection, such as applying the learned models as forward models, which could predict the affordance of a number of "common" grasps of the available tools (left, front and right, for example), and use the best prediction to select tool *and* pose. Of course, most effects can be achieved by many different tool-poses, so extra constraints and evaluations would have to be put in place to limit the selection process. At this aim, object recognition software should be used to identify the available tools from vision, such as the one implemented for iCub in [130].

Human interaction could make the test scenarios and the resulting demos much more intuitive, if specific tasks or goals could be verbally asked to the robot. In the current study, all the data available to the robot, both tools and affordances representations, has been presented in terms of low level descriptors directly linked to the robots perceptions and action. This representations could be easily used to ground higher level symbols, by, for example, clustering the SOM representations of tools and affordances,

and assigning verbal labels to them. This approach could provide definitions of concepts such as "capable of pulling" to the robot based on its own experience, which would ease interaction with other agents.

In the longer term, another aspect that should be studied in more depth are the online learning capabilities of the system. Similar to tool selection, the current methods allow in principle to be trained online. However, the convenience and performance of learning incrementally should be evaluated, and probably some changes be applied in order to make it effective. Theoretically, this approach could render the methods more apt for long term exploration allowing to experiment with much larger datasets, tools and richer set of actions.

Also, if similar systems are applied for a larger number of activities, we should take into account that in order to perform some tasks, specific large scale tool characteristics may be required that can not be defined by low level features, and viceversa for high level features and detailed characteristics. For example, screwing and pushing can be applied with a screwdriver, but the former requires a very specific cross shape on the tip, while the latter depends on the length of the tool. Therefore, although OMS-EGI already encapsulates some hierarchical structure in its multi-scale resolution, it might be beneficial to evaluate some other features where the hierarchical structure is more explicit, so that tool characteristics can be described simultaneously at all levels.

This is also in line with the current trend of research into end-to-end learning systems, which automatically learn hierarchical features that best suit the required prediction tasks. It would be very interesting to test if such an approach could be applied to the scenario of affordance learning from interaction. The closest approach we know of was the one published in [89], but it was based on reinforcement learning for particular tasks. We wonder if similar techniques could scale to more goal-free exploration approaches such as the one presented here, and whether raw 3D information could be provided as an input, from which the system would learn the relevant features for affordance prediction.

8.3. Conclusions

In this Thesis, we have presented a set of methods to represent, learn and generalize tool affordances based on the tools geometry and the way in which they are grasped, implemented and validated in the iCub robot and its simulator on a large dataset of tools. In order to retrieve all the necessary information about the considered tools, we

have described a complete method for autonomous tool incorporation, which enables the robot to recognize tools, reconstruct their geometry, estimate their pose and determine the position of the tooltip. As part of this approach, we have introduced mathematical formalizations of tool pose, frame and tooltip, which can be applied beyond the scope of the present work. Then, we have presented two sets of functional features in order to represent tools in interactive scenarios, considering both their pose and their 2D and 3D geometry, respectively. Based on these representations and on a novel formalization of affordances adapted to encompass tool use, we have proposed and evaluated a set of architectures and experimental setups in order to learn tool affordances from interactions. While some general elements are shared among all these architectures, the focus varies in the machine learning techniques applied as well as the way in which the elements of an affordance are characterized and connected. Together, they represent an incremental approach towards more capable methods for tool affordance prediction, in terms of both accuracy and generalization. Finally, we have described how these methods have been integrated into a live demo which displays useful and robust behaviors on the iCub. Overall, we believe that the work developed during this Thesis has achieved the desired goal of advancing the state-of-the-art with respect to tool affordance learning and tool use with humanoid robots, by introducing the aforementioned novel and effective methods.

Bibliography

[1] Paulo Abelha, Frank Guerin, and Markus Schoeler. A Model-Based Approach to Finding Substitute Tools in 3D Vision Data. In *International Conference on Robotics and Automation*, pages 2471–2478. IEEE, may 2016.

[2] Karen E. Adolph and Kari S Kretch. Gibson's Theory of Perceptual Learning. In *International Encyclopedia of the Social and Behavioral Sciences*, volume 10, pages 127–134. 2015.

[3] Philip E Agre and David Chapman. Pengi: An Implementation of a Theory of Activity. *Proceedings of the Sixth National Conference on Artificial Intelligence*, 278:268–272, 1987.

[4] John Alcock. The evolution of the use of tools by feeding animals. *Evolution*, pages 464–473., 1972.

[5] A Alexandre. 3D Descriptors for Object and Category Recognition : a Comparative Evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[6] Alexandre Antunes, Lorenzo Jamone, Giovanni Saponaro, Alexandre Bernardino, and Rodrigo Ventura. From Human Instructions to Robot Actions : Formulation of Goals , Affordances and Probabilistic Planning. *IEEE Transactions on Cognitive and Developmental Systems*, (May):16–21, 2016.

[7] Michael A Arbib, James B Bonaiuto, Stéphane Jacobs, and Scott H Frey. Tool use and the distalization of the end-effector. *Psychological Research*, 73(4):441–462, 2009.

[8] Ronald C. Arkin. *Behavior-based robotics*. MIT Press, 1998.

[9] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. Cognitive Developmental Robotics: A Survey. *IEEE Transactions on Autonomous Mental Development*, 1(1):12–34, may 2009.

[10] Carl Barck-holst, Maria Ralph, Fredrik Holmar, and Danica Kragic. Learning Grasping Affordance Using Probabilistic and Ontological Approaches. In *International Conference on Advanced Robotics, 2009.*, pages 1–6, 2009.

[11] Eleonora Bartoli, Laura Maffongelli, Marco Jacono, and Alessandro D'Ausilio. Representing tools as hand movements: Early and somatotopic visuomotor transformations. *Neuropsychologia*, 61:335–344, 2014.

[12] B.B. Beck. *Animal tool behaviour: the use and manufacture of tools by animals*. Garland STPM Publishing, New York, New York, USA, 1980.

[13] Randall D. Beer. Intelligence as adaptive behavior: an experiment in computational neuroethology. Academic Press Professional, 1990.

[14] Antonio Bicchi and Centro E Piaggio. Robotic Grasping and Contact : A Review. In *International Conference on Robotics and Automation*, pages 348–353, 2000.

[15] Ergun Bicici and Robert St Amant. Reasoning About the Functionality of Tools and Physical Artifacts. Technical report, 2003.

[16] Luca Bogoni. An Active Approach to Characterization and Recognition of Functionality and Functional Properties. Technical Report May, 1993.

[17] Luca Bogoni. *Identification of Functional Features through Observations and Interactions*. PhD thesis, 1995.

[18] Rodney A Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1811):139–159, 1991.

[19] Solly Brown and Claude Sammut. Tool Use Learning in Robots. In *Advances in Cognitive Systems*, pages 58–65, 2011.

[20] Maya Cakmak, Mehmet R Dogar, Emre Ugur, and Erol Sahin. Affordances as a Framework for Robot Control. In *Proceedings of the Seventh International Conference on Epigenetic Robotics*, 2007.

[21] B. Campbell. *Human Evolution: An Introduction to Man's Adaptations*. Aldine, New York, 3rd edition, 1985.

[22] Angelo Cangelosi, Giorgio Metta, Gerhard Sagerer, Stefano Nolfi, Chrystopher Nehaniv, Kerstin Fischer, Jun Tani, Tony Belpaeme, Giulio Sandini, Francesco Nori, Luciano Fadiga, Britta Wrede, Katharina Rohlfing, Elio Tuci, Kerstin Dautenhahn, Joe Saunders, and Arne Zeschel. Integration of Action and Language Knowledge: A Roadmap for Developmental Robotics. *IEEE Transactions on Autonomous Mental Development*, 2(3):167–195, sep 2010.

[23] Chih-chung Chang and Chih-jen Lin. LIBSVM : A Library for Support Vector Machines. Technical report, 2013.

[24] Yu-wei Chao, Zhan Wang, Rada Mihalcea, and Jia Deng. Mining Semantic Affordances of Visual Object Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[25] David. Chapman and David. *Vision, instruction, and action*. MIT Press, 1991.

[26] Anthony Chemero. An Outline of a Theory of Affordances. *Ecological Psychology*, 15(2):181–195, 2003.

[27] Carlo Ciliberto, Ugo Pattacini, Lorenzo Natale, Francesco Nori, and Giorgio Metta. Reexamining Lucas-Kanade method for real-time independent motion detection: Application to the iCub humanoid robot. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160, sep 2011.

[28] Claire F. Michaels and Claudia Carello. *Direct perception*. 1981.

[29] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):1–37, 2002.

[30] Ignasi Cos-Aguilera, Lola Canamero, and Gillian Hayes. Using a SOFM to learn Object Affordances. Technical Report March, 2004.

[31] Charles Darwin. The descent of man and selection in relation to sex, in Charles Darwin, The origin of species and The descent of man (combined volume). *Journal of anatomy and physiology*, 5(Pt 2):363–372, 1871.

[32] David L. Davies and Donald W. Bouldin. A Cluster Separation Measure. *IEEE transactions on pattern analysis and machine intelligenceP*, 1(2):224–227, 1979.

[33] Ignacio de la Torre. The origins of stone tool technology in Africa: a historical perspective. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 366(1567):1028–37, apr 2011.

[34] Atabak Dehban, Lorenzo Jamone, Adam R Kampff, and José Santos-Victor. Denoising Auto-encoders for Learning of Objects and Tools Affordances in Continuous Space. In *International Conference on Robotics and Automation*, pages 1–6, 2016.

[35] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J Piater. Learning Grasp Affordance Densities. *Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[36] Renaud Detry, E. Baseski, Mila Popovic, Y. Touati, O. Kroemer, Norbert Krüger, Jan Peters, Justus Piater, O. Kroemer, Justus Piater, and Jan Peters. Learning Continuous Grasp Affordances by Sensorimotor Exploration. In *From Motor Learning to Interaction Learning in Robots*, volume 264, pages 451–465. Springer-Verlag Berlin Heidelberg, 2010.

[37] John Dewey. The Reflex Arc Concept in Psychology. *Psychological Review*, 3(4):357–370, 1896.

[38] Jonathan F Diaz, Alexander Stoytchev, Ronald C Arkin, and U S A Georgia. Exploring Unknown Structured Environments. In *FLAIRS*, pages 145–149, 2001.

[39] Mehmet R. Dogar, Maya Cakmak, Emre Ugur, and Erol Sahin. From primitive behaviors to goal-directed behavior using affordances. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 729–734, oct 2007.

[40] Dobromir G Dotov, Lin Nie, and Matthieu M De Wit. Understanding affordances : of Gibson ' s central concept. *Avant: Journal of the Philosophical-Interdisciplinary Vanguard*, III(2):28–39, 2012.

[41] a. P. Duchon, L. P. Kaelbling, and W. H. Warren. Ecological Robotics. *Adaptive Behavior*, 6(3-4):473–507, jan 1998.

[42] S R Fanello, U Pattacini, I Gori, and V Tikhanoff. 3D Stereo Estimation and Fully Automated Learning of Eye-Hand Coordination in Humanoid Robots. In *Humanoids 2014*, pages 1028–1035, 2014.

[43] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, sep 2004.

[44] Pier Francesco Ferrari, Stefano Rozzi, and Leonardo Fogassi. Mirror neurons responding to observation of actions made with tools in monkey ventral premotor cortex. *Journal of cognitive neuroscience*, 17:212–226, 2005.

[45] Severin Fichtl, Dirk Kraft, Norbert Kruger, and Frank Guerin. Bootstrapping Relational Affordances of Object Pairs using Transfer. *IEEE Transactions on Cognitive and Developmental Systems*, 8920(c):1–1, 2016.

[46] Severin Fichtl, Andrew McManus, Wail Mustafa, Dirk Kraft, Norbert Krüger, and Frank Guerin. Learning Spatial Relationships From 3D Vision Using Histograms. *Icra*, 2014.

[47] Sanja Fidler and Ales Leonardis. Towards Scalable Representations of Object Categories : Learning a Hierarchy of Parts. In *International Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[48] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini. Learning About Objects Through Action - Initial Steps Towards Artificial Cognition. In *Procceedings of the IEEE Intrernational Conference on Robotics and Automation*, pages 3140–3145, 2003.

[49] Christopher Geib, Kira Mour, Ron Petrick, Nico Pugeault, Mark Steedman, Norbert Krueger, and W Florentin. Object Action Complexes as an Interface for Planning and Robot Control. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006).*, pages 4–6, 2006.

[50] Eleanor J. Gibson. *Principles of perceptual learning and development*, volume 4. 1969.

[51] Eleanor J. Gibson. Perceptual Learning in Development: Some Basic Concepts. *Ecological Psychology*, 12(4):295–302, oct 2000.

[52] Eleanor Jack. Gibson and Anne D. Pick. *An ecological approach to perceptual learning and development*. Oxford University Press, 2000.

[53] J J Gibson. The theory of affordances. In Robert Shaw and John Bransford, editors, *Perceiving acting and knowing Toward an ecological psychology*, volume Perceiving, chapter 3, pages 67–82. Lawrence Erlbaum, 1977.

[54] J J Gibson. *The ecological approach to visual perception*, volume 39. 1979.

[55] J J Gibson. Gibson Theory of Affordances.pdf, 1986.

[56] Arren Glover. *Developing grounded representations for robots through the principles of sensorimotor coordination*. PhD thesis, Queensland University of Technology, 2014.

[57] Arren J Glover and Gordon F Wyeth. Towards Lifelong Affordance Learning using a Distributed Markov Model. 2016.

[58] Afonso Gonçalves, João Abrantes, Giovanni Saponaro, Lorenzo Jamone, and Alexandre Bernardino. Learning Intermediate Object Affordances : Towards the Development of a Tool Concept. In *IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob 2014)*, number October, pages 1–8, 2014.

[59] Afonso Gonçalves, Giovanni Saponaro, Lorenzo Jamone, and Alexandre Bernardino. Learning visual affordances of objects and tools through autonomous robot exploration. *2014 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2014*, (May):128–133, 2014.

[60] Ilaria Gori, Ugo Pattacini, Vadim Tikhanoff, and Giorgio Metta. Three-finger precision grasp on incomplete 3D point clouds. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5366–5373, 2014.

[61] Charles De Granville, Di Wang, Joshua Southerland, Robert Platt, and Andrew H Fagg. Grasping Affordances : Learning to Connect Vision to Hand Action. In *The Path to Autonomous Robots*, pages 1–22. 2009.

[62] James G. Greeno. Gibson's affordances. *Psychological Review*, 101(2):336–342, 1994.

[63] Shane Griffith, Jivko Sinapov, Matthew Miller, Alexander Stoytchev, and A Developmental Psychology. Toward Interactive Learning of Object Categories by a Robot : A Case Study with Container and Non-Container Objects. In *IEEE 8TH INTERNATIONAL CONFERENCE ON DEVELOPMENT AND LEARNING Toward*, pages 1–6, 2009.

[64] Shane Griffith, Jivko Sinapov, Vladimir Sukhoy, and Alexander Stoytchev. A Behavior–Grounded Approach to Forming Object Categories: Separating Containers from Non-Containers. *IEEE Transactions on Autonomous Mental Development*, 4(1):54–69, 2012.

[65] Frank Guerin, Norbert Krüger, and Dirk Kraft. A Survey of the Ontogeny of Tool Use: From Sensorimotor Experience to Planning. *IEEE Transactions on Autonomous Mental Development*, 5(1):18–45, 2013.

[66] Harry Heft. *Ecological psychology in context : James Gibson, Roger Barker, and the legacy of William James's radical empiricism.* 2001.

[67] Tucker Hermans, James M Rehg, and Aaron Bobick. Affordance Prediction via Learned Object Attributes. In *IEEE International Conference on Robotics and Automation (ICRA 2011). Workshop on Semantic Perception, Mapping, and Exploration*, 2011.

[68] Rob Hess and Alan Fern. Discriminatively trained particle filters for complex multi-object tracking. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 240–247, 2009.

[69] Berthold K P Horn. EXTENDED GAUSSIAN IMAGES. *Proceedings of the IEEE*, 72(12):1671–1686, 1984.

[70] Thomas Eugene Horton. *A Partial Contour Similarity-Based Approach to Visual Affordances in Habile Agents.* PhD thesis, 2011.

[71] Itseez. Open Source Computer Vision Library, 2015.

[72] Shahram Izadi, Andrew Davison, Andrew Fitzgibbon, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Dustin Freeman. Kinect Fusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, page 559, 2011.

[73] Jeffrey J. Lockman. A perception–action perspective on tool use development. *Child development*, 71(1):137–144, 2000.

[74] Raghvendra Jain and Tetsunari Inamura. Learning of Tool Affordances for autonomous tool manipulation. *2011 IEEE-SICE International Symposium on System Integration SII*, pages 814–819, 2011.

[75] Raghvendra Jain and Tetsunari Inamura. Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools. *Artificial Life and Robotics*, 18(1-2):95–103, sep 2013.

[76] Lorenzo Jamone, Alexandre Bernardino, and Jose Santos-Victor. Benchmarking the grasping capabilities of the iCub hand with the YCB Object and Model Set. *IEEE Robotics and Automation Letters*, 3766(c):1–1, 2016.

[77] Lorenzo Jamone, Emre Ugur, Angelo Cangelosi, Luciano Fadiga, Alexandre Bernardino, Justus Piater, and Jose Santos-Victor. Affordances in psychology, neuroscience and robotics: a survey. *IEEE Transactions on Cognitive and Developmental Systems*, (August):1–1, 2016.

[78] David Inkyu Kim. Semantic Labeling of 3D Point Clouds with Object Affordance for Robot Manipulation. In *International Conference on Robotics and Automation*, pages 5578–5584, 2014.

[79] Hedvig Kjellström, Javier Romero, and Danica Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 115(1):81–90, jan 2011.

[80] Kurt Koffka. PRINCIPLES OF GESTALT PSYCHOLOGY Chapter I Why Psychology? AN INTRODUCTORY QUESTION. *Principles of Gestalt Psychology*, pages 1–14, 1935.

[81] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[83] Eric Krotkov. Robotic Perception of Material. *Proceedings of the IJCAI*, (March):88–94, 1995.

[84] Norbert Krüger, Justus Piater, Florentin Wörgötter, Christopher Geib, Ron Petrick, Mark Steedman, Tamim Asfour, Dirk Kraft, Damir Omr, Bernhard Hommel, Alejandro Agostini, Danica Kragic, and Jan-olof Eklundh. A Formal Definition of Object-Action Complexes and Examples at Different Levels of the Processing Hierarchy. Technical report, 2009.

[85] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised Feature Learning for 3D Scene Labeling. In *IEEE International Conference on Robotics and Automation (ICRA 2014)*, pages 3050–3057, 2014.

[86] Yann Lecun. LNCS 7583 - Learning Invariant Feature Hierarchies. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 496–505, 2012.

[87] Honglak Lee. *Unsupervised Feature Learning Via Sparse Hierarchical Representations*. PhD thesis, 2010.

[88] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5), 2015.

[89] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[90] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv*, pages 1–1, 2016.

[91] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Affordance-based imitation learning in robots. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1015–1021, oct 2007.

[92] Christopher Lörken. *Introducing Affordances into Robot Task Execution*. PhD thesis, 2007.

[93] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, dec 2003.

[94] Karl F Macdorman. Grounding Symbols through Sensorimotor Integration. *ROBOTICS SOCIETY OF JAPAN*, 17:20–24, 1999.

[95] Marianna Madry, Carl Henrik Ek, Renaud Detry, Kaiyu Hang, and Danica Kragic. Improving generalization for 3D object categorization with Global Structure Histograms. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1379–1386. Ieee, oct 2012.

[96] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Multi-model approach based on 3D functional features for tool affordance learning in robotics. In *Humanoids 2015*, Seoul, 2015.

[97] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot. In *International Conference on Robotics and Automation*, pages 3200 – 3206, 2015.

[98] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of tool affordances from 3D tool representation through parallel SOM mapping. In *International Conference on Robotics and Automation*, 2017.

[99] Tanis Mar, Vadim Tikhanoff, and Lorenzo Natale. What can I do with this tool? Self-supervised learning of tool affordances from their 3D geometry. 2017.

[100] Angelo Maravita and Atsushi Iriki. Tools for the body (schema). *Trends in cognitive sciences*, 8(2):79–86, feb 2004.

[101] Angelo Maravita, Charles Spence, and Jon Driver. Multisensory integration and the body schema: close to hand and within reach. *Current Biology*, 13(13):R531–R539, jul 2003.

[102] David Marr and David. *Vision : a computational investigation into the human representation and processing of visual information*. W.H. Freeman, 1982.

[103] Ruben Martinez-Cantin, Manuel Lopes, and Luis Montesano. Body schema acquisition through active learning. In *2010 IEEE International Conference on Robotics and Automation*, pages 1860–1866. Ieee, may 2010.

[104] Jeffrey K. McKee, Frank E. Poirier, and W. Scott Mcgraw. *Understanding human evolution*. 2015.

[105] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: an open-systems platform for research in cognitive development. *Neural networks : the official journal of the International Neural Network Society*, 23(8-9):1125–34, 2010.

[106] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The iCub humanoid robot : an open platform for research in embodied cognition. *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 50–56, 2008.

[107] Huaqing Min, Changan Yi, Ronghua Luo, Sheng Bi, Xiaowen Shen, and Yuguang Yan. Affordance Learning Based on Subtask's Optimal Strategy. *International Journal of Advanced Robotic Systems*, page 1, 2015.

[108] Huaqing Min, Chang'an Yi, Ronghua Luo, Jinhui Zhu, and Sheng Bi. Affordance Research in Developmental Robotics: A Survey. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):1–1, 2016.

[109] Bogdan Moldovan and Luc De Raedt. Learning relational affordance models for two-arm robots. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):2916–2922, 2014.

[110] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, Jose Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. *2012 IEEE International Conference on Robotics and Automation*, pages 4373–4378, may 2012.

[111] Luis Montesano and Manuel Lopes. Learning grasping affordances from local visual descriptors. In *2009 IEEE 8th International Conference on Development and Learning*, pages 1–6. IEEE, 2009.

[112] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and Jose Santos-Victor. Modeling affordances using Bayesian networks. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4102–4107. IEEE, oct 2007.

[113] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and JosÉ Santos-Victor. Learning Object Affordances: From Sensory–Motor Coordination to Imitation. *IEEE Transactions on Robotics*, 24(1):15–26, feb 2008.

[114] R.R. Murphy. Case studies of applying Gibson's ecological approach to mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(1):105–111, 1999.

[115] Austin Myers, Ching L Teo, Cornelia Ferm, and Yiannis Aloimonos. Affordance Detection of Tool Parts from Geometric Features. In *International Conference on Robotics and Automation*, pages 1374–1381, 2015.

[116] Cota Nabeshima, Yasuo Kuniyoshi, and Max Lungarella. Adaptive body schema for robotic tool-use. *Advanced Robotics*, 20(10):1105–1126, oct 2006.

[117] Vinod Nair and Geoffrey E Hinton. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.

[118] Lorenzo Natale, Ali Paikan, Marco Randazzo, and Daniele E. Domenichelli. The iCub Software Architecture: Evolution and Lessons Learned. *Frontiers in Robotics and AI*, 3(April):1–21, 2016.

[119] Nikhilesh Natraj, Victoria Poole, J. C. Mizelle, Andrea Flumini, Anna M. Borghi, and Lewis a. Wheaton. Context and hand posture modulate the neural dynamics of tool-object perception. *Neuropsychologia*, 51(3):506–519, 2013.

[120] Ulric Neisser. *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W.H. Freeman & Company, 1976.

[121] Anh Nguyen, Dimitrios Kanoulas, Darwin G Caldwell, and Nikos G Tsagarakis. Detecting Object Affordances with Convolutional Neural Networks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.*, pages 2765–2770, 2016.

[122] Shun Nishide, Jun Tani, Toru Takahashi, Hiroshi G. Okuno, Tetsuya Ogata, and H.G. Nishide, S. ; Dept. of Intell. Sci. & Technol., Kyoto Univ., Kyoto, Japan ; Jun Tani ; Takahashi, T. ; Okuno. Tool–Body Assimilation of Humanoid Robot Using a Neurodynamical System. *Autonomous Mental Development, IEEE Transactions on*, 4(2):139–149, jun 2012.

[123] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[124] Pedro Osório, Alexandre Bernardino, and Ruben Martinez-cantin. Gaussian Mixture Models for Affordance Learning using Bayesian Networks. In *International Conference on Intelligent Robots and Systems*, pages 1–6, 2010.

[125] Erhan Oztop, Nina S Bradley, and Michael a Arbib. Infant grasp learning: a computational model. *Experimental brain research*, 158(4):480–503, oct 2004.

[126] Ali Paikan, Vadim Atikhanoff, Giorgio Metta, Lorenzo Natale, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Enhancing software module reusability using port plug-ins: An experiment with the iCub robot. *IEEE International Conference on Intelligent Robots and Systems*, pages 1555–1562, 2014.

[127] Ali Paikan, Daniele E. Domenichelli, and Lorenzo Natale. Communication channel prioritization in a publish-subscribe architecture. *Software Engineering and Architectures for Realtime Interactive Systems Working Group SEARIS 2015*, 2015.

[128] Ali Paikan, Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Data Flow Port Monitoring and Arbitration. *Pasa.Liralab.It*, 5(May):80–88, 2014.

[129] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. Real-world Object Recognition with Off-the-shelf Deep Conv Nets: How Many Objects can iCub Learn? *arXiv:1504.03154 [cs]*, 2015.

[130] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. Teaching iCub to recognize objects using deep Convolutional Neural Networks. *Proceedings of The 4th Workshop on Machine Learning for Interactive Systems*, pages 21–25, 2015.

[131] Giulia Pasquale, Tanis Mar, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Enabling depth-driven visual

attention on the iCub humanoid robot: Instructions for use and new perspectives. *Frontiers in Robotics and AI*, 3(June):1–11, 2016.

[132] Ugo Pattacini. *Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub*. PhD thesis, 2011.

[133] Ugo Pattacini, Francesco Nori, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1668–1674, oct 2010.

[134] Herbert L. Pick. Learning to Perceive and Perceiving to Learn. *Developmental Psychology*, 28(5):787–794, 1992.

[135] Lerrel Pinto and Abhinav Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. *Icra*, 2016.

[136] Victor Adrian Prisacariu, Aleksandr V. Segal, and Ian Reid. Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7724 LNCS(PART 1):593–606, 2013.

[137] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[138] Fei-wei Qin, Lu-ye Li, Shu-ming Gao, Xiao-ling Yang, and Xiang Chen. A deep learning approach to the classification of 3D CAD models. *Journal of Zhejiang University SCIENCE C*, 15(2):91–106, 2014.

[139] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, jun 2007.

[140] Carl Yuheng Ren, Victor Prisacariu, David Murray, and Ian Reid. STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1561–1568, 2013.

[141] B. Ridge, D. Skocaj, and A. Leonardis. Self-Supervised Cross-Modal Online Learning of Basic Object Affordances for Developmental Robotic Systems. In *IEEE International Conference on Robotics and Automation*, pages 5047–5054, 2010.

[142] Barry Ridge, Ales Leonardis, Ales Ude, Miha Denisa, and Danijel Skocaj. Self-Supervised Online Learning of Basic Object Push Affordances. *International Journal of Advanced Robotic Systems*, (November):1, 2015.

[143] Barry Ridge and Ales Ude. Action-grounded push affordance bootstrapping of unknown objects. *IEEE International Conference on Intelligent Robots and Systems*, (January):2791–2798, 2013.

[144] Barry Ridge and Ales Ude. Action-grounded surface geometry and volumetric shape feature representations for object affordance prediction. *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, (December):1022–1028, 2016.

[145] Alessandro Roncone, Ugo Pattacini, Giorgio Metta, and Lorenzo Natale. A Cartesian 6-DoF Gaze Controller for Humanoid Robots. *Robotics: Science and Systems (RSS)*, 2016.

[146] Anirban Roy and Sinisa Todorovic. A Multi-Scale CNN for Affordance Segmentation in RGB Images. In *European Conference on Computer Vision (ECCV2016)*, pages 186–201, 2016.

[147] Radu Bogdan Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *Kuenstliche Intelligenz*, 24(4):1–4, aug 2010.

[148] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[149] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[150] E Sahin, M Cakmak, M R Dogar, E Ugur, and G Ucoluk. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. *Adaptive Behavior*, 15(4):447–472, 2007.

[151] Marti Sanchez-Fibla, Armin Duff, and Paul F.M.J. Verschure. The acquisition of intentionally indexed and object centered affordance gradients: A biomimetic controller and mobile robotics benchmark. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1115–1121, sep 2011.

[152] Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[153] T. Serre, L. Wolf, and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2:994–1000, 2005.

[154] R. E. Shaw, M. T. Turvey, and W. M. Mace. Ecological psychology. The consequence of a commitment to realism. *Cognition and Symbolic processes*, 2:159–226, 1982.

[155] K.B. Shimoga. Robot Grasp Synthesis Algorithms: A Survey. *The International Journal of Robotics Research*, 15(3):230–266, jun 1996.

[156] Robert W. Shumaker, Kristina R. Walkup, and Benjamin B. Beck. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011., 2011.

[157] Jivko Sinapov, Taylor Bergquist, Connor Schenck, Ugonna Ohiri, Shane Griffith, and Alexander Stoytchev. Proprioceptive and Auditory Feedback. *The International Journal of Robotics Research*, 30(10):1250–1262, 2011.

[158] Jivko Sinapov and Alexadner Stoytchev. Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *2008 7th IEEE International Conference on Development and Learning*, pages 91–96. Ieee, aug 2008.

[159] Kasey C Soska, Karen E Adolph, and Scott P Johnson. Systems in development: motor skill acquisition facilitates three-dimensional object completion. *Developmental psychology*, 46(1):129–38, jan 2010.

[160] D F Specht. A general regression neural network. *Neural Networks, IEEE Transactions on*, 2(6):568–576, 1991.

[161] Abhilash Srikantha and Juergen Gall. Weakly Supervised Learning of Affordances. *arXiv preprint arXiv*, pages 1–16, 2016.

[162] Robert St Amant and Thomas E. Horton. Revisiting the definition of animal tool use. *Animal Behaviour*, 75(4):1199–1208, 2008.

[163] Michael Stark, Philipp Lies, Michael Zillich, Jeremy Wyatt, and Bernt Schiele. Learned Affordance Cues. Technical report, 2008.

[164] Mark Steedman. Plans, Affordances, And Combinatory Grammar. *Linguistics and Philosophy*, 25(5/6):723–753, 2002.

[165] Thomas Stoffregen. Affordances as properties of the animal environment system. *Ecological Psychology*, 15(2):115–134, 2003.

[166] Thomas A. Stoffregen, Chih-Mei Yang, and Benoît G. Bardy. Affordance judgments and non-locomotor body movement. *Ecological Psychology*, 17(2):75–104, 2005.

[167] A. Stoytchev. Some Basic Principles of Developmental Robotics. *IEEE Transactions on Autonomous Mental Development*, 1(2):122–130, aug 2009.

[168] Alexander Stoytchev. Toward Learning the Binding Affordances of Objects : A Behavior-Grounded Approach. In *AAAI Symposium on Developmental Robotics*, pages 21–23, 2005.

[169] Alexander Stoytchev. *Robot tool behavior: A developmental approach to autonomous tool use*. PhD thesis, Georgia Institute of Technology, 2007.

[170] Yuyin Sun, Liefeng Bo, and Dieter Fox. Attribute based object identification. *2013 IEEE International Conference on Robotics and Automation*, pages 2096–2103, 2013.

[171] Kuniyuki Takahashi, Tetsuya Ogata, and Hadi Tjandra. Tool – body Assimilation Model Based on Body Babbling and Neuro-dynamical System. In *International Conference on Artificial Neural Networks*, volume 2015, 2014.

[172] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, dec 2007.

[173] V Tikhanoff, A Cangelosi, P Fitzpatrick, G Metta, L Natale, and F Nori. An Open-Source Simulator for Cognitive Robotics Research: The Prototype of the iCub Humanoid Robot Simulator. In *Workshop on Performance Metrics for Intelligent Systems*, 2008.

[174] Vadim Tikhanoff, Ugo Pattacini, Ieee Senior Member, Lorenzo Natale, and Giorgio Metta. Exploring affordances and tool use on the iCub. In *Humanoids 2013*, 2013.

[175] Trimble. 3D modeling for everyone | SketchUp, 2016.

[176] M T Turvey. Affordances and Prospective Control: An Outline of the Ontology. *Ecological Psychology*, 4:173–187, 1992.

[177] Emre Ugur, Mehmet R. Dogar, Maya Cakmak, and Erol Sahin. Curiosity-driven learning of traversability affordance on a mobile robot. *2007 IEEE 6th International Conference on Development and Learning*, pages 13–18, jul 2007.

[178] Emre Ugur, Mehmet R. Dogar, Cakmak Maya, and Erol Şahin. The learning and use of traversability affordance using range images on a mobile robot. In *International Conference on Robotics and Automation*, number April, pages 10–14, 2007.

[179] Emre Ugur and S Erol. Traversability : A Case Study for Learning and Perceiving Affordances in Robots. *Adaptive Behavior*, 18(3-4):1–28, 2010.

[180] Emre Ugur, Erhan Oztop, and Erol Sahin. Goal emulation and planning in perceptual space using learned affordances. *Robotics and Autonomous Systems*, 59(7-8):580–595, jul 2011.

[181] Emre Ugur, Erhan Oztop, and Erol Sahin. Going beyond the perception of affordances: Learning how to actualize them through behavioral parameters. In *2011 IEEE International Conference on Robotics and Automation*, pages 4768–4773. Ieee, may 2011.

[182] Emre Ugur and Justus Piater. Emergent Structuring of Interdependent Affordance Learning Tasks. In *IEEE International Conference on Development and Learning and Epigenetic Robotics*, pages 481–486, 2014.

[183] Emre Ugur and Justus Piater. Bottom-Up Learning of Object Categories , Action Effects and Logical Rules : From Continuous Manipulative Exploration to Symbolic Planning. In *International Conference on Robotics and Automation*, 2015.

[184] Emre Ugur and Justus Piater. Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. *IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS*, pages 1–13, 2016.

[185] Emre Ugur, Erol Sahin, and Erhan Oztop. Predicting future object states using learned affordances. *2009 24th International Symposium on Computer and Information Sciences*, pages 415–419, sep 2009.

[186] Emre Ugur, Erol Sahin, and Erhan Oztop. Self-discovery of motor primitives and learning grasp affordances. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3260–3267, oct 2012.

[187] Emre Ugur, Sandor Szedmak, and Justus Piater. Bootstrapping paired-object affordance learning with learned

single-affordance features. In *IEEE International Conference on Development and Learning and Epigenetic Robotics*, pages 468–473, 2014.

[188] Laurens van der Maaten, Geoffrey E. Hinton, Laurens van der Maaten, and Geoffrey E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[189] Ad; van Leeuwen, Lieselotte; Smitsman and Cees van Leeuwen. Affordances, perceptual complexity, and the development of tool use. *Journal of Experimental Psychology*, 20(1):174–191, 1994.

[190] Vladimir Vapnik. *The nature of statistical learning theory*. 2013.

[191] T. Vatanen, M. Osmala, T. Raiko, K. Lagus, M. Sysi-Aho, M. Orešič, T. Honkela, and H. Lähdesmäki. Self-organization and missing values in SOM and GTM. *Neurocomputing*, 147:60–70, 2015.

[192] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Self-Organizing Map in Matlab: the SOM Toolbox. In *Matlab DSP Conference*, pages 35–40, 2000.

[193] Elisabetta Visalberghi and Luca Limongelli. Lack of comprehension of cause-ffect relations in tool-using capuchin monkeys. *Journal of Comparative Psychology*, 108(1):15–22, 1994.

[194] Jakob von Uexkull and Jakob von Uexkull. *Kompositionslehre der Natur*. Propylaen-Verl, 1920.

[195] Dejan V Vranić and Dietmar Saupe. 3D model retrieval. In *Proc. Spring Conference on Computer Graphics and its Applications (SCCG2005)*, pages 89–93, 2005.

[196] S L Wahsburn. Tools and human evolution. *Scientific American*, 203:63–75, sep 1960.

[197] W H Warren. Perceiving affordances: visual guidance of stair climbing. *Journal of experimental psychology. Human perception and performance*, 10(5):683–703, oct 1984.

[198] W H Warren and S Whang. Visual guidance of walking through apertures: body-scaled information for affordances. *Journal of experimental psychology. Human perception and performance*, 13(3):371–383, 1987.

[199] Handy Wicaksono and Claude Sammut. Relational Tool Use Learning by a Robot in a Real and Simulated World.

[200] Alexander B. Wood, Thomas E. Horton, R St Amant, and Robert St. Amant. Effective Tool Use in a Habile Agent. *2005 IEEE Design Symposium, Systems and Information Engineering*, pages 75–81, 2005.

[201] Zhirong Wu and Shuran Song. 3D ShapeNets : A Deep Representation for Volumetric Shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, pages 1–9, 2015.

[202] Di Xu, Jianfei Cai, Tat Jen Cham, Philip Fu, and Juyong Zhang. Kinect-Based Easy 3D Object Reconstruction. In *Pacific-Rim Conference on Multimedia*, pages 476–483, 2012.

[203] Dengsheng Zhang and Guojun Lu. A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures. Technical report, 2001.

[204] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, jan 2004.

[205] Lisha Zhang, Manuel João, and Alfredo Ferreira. Survey on 3D shape descriptors. Technical report, 2004.

[206] T.Y. Zhang and C.Y. Suen. A Fast Parallel Algorithm for Thinning Digital Patterns. *Image Processing and Computer Vision*, 27(3):236–239, 1984.

[207] Yiwei Zhang, Graham M. Gibson, Rebecca Hay, Richard W. Bowman, Miles J. Padgett, and Matthew P. Edgar. A fast 3D reconstruction system with a low-cost camera accessory. *Scientific Reports*, 5:10909, 2015.

[208] Yixin Zhu, Yibiao Zhao, and S Zhu. Understanding Tools: Task-Oriented Object Modeling, Learning and Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2855–2864, 2015.

ISTITUTO ITALIANO DI TECNOLOGIA
DEPARTMENT OF ICUB FACILITY
Via Morego, 30
16163 Genova
www.iit.it