What can I do with this tool? Self-supervised learning of tool affordances from their 3D geometry.

Tanis Mar[1], Vadim Tikhanoff[1], Lorenzo Natale[1]

***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. Robots capable of autonomously learning affordances in a similar self-supervised way would be far more versatile and simpler to design than purpose-specific ones. This paper proposes and evaluates an approach to allow robots to learn tool affordances from interaction, and generalize them among similar tools based on their 3D geometry. A set of actions is performed by the iCub robot with a large number of tools grasped in different poses, and the effects observed. Tool affordances are learned as a regression between tool-pose features and action-effect vector projections on respective Self-Organizing Maps, which enables the system to avoid categorization and keep gradual representations of both elements. Moreover, we propose a set of robot-centric 3D tool descriptors, and study their suitability for interaction scenarios, comparing also their performance against features derived from Deep Convolutional Neural Networks. Results show that the presented methods allow the robot to predict the effect of its tool use actions accurately, even for previously unseen tool and poses, and thereby to select the best action for a particular goal given a tool-pose.**

***Index Terms*—tool use; affordances; interaction learning; humanoid robot; iCub; 3D features.**

I. Introduction

In a tool affordance scenario, learning corresponds to finding the mapping between a set of features that describe tools and the effects that these tools are capable of achieving through actions on an object. In most previous studies, in order to ease learning, this mapping is achieved by clustering one of the elements (tools or effects) and classifying the features into the discovered categories. By contrast, the architecture proposed in this work maps the relationship 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 tool functional features and tool use effect measurements 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 representations of both elements on their corresponding SOMs. A diagram of the proposed learning architecture can be observed in Figure 1, whose components will be detailed in Section III-D.

[1] T. Mar, V. Tikhanoff, G. Metta and L. Natale are with the iCub Facility, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy (email: tanis.mar@iit.it,vadim.tikhanoff@iit.it, and lorenzo.natale@iit.it).

This work was supported by the European FP7 ICT project No. 270273 (Xperience), and project No. 288382 (POETICON++)

On the functional feature side, the representation applied to describe tools should enable the learning algorithm to generalize the affordances learned for a tool to similar ones. At the same time, it should take into account that the effect that an action with a tool can achieve depends also on how the tool is being grasped. For convenience, we use the term *tool-pose* to specify a tool in a given pose, following the nomenclature in [1]. For example, a rake oriented to the right would be a different tool-pose than the same rake oriented to the left.

In the current study, we apply therefore the Oriented Multi-Scale Extended Gaussian Image (OMS-EGI) feature set, proposed in [2]. This is, to the best of our knowledge, the only 3D descriptor which implicitly encodes information about how tools are being grasped, making it specially suited for tool representation in interaction scenarios. Moreover, we compare three different OMS-EGI parameter settings to assess which one enables more effective affordance learning. These settings represent tool-poses based only on their surface normal histograms, their occupation of the space within their bounding box, and on a balanced combination of both, respectively. Additionally, we compare their performance against state-of-the-art computer vision features, namely the feature vector extracted from the last convolutional layer of AlexNet Deep Convolutional Neural Network (DCNN) [3], obtained from images of the tool-poses that the robot held to perform exploration.

On the other side, the affordance of each tool-pose is represented as the effect – measured as a displacement on a target object– that the robot can achieve with that tool-pose for a set of actions. In particular, the action repertoire considered in this study consists in a drag action parametrized by the angle of the drag. The combination of the effects of all the actions in the repertoire into a single vector, which we will refer to as *affordance vector* as in [4], represents, for the robot, how well a tool-pose affords dragging an object in different directions. Despite its simplicity, this scenario was chosen so that with a relatively simple repertoire of actions, different tool-poses achieved distinct effects, and moreover, because it allows actions and effect computation to be done safely and automatically by the iCub robot.

Affordance knowledge, thus, 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. In the present study, both sets of data are obtained automatically by the robot on an exploration phase, and subsequently applied for learning in a self-supervised manner, without the need for any human labeling. Finally, the system's performance is evaluated on-

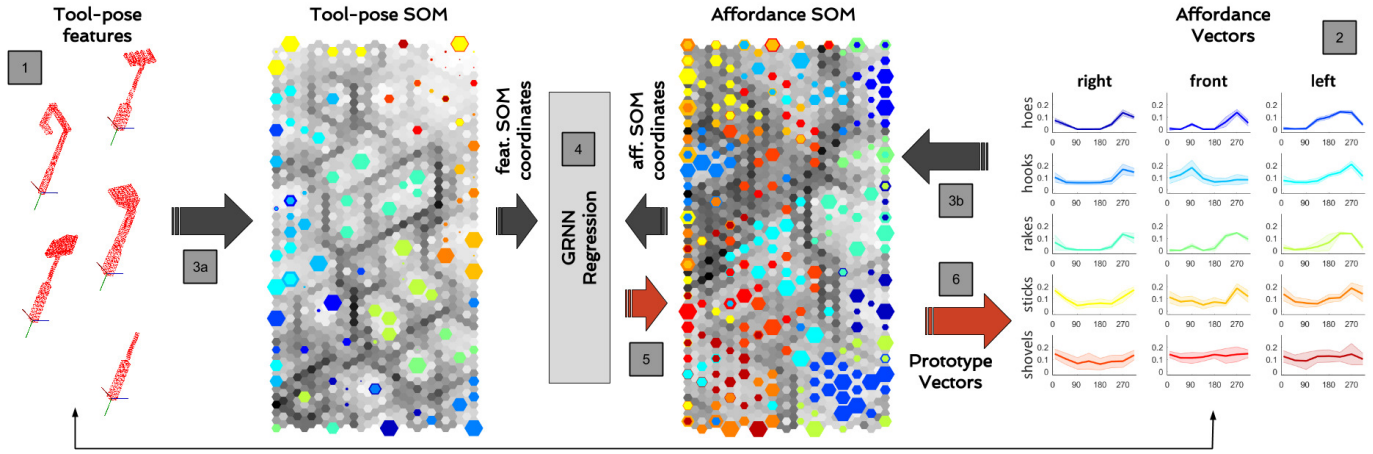


Fig. 1: Diagram of the proposed approach to discover, learn and predict tool-pose affordances. On the training phase (black arrows), a set of tool-pose features (1), and their corresponding affordance vectors (2) are available to the system from previous recording. Keeping the correspondence, tool-pose 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 as in the affordance vector graphs on the right of the diagram. This information was, however, not used for training.

line by applying the learned models to select the best action for a given task.

The methods presented in this study constitute a novel approach towards endowing humanoid robots with the skill to autonomously learn tool affordances and generalize the knowledge to similar tools. 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, each of them based on different information from the tool-pose, as well as against features derived from DCNN. On the other hand, we present a novel self-supervised learning architecture which is able to predict the effect of a set of actions with a given tool and grasp in function of its geometry, by applying a fine grained representation of tools and effects that avoids the need to categorize either, resulting in a higher predictive performance.

II. Related Work

The concept of affordances was introduced by the cognitive psychologist James J. Gibson in the late 70's as a central element of his theory of direct perception [5]. According to this theory, affordances are *"latent action possibilities available to the agent"* that it can perceive directly from the environment in order to interact with. The initial formalization of this idea, however, did not provide a complete explanation on how affordances are perceived or learned by agents. Affordance perception, indeed, has been matter of a long-standing debate in cognitive psychology [6], [7], [8], joined later by the neuroscience community with new hypothesis and evidence from brain studies [9], [10], [11]. See [12] for a comprehensive review. Affordance learning, on the other hand, was initially studied by Eleanor J. Gibson [13], [14] as a developmental strategy, and further discussed and extended by other cognitive and developmental psychologists [15], [16]. This concept soon grabbed the attention of the developmental robotics community too, as a framework to develop mechanisms to enable robots to learn about objects by means of interacting with them, thus grounded in their own embodiment [17].

The first steps towards affordance learning robots were taken by Fitzpatrick et al. in their pioneer work [18], which showed that a robot can learn affordances by observing the effect that certain actions produce on objects. In that study, rolling or sliding affordances were learned as histograms that represent the probability of displacing an object in a particular direction given a certain action. Although basic in its premises, this work was the first to implement a formalization of objects' affordances as the relationship between the elements of the tuple $\{object, action, effect\}$, which would become a commonplace in later works on robotic affordances. In [19], affordances were learned by populating a look-up table with the combination of action sequences and object labels that lead to successful *binding* between the objects. Geib et al. proposed a general way to learn affordances in robotics, based on Object-Action Complexes (OACs) [20]. OACs were defined

as $\{E,T,M\}$ triplets, where E identifies the robot action (called Execution specification), T represents the state change due to such action, that is, the effect, and M measures the success rate of E in achieving T over a given window of time [21]. The main drawback of these studies was that objects were identified by labels, so they did not allow generalization to new objects.

In order to overcome this issue, Montesano et al. [22] proposed a model in which simple object features (shape, color, etc) were connected through a Bayesian Network to the action and the observed outcomes of the interaction. This method was subsequently refined to include more robust features, better discretization [23], support multi-object interaction [24], and improved network structuring [25]. Another popular model was proposed by Sahin et al. [26], in which the object/entity was represented as a vector of raw features, and generalization was defined in terms of equivalence between entities or behaviors, if they produced the same effect. This method has been further improved to support the discovery of new effect categories [27], actualize behavior parameters to reach goals [28], self-structuring of the learning process [29], bootstrapping knowledge of simple affordances to more complex ones using intrinsic motivation [30], [31], and use of affordance knowledge for plan generation and execution [32].

Other methods for learning object affordances have also been explored. In [33], [34], for example, 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 [35] Semi-Markov Decision Processes were used to model the sequences of actions that lead to the activation of a desired affordance, whereas in [36], affordance learning is integrated within an Extended Classifier System reinforcement learning method, allowing the robot to learn about object's affordances and policies for goal-directed tasks simultaneously.

A few studies have also considered multi-object scenarios. Moldovan et al. [24] coupled Statistical Relational Learning methods with the Bayesian Network approach proposed in [22] to perform inference about more than one object. Fitchl et al. [37] studied how to represent the necessary spatial preconditions that enable certain 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 [32].

In the studies described so far the robot used its own manipulator to interact with objects, and observed the effect of its actions upon them. In order to differentiate multi-object affordances from tool affordances, we consider that tools correspond to objects or elements that the action is performed *with*, different of 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 objects, through an action on a third entity, enable the agent to achieve*.

Pioneer work on tool affordances was carried out by Stoytchev [38], [39], where for each tool, affordances were learned as a list containing the actions performed and the probability of success in moving an external object. In Tikhanoff et al. tool affordance models were learned by fitting curves to the effects measured after a series of pull actions, and coupled with a geometric reasoner in order to determine the feasibility of exploiting the learned affordances on a given scenario [40]. The main drawback of these studies is the lack of a representation that allows generalization among tools, so the learned knowledge can not be applied to tools outside the initial training set.

Jain and Inamura tackled this issue by defining a set of tool *functional features* based on geometric features (corners, bars, etc) that were linked to the effect achieved by the tool by means of a Bayesian Network [41], [42]. However, these functional features had to be annotated by hand. 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 [43], which were linked to actions' effects also through 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 [44]. In [4], we applied a larger set of candidate functional features from the 2D contour of the tool in such a way that the grasp position of the tool was also considered when learning tool's affordances. This approach was extended in [2] with the introduction of the Oriented Multi-Scale Extended Gaussian Image (OMS-EGI) feature vector, a holistic descriptor extracted from the tools' 3D models. Unlike previously proposed tool descriptors, the feature sets proposed in the last two studies are robot-centric insofar as they depend not only on the geometry of the tools, but also on how the robot is grasping them.

On the opposite side of the spectrum in terms of robot-centeredness, a popular approach to affordance learning in the recent years has been to apply state-of-the-art computer vision methods to predict human labeled affordances of objects. While the predicted affordances are meaningless to robots (for example, a robot without arms would not be able to use any tools, independently of their human labeled affordance), this approach can provide valuable insights into the kind of features that could be applied later to learn robotic affordances more effectively. For example, in [45], SOM-based sparse coding features from RGB-D data were applied to discriminate container from non-container objects. In [46], affordances of office objects were predicted by means of 3D geometry features computed from the pointcloud reconstruction of the scene. Deep learning methods have been applied in [47], where multi-scale CNNs were trained at different resolutions with RGB-D data from an affordance database. Also, Myers et al. proposed a part based affordance detection system based on 3D features extracted from a superpixel segmentation from objects [48]. 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 [49] learned the features from RGB-D images by training end-to-end a Deep Convolutional Neural Network within an expectation maximization framework, which enabled weakly supervised learning, while in [50], they trained an encoder-decoder DCNN with HHA features (horizontal disparity, height and angle between pixel normals and inferred gravity). Full 3D models were used in Schoeler et al., who proposed global part-based descriptors obtained as graphs that contain the part-signatures and their pose-relations in order to predict the function of a set of tool-like 3D models [51], as well as in [52], where Abelha et al. proposed to estimate 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.

Besides the cited papers, there are many other proposed features that have not been attempted within an affordance learning scenario, but that have descriptive characteristics that could also potentially 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 [53], [54] as well as unsupervised or weakly supervised feature learning [55], [56], 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 are the DESIRE feature [57] or the more recent Global Structure Histogram [58]. Other local 3D features, such as the ones proposed in [59], [60] could enable end-to-end 3D feature learning for affordance applications.

Readers interested in a more comprehensive review on affordances in robotics, with focus also in studies from psychology and neuroscience, should refer to the recent review by Jamone et al. [61].

III. Materials and Methods

A. Robotic Platform

All the experiments presented in this paper were carried out using the iCub humanoid robot and its simulator. The iCub is a full body humanoid robot with 53 Degrees of Freedom (DoF) [62]. In the current experiment we only made use of the upper body, which comprises 41 DoF, including head, arms and torso. Binocular vision is provided by the cameras mounted in the robot's eyes, with which depth can be estimated from disparity [63]. The iCub simulator provides a compatible model of the robot that allows simulation of rigid body dynamics and collision detection by making use of ODE (Open Dynamic Engine) [64].

The iCub software is structured as modules that communicate with each other using YARP middleware, which enables multi-machine and multi-platform integration [65]. Modules provide specific functionalities, and work together to achieve desired behaviors on the iCub. All the tool 3D models that were used for feature extraction in the experiments were modeled using Trimble's SketchUp software [66], and transformed into pointclouds using the Point Cloud Library [67], which was also used for 3D processing and visualization. Experimental data analysis, including visualization and learning, was implemented in MATLAB, employing the third party SOM toolbox for dimensionality reduction and data visualization [68], 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.

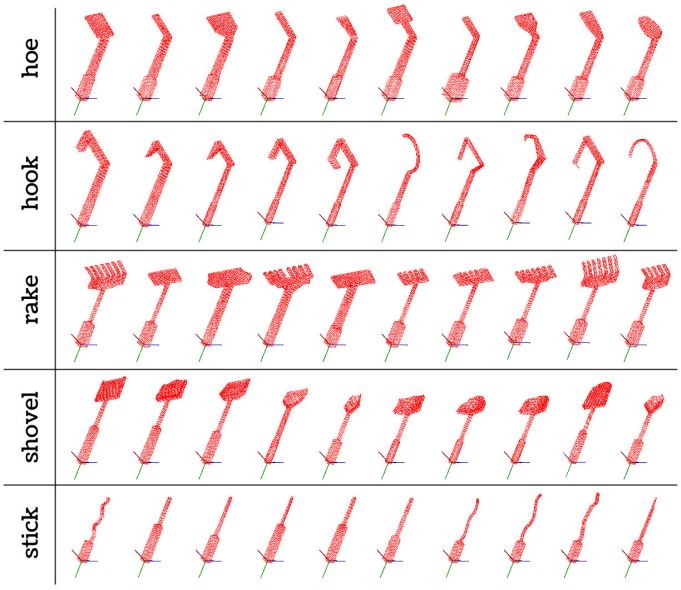
All the code used in the present study is available at `www.github.com/robotology/tool-affordances`.

B. Experimental setup

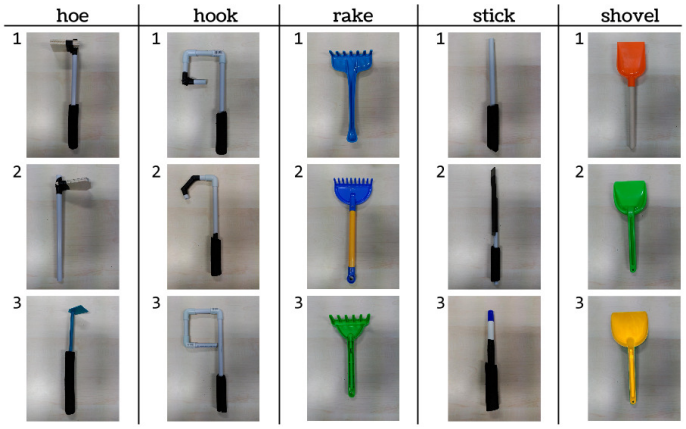
The experiment performed consisted of three phases: data gathering, training and testing. In the first phase, the iCub recorded the drag affordances, represented by their affordance vectors, of a set of tools in different orientations. In particular, 50 different tools were considered in simulation and 15 in the real setup, divided in 5 categories for visualization and clarity: rake, hoe, hook, shovel and stick (displayed in Figure 2). Each of them was held by the robot in 3 possible orientations: right, front, left, depending on the grasp parameter φ, described in Figure 3a. Thus, affordance vectors were recorded for 150 distinct tool-poses in simulation and 45 on the real robot. Moreover, for each tool-pose, in simulation and real setup, 4 recording trials were performed; each trial consisted in the execution of the drag action in all the considered directions (see Figure 3b), by means of which an affordance vector was recorded per trial.

Each trial begun by placing a tool in the robot's hand, for which a different procedure was followed depending on the setup. In the real setup, the iCub opened its hand in a receiving position and a tool was handed by the experimenter. After grasping the tool, the iCub automatically detected the tool-pose it had been given in the following way: First, the tool label was recognized using the deep learning implementation presented in [69], pre-trained with our set of tools. This method uses a linear classifier whose input are the features obtained from the FP7 layer of AlexNet [3], which were recorded here in order to compare their performance against the ones proposed in this paper in the task of affordance prediction. Once the tool was recognized, its corresponding pointcloud model was automatically loaded in the *canonical pose* (see Figure 5). Finally, its orientation was estimated by aligning the *canonical* pointcloud model to a single partial 3D reconstruction of the real tool in the robot's hand, obtained with the robot's stereo-vision. In simulation, by contrast, the tool name and its orientation were given to the simulator. Thus, the tool CAD model was loaded and attached to the simulated iCub's hand based on the given grasp parameter φ.

In either case, the canonical pointcloud model of the grasped tool was rotated according to the –estimated or given– grasp parameter φ in order to obtain the *oriented pointcloud model*.

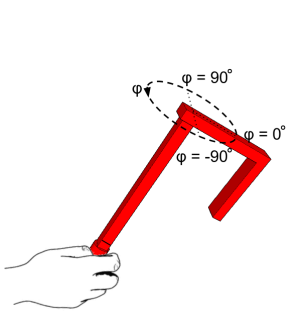


(a) Tools in simulation.

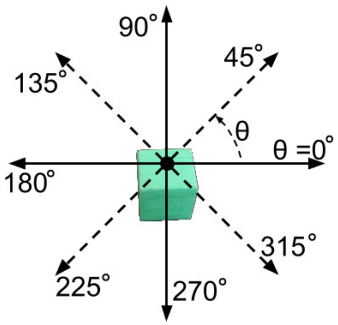


(b) Tools in real setup.

Fig. 2: 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.



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



(b) Diagram of the parametrized drag action. 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.

Fig. 3: Parameters controlling tool-pose and interaction: (a) Grasp and (b) Action.

This term refers to the pointcloud model of the grasped tool whose coordinates match the position of the actual tool with respect to the robot's hand reference frame. The oriented pointcloud model of every tool-pose considered was subsequently used for two purposes. On the one hand, it was sent to the processing modules for 3D feature extraction (detailed in Section III-C). 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.

Once the tool was grasped and the robot's end-effector successfully extended to the tip of the tool-pose, the robot performed a series of exploratory actions in order to discover the tool-pose's drag affordance. Specifically, a 15 cm radial drag action was executed upon a small target object along directions at intervals of 90 degrees on the real robot and 45 in simulation (resulting in 4 actions per trial in the real robot and 8 in simulation, as illustrated in Figure 3b).

The target object was a small cube of 6 cm in side placed before each action execution at a spot on a table randomly chosen at 40 $\pm$ 4 cm in front of the iCub and 10 $\pm$ 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 [70]), 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).

On each action execution, the robot first positioned the tooltip above the target object and then lowered it to the table plane, in order to avoid pushing the object when reaching for it. Then, the drag was executed in the given direction, and subsequently, its effect computed as the Euclidean distance that the object had been displaced on the table's plane. An image of the iCub reaching for the target cube in simulation and the real setup can be observed in Figure 4. As mentioned above, this experimental scenario was devised so that actions were safe and the effect could be computed automatically, as well as so that with a relatively simple repertoire of actions, different tool-poses could achieve distinct effects.

The total duration of the data gathering phase was of about 15 hours in simulation and in the real setup, estimating roughly an average of 10 seconds per action execution in simulation and 20 in the real setup (due to the longer time required to estimate the target object's final position and to reposition it after each action), plus some extra time to load/grasp and

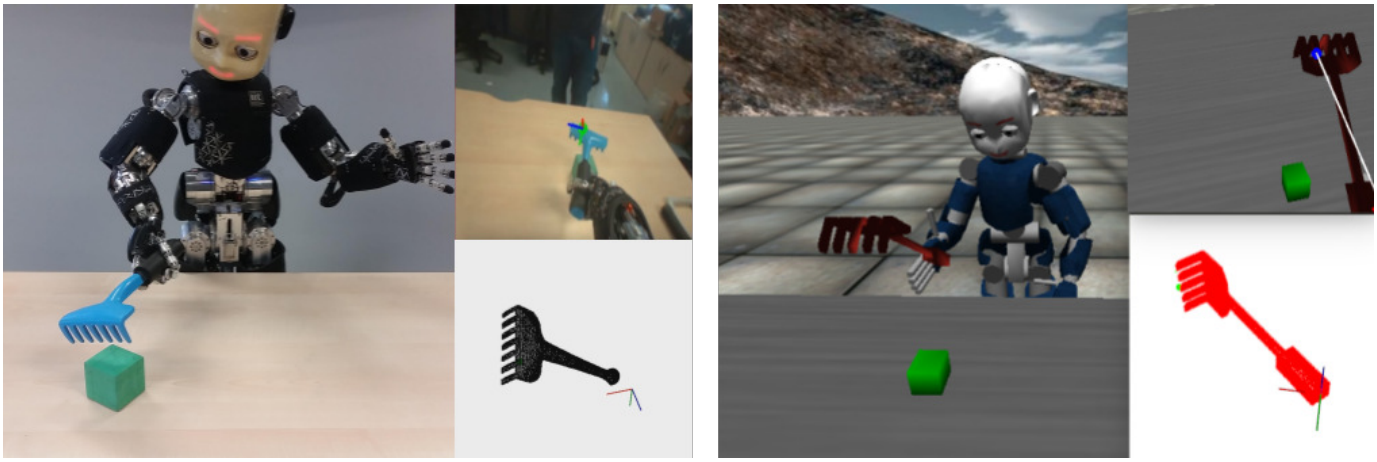


Fig. 4: Experimental setups in the real robot (left) and simulation (right). The frames on the upper right of the robot image show its perspective on the action, with the superimposed end-effector extension to the tool's tip. On the lower right side, the corresponding oriented pointcloud model is shown, whereby the displayed coordinate origin represents the hand reference frame.

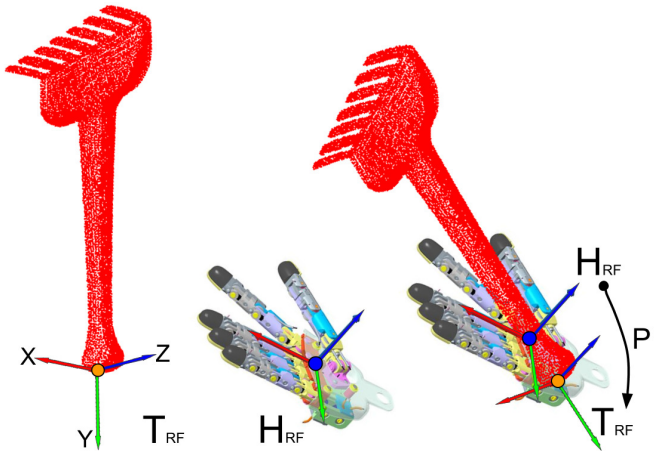


Fig. 5: Representation of the tool and hand reference frames. The hand reference frame ($< H >$, center) is defined by the robot kinematics. The tool's reference frame ($< T >$, left) is set so that the origin is at the lowest point on the tool's model, the X axis points in the direction of the tool effector, and the tool's handle goes in the direction of -Y. This way, if the tool is grasped with its effector looking to the front and its handle upwards (along the direction of the extended thumb), $< H >$ and $< T >$ coincide. This is therefore considered to be the tool's *canonical pose*, $< T >^{can}$, and the pointcloud model is thus the *canonical* one. The oriented pointcloud model corresponds to the pointcloud representation of the tool which matches the position of the actual tool in the hand, expressed with respect to the reference frame of the hand, i.e. $< H >$ (right).

find the pose of each tool. On the second phase, all the data recorded by the robot –tool-poses' functional features and corresponding affordance vectors– were used to train the system to predict, given a tool-pose, the effect of all the actions in the repertoire, as detailed below in Section III-D. On the test phase, the generalization capabilities of the trained system were evaluated with two different procedures. On the first one, employed to assess the general prediction capabilities of the system, the system's performance was assessed by comparing the predictions returned by the model to the previously recorded affordance vectors. On the second one, the iCub predicted the affordances of tool-poses not seen during training, and used these predictions to select the best actions for a given task, as described in Section III-E.

C. 3D features for tool-pose representation in interactive scenarios.

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 [18] and [19]), will not be able to generalize the learned affordances to any object outside the initial training set. On the other hand, care must be taken when defining the representation, because it can quickly lead to feature spaces too large to be explored, especially on a real robotic setup.

Moreover, most robotic tool affordance studies represent tools as external elements, independent of the robot which handles them. However, we believe that complete understanding of tool affordances can only be achieved if we consider the way in which tools are being grasped. Therefore, on the present study we applied the Oriented Multi-Scale Extended Gaussian Images (OMS-EGI) descriptor, a holistic descriptor specifically devised to represent grasped radial tools in interaction scenarios, proposed in [2]. OMS-EGI encapsulates

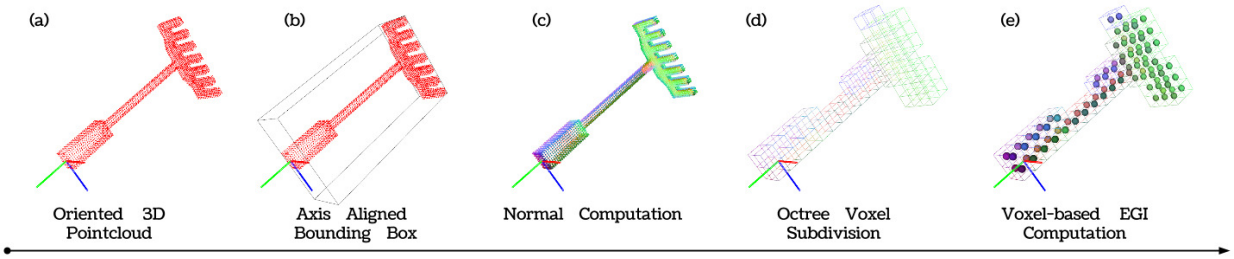


Fig. 6: OMS-EGI Computation Steps: First, the 3D pointcloud model of the tool grasped by the robot is loaded from memory and rotated to obtain the oriented pointcloud model (a). Then, the oriented pointcloud model's Axis-Aligned Bounding Box (AABB) is computed w.r.t. the hand reference frame axes (b). 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.

in a compact way the geometrical properties 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 brief, 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 6.

The reason why this descriptor is able to incorporate information about how the tool is being grasped is that it is computed from the tool-pose's oriented pointcloud model, which represents how the robot is actually holding the tool. Moreover, the voxels from which the surface histograms are obtained are computed as iterative subdivisions of the *Axis-Aligned* Bounding Box, so their concatenation order and orientation depends on the robot's hand reference frame, not on how the tool is oriented. Therefore, the same tool in different orientations will produce different OMS-EGI vectors.

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 following parameters:

- 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.
- 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 is 8^l (thus the name "octant"). N represents thus the resolution at which the voxel-based EGIs will be computed, by controlling the number and size of these voxels.

Therefore, by setting different pairs of parameters, the OMS-EGI descriptor allows us to study whether affordances are better learned based on the tool-pose's 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). We conducted such an evaluation by comparing the predictive performance of the following 3 settings:

- **Balanced information (BALAN)**: Setting $N = 2$ and $D = 2$, the feature vector corresponds to a *balanced* OMS-EGI, as applied in [2], where both surface and spatial information are represented. The length of the OMS-EGI vector with BALAN settings is of $L_{eff}^{BALAN} = 296$.
- **Spatial information (OCCUP)**: If $N = 1$, all normals in each voxel are assigned to the same bin irrespective of their orientation, and therefore the full histogram 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 an 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; : $L_{eff}^{OCCUP} = 293$.
- **Surface information (EGI)**: When $D = 0$, the only voxel considered corresponds to the AABB of the oriented pointcloud model, without further subdivisions. Therefore, the OMS-EGI is equivalent to the original EGI of the object [71], 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 in a similar range of the other settings; $L_{eff}^{EGI} = 216$.

Algorithm 1 OMS-EGI Feature Extraction

omsegi = compute_omsegi(model_or, Depth, NumBins), *where*
model_or = oriented pointcloud model,
***initialization*:**
    AABB ← *findAABB(model_or)*
    normals ← *compute_normals(model_or)*
    oms-egi ← *[]*
***loop*:**
for *level = 0* → *Depth* **do**
    voxel_list ← *compute_voxel_grid(AABB, level)*
    for all *v* ∈ *voxel_list* **do**
        if *is_empty(v)* **then**
            oms-egi ← concatenate(*oms-egi*, *(0 0 ... 0)*)
        else
            norm_hist ← *comp_norm_hist(normals, v, NumBins)*
            oms-egi ← concatenate(*oms-egi*, *norm_hist*)
        end if
    end for ▷ End voxel loop
end for ▷ End depth level loop

D. 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. 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 ([39], [27], [33], [4]), or by predefining effect categories to which the observed effects are assigned [72], [25]. Similarly, it is a common practice in studies where objects or tools are represented by features to cluster them before further processing [23], [2]. 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 $X \in \mathbb{R}^{L_{eff}}$, 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 (see Section III-B). L_{eff} is the length of the tool-pose feature vector and K the number of actions considered, which determines the length of the affordance vector. The proposed architecture enables learning the mapping between these two spaces, $f : \mathbb{R}^{L_{eff}} \rightarrow \mathbb{R}^K$, 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 dimensionality reduction in the original spaces limits the complexity of the subsequent supervised regression problem by reducing the original problem of finding $f : \mathbb{R}^{L_{eff}} \rightarrow \mathbb{R}^K$ to $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, being $K > 2$ and $L_{eff} >> 2$. Thereby, it helps preventing the numerical errors due to overfitting that commonly occur in classifiers when the dimensionality of the input space is larger or of the same order as the number of available samples.

Moreover, as dimensionality reduction is an unsupervised procedure, data augmentation techniques can be applied to obtain an enough number of unlabeled samples with which to perform it. In the present scenario, data augmentation was achieved by extracting extra tool-pose features from slight rotations of the oriented pointcloud models recorded in the experiment. These were "unlabeled" in the sense that they did not have a corresponding affordance vector to which they could be mapped. This allowed the tool-pose SOM to be trained with a number of samples much larger than the dimensionality of the input space.

SOMs were chosen over other dimensionality reduction methods because they allow to map new data points incrementally (unlike, for example, t-SNE [73]), and they maintain to a great extent the topology of the data in the original high-dimensional space [74]. 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. This characteristic provides a mechanism to retrieve predictions in the original affordance vector space from the lower dimensionality regression results. In that sense, prototype vectors are analogous to cluster centroids (used in [4]), but unlike them, which are by definition distinct from each other, prototype vectors can gradually vary 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 [75] for multimodal object learning, and [33], [34] for object affordance learning, but applying very different data modalities, and learning and prediction methods.

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 [76]. 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-

[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.

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, that is, the expected effect of executing any of the actions in the repertoire with the given tool-pose. In order to achieve this, the first step was to extract the tool-pose feature vector $x \in X$ that represents the tool-pose being held 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}, 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 $\hat{y}_{SOM}$.

E. Prediction based action selection

The methods described above enable the robot to predict the affordance vector of any tool-pose, i.e., the expected effect for any of the actions in the repertoire. Therefore, 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 with any of the considered tool-poses, we devised a secondary test experiment, which we refer to as Action Selection experiment.

In particular, the task chosen was to achieve the maximum possible displacement on the target object, given a certain tool-pose. Therefore, in this experiment the robot received a tool in a certain pose –loaded in simulation and handed by the experimenter to the real robot–, predicted its affordance vector based on its OMS-EGI features, as described in Section III-D, and executed the drag with the angle parameter predicted to generate a larger displacement. After the action execution, the actual achieved effect was measured in order to evaluate the success of the given task. In order to provide a good assessment of the generalization capabilities of the proposed learning architecture, we applied leave-one-out data separation scheme, which meant that every time a tool was tested, it had not been used at any step of the training process, in any of its poses.

IV. Results

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 2). Each tool was used by the robot in 3 different orientations, right, front and left, making up to 45 considered tool-poses on the real robot and 150 in simulation.

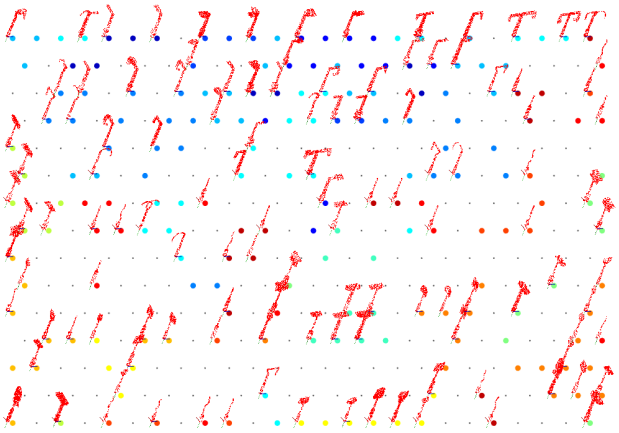
Interaction data was gathered by performing 4 experimental trials with each tool-pose, where each trial consisted of a cycle of 4 drag actions in the real robot and 8 in simulation, performed as described in Figure 3b. As a result, 180 trials corresponding to 720 actions were performed on the real setup, and 600 trials corresponding to 4800 actions in simulation. On each of these trials, the recorded data consisted of the affordance vector representing the recorded effects of the actions performed in each action cycle, and the OMS-EGI feature vectors used to describe the tool-pose being used to perform those actions; EGI, BALAN, OCCUP, and in the real setup, the deep learned features. In order to have more data to improve the unsupervised training of the tool-pose SOM, the number of samples per tool-pose of each of these OMS-EGI variants was increased by 30 by applying the "data augmentation" method described in Section III-D. This number was selected in order to have a number of OMS-EGI samples considerably larger than its dimension 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.

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 by randomly selecting 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.

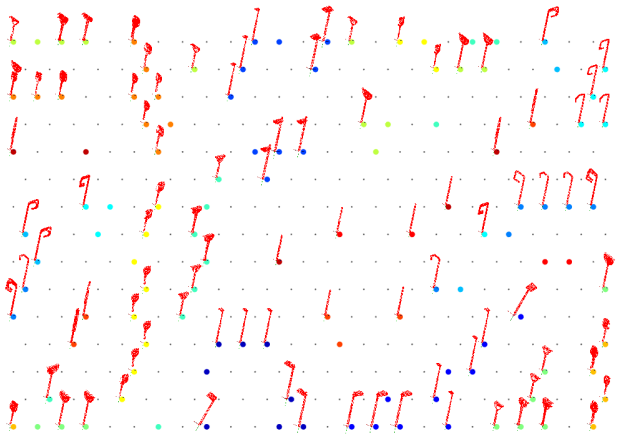
B. SOM-based unsupervised dimensionality reduction

As described in Section III-D, 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 vectors not used for testing the affordance prediction, that is, all the OMS-EGI vectors corresponding to the affordance vectors on the train set, plus all the ones obtained through data augmentation. The results of this mapping process can be ob-

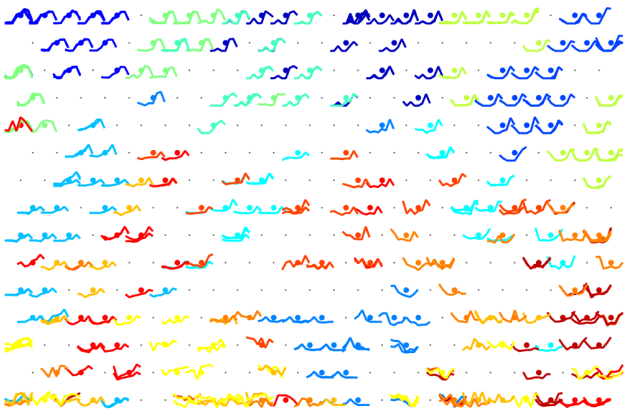


(a) Trained **tool-pose SOM**, from **simulation** data.

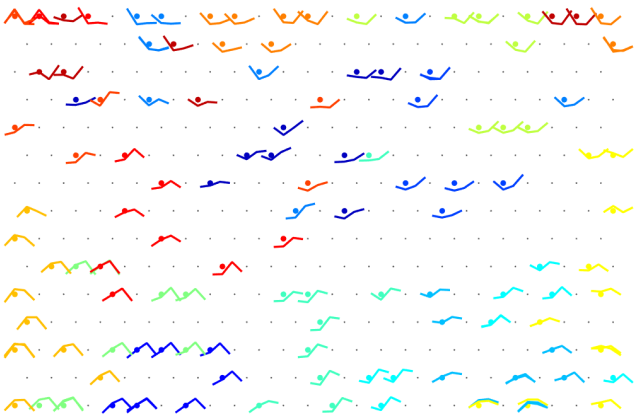


(b) Trained **tool-pose SOM**, from **real setup** data.

Fig. 7: Trained tool-pose SOMs with BALAN parameter settings in simulation (a) and real setup (b). On the underlying hexagonal grid, each dot represents a neuron on the SOMs. The red figures on top of some neurons represent the model of the tool-pose whose OMS-EGI feature activated that neuron. The color of the dot codes the tool type and pose, as in Figure 1. Grey neurons correspond to neurons which were not activated by any tool-pose's OMS-EGI vector in the training set. To avoid cluttering, only a fraction of all the tool-poses generated with the data augmentation process are shown.



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



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

Fig. 8: Trained affordance SOMs from data gathered in simulation (a) and on the real setup (b). On the underlying hexagonal grid, each dot represents a neuron on the SOMs. The color vectors on top of some neurons represent the affordance vector that activated those neurons, where the colors encode the tool type and pose that activated that neuron, as in Figure 1. Neurons represented by a gray dot correspond to neurons which were not activated by any affordance vector in the training set.

served in Figure 7. As can be observed, both tool-pose SOMs show similar tool-poses clustered together. Interestingly, tools from the same category tend to be clustered closely if they have the same orientation, while similar tools with different orientations are in most cases further apart. This results attest that the OMS-EGI descriptor preserves relevant information of the tool's geometry and orientation, and that the dimensionality reduction step is able to retain it, rendering the tool-pose SOM coordinates a reliable representation of the considered tool-poses.

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 8. In these maps we can observe how most tools of the same type in the same orientation (indicated by color) generated similar affordance vectors, which supports our hypothesis that similar tool-poses have similar affordances, although some exceptions can be observed (some yellow-orange and reddish affordance vectors, corresponding to sticks and shovels, are quite spread). Moreover, these maps explicitly display the fine-grained representation of affordances mentioned above, which can be observed in the gradual variation of the affordance vectors throughout the SOM.

C. Prediction of tool-pose affordances

The performance of the proposed method for affordance prediction was evaluated by comparing the predicted affordance vectors $\hat{Y}$ for the test set of tool-poses with the affordance vectors Y previously recorded for those tool-poses, 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. Accordingly, the learning performance was assessed by means of the percentage of improvement (PI), which indicates how much better the trained system performed when compared to the baseline one, so that if nothing was learned –prediction error had not improved– PI would be 0% while if error was reduced to 0, $PI = 100\%$. Formally:

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

where N is the number of test trials, and

$$PI = 100\Big(\frac{MAE_{BL} - MAE}{MAE_{BL}}\Big) \tag{2}$$

Table I displays the prediction error in each of the evaluation scenarios with the proposed 3D features, 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 in the 1OUT data separation mode (15 in the real setup and 50 in simulation). For comparison, Table II 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). In Figure 9 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.

D. Action Selection

The last evaluation step consisted in applying the learned models to select, given a tool-pose not observed during training, the best action for the task of achieving maximum displacement of the target object, as explained in Section III-E. 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 models trained with the 1OUT data separation scheme, so that the data corresponding to the tested tool had never been used to train 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 I that in simulation the models learned using OCCUP parameter settings perform slightly better than the rest for the 1OUT data separation scheme, in terms of the achieved PI. 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 OMS-EGI parameter settings. Therefore, we chose to perform the action selection test applying the models trained with BALAN parameter settings, which provide more consistent performance among both scenarios. For each test tool-pose, thus, the robot obtained a prediction of its affordance vector from the model trained using 1OUT data separation and BALAN OMS-EGI settings, and executed the action with maximum expected displacement, as described in Section III-E. This procedure was run twice for each tool-pose, yielding the results that can be observed in Figure 10.

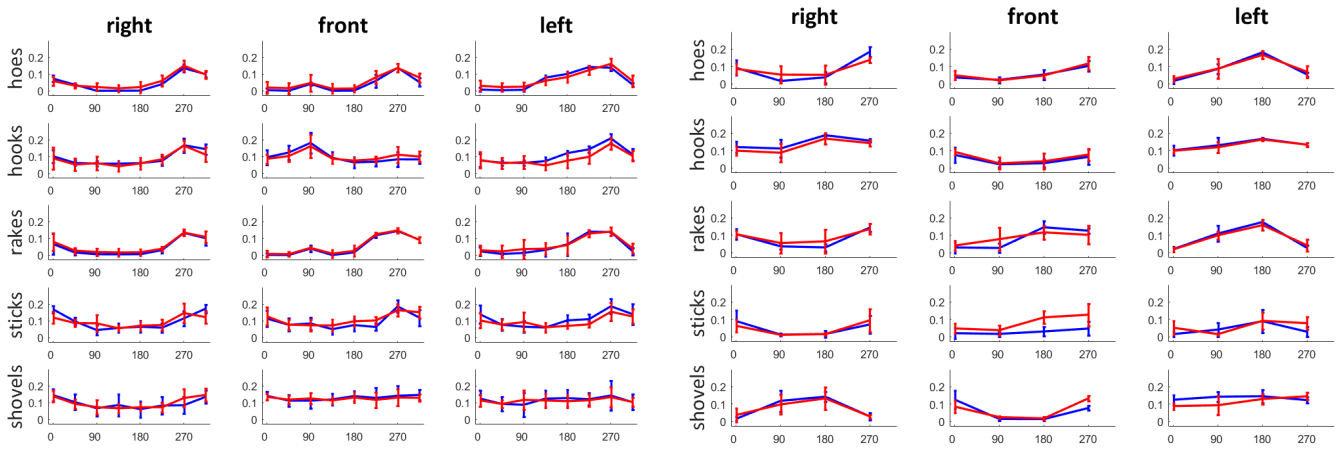
Based on these results, the degree of accomplishment of the given task was assessed by comparing the achieved effects against an action selection baseline, which was defined for each tool-pose as the median effect among all effects recorded for that tool-pose during the data gathering part of the experiment. In principle, if actions had been selected at random, the achieved effect would be over this baseline 50% of the times. Figure 10 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 [25], 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:

$$G = (ST - (UT \cdot UO))/T \tag{3}$$

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

V. Discussion and Future Work

In the current study we presented a set of methods that allows 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



(a) Simulation

(b) Real

Fig. 9: 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.

SIM	**BALAN**			**EGI**			**OCCUP**		
	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$
RAND	2.7±0.19	5.02±0.18	43.4	2.77±0.24	5.01±0.2	44.6	2.60±0.19	4.99±0.23	47.7
1OUT	3.32±1.7	5.12±0.97	35.7	3.31±1.84	5.07±0.9	34.6	3.12±1.87	4.97±0.98	37.2
REAL	**BALAN**			**EGI**			**OCCUP**		
	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$
RAND	2.46±0.23	5.75±0.35	57.3	2.4±0.26	5.67±0.38	57.7	2.25±0.19	5.74±0.33	60.8
1OUT	3.58±0.81	5.81±1.54	38.3	3.99±1.3	5.71±1.3	30.1	4.15±1.87	5.51±1.17	24.6

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

REAL	**AlexNet FP7 feats**		
	MAE	MAE_{BL}	$PI(\%)$
RAND	3.18	5.71	45.6
1OUT	5.53	5.76	4.01

TABLE II: Prediction results obtained with the DL features.

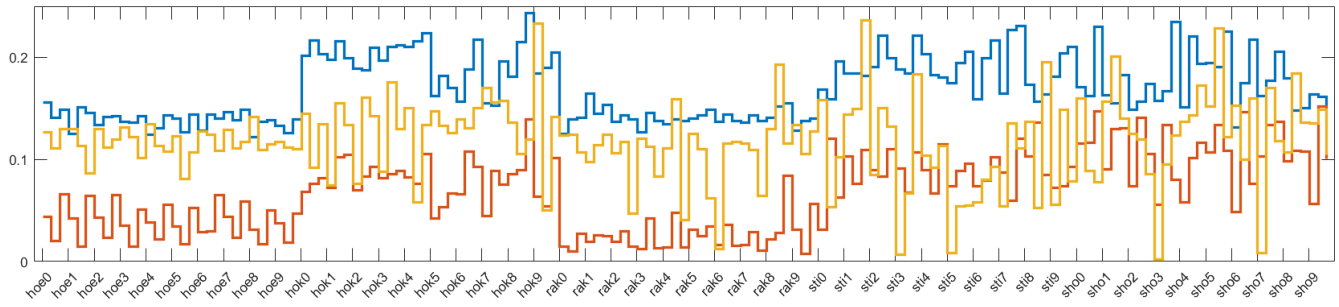
SIM	hoes	hooks	rakes	sticks	shovels	**Total**
$S(\%)$	100	90	96.7	50	66.7	80.67
G	100	63.3	86.7	0	40	56.7
REAL	hoes	hooks	rakes	sticks	shovels	**Total**
$S(\%)$	100	88.9	88.9	77.8	88.9	88.9
G	100	77.8	77.8	44.4	77.8	75.6

TABLE III: Action selection performance results, in simulation and the real setup.

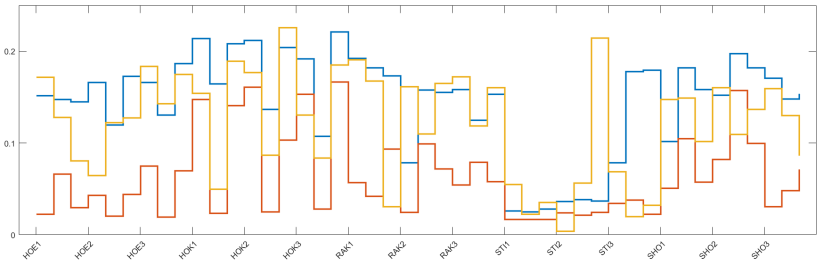
predict tool affordances based on their geometry, and to successfully use this knowledge to select valid actions to achieve a displacement task on the target object.

The prediction results 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 can be observed in the substantial reduction of the prediction error, reflected in the PI values displayed in Table I, as well as in the close match between the predicted and the recorded affordance vectors illustrated in Figure 9.

Observing the individual results for each of the OMS-EGI parameter settings in the different scenarios, it is noteworthy the degradation of the 1OUT results obtained with the OCCUP descriptor on the real setup when compared with simulation, which does not happen for the other OMS-EGI settings. 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



(a) Action selection experiment results, in simulation.



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

Fig. 10: 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.

oriented pointclouds within their bounding box. Therefore, all 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 this 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.

Taking into account this effect, the BALAN OMS-EGI parameter setting provides the most reliable information to predict tool-pose affordances. We believe this is due to the fact that it can better benefit from the data augmentation technique to improve the unsupervised mapping, while at the same time integrating relevant information about the model's surface directions and its occupation of space (relative to the robot's hand).

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 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 10 and Table III, 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, as shown by the large percentage of effects generated over the baseline. Yet, in some

cases 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 generated unpredictable effects, which prevented proper learning of these tool's affordances. 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 chances that the object would "jump" a few centimeters away in any direction. This situation happened with tool-poses where the tooltip was situated in the same vertical axis as the handle – sticks, hooks oriented to the front, and some shovels oriented to either side – and explains the poor results obtained with these tool-poses.

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. [25], 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, in our study the overall accuracy is nevertheless around 6.5% higher. 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 [25], 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 representation and generalization among tools came at the expense of necessary simplification in the representation of the rest of the elements present in the affordance and in the implementation of the methods used by the robot to explore its environment, which reduce the autonomy of the presented method.

Concerning the representation of the affordance elements, we acknowledge that the action repertoire and possible grasps, as well as the way in which the effect is measured, are quite limited. While the presented learning methods could in principle cope with higher dimensionality in inputs and outputs thanks to the dimensionality reduction step, increasing the complexity of these elements, specially the action repertoire, could easily lead to search spaces impossible to explore sufficiently on a real robotic setup unless other constraints are in place. In a similar way, we have only considered the location of the target object, 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 the previous literature [40], we assume that it can or has been learned in previous stages of the robot development.

Regarding the method's autonomy, we claim that the presented method is self-supervised in the sense that all the elements required to learn the affordance are obtained automatically by the iCub, and their relationship learned without any need of human labeling, which renders the learned knowledge meaningful for the iCub itself. This approach constrains the type of affordances that the robot can learn to those that it can safely generate, and automatically identify, which would be a much harder challenge for other actions such as hammering, cutting or scooping. This is in contrast to computer vision affordance learning methods, whose goal is to predict a set of human labeled affordances, and therefore can in principle learn any affordance a human can generate, but would be meaningless for any robot, unless otherwise transferred or verified (as in [50]).

Nevertheless, there are some aspects that substantially limit the autonomy of the presented method, understood as the need for human intervention in order to learn affordances. First, the exploration strategy consisted on the repetition of a set of predefined actions, and occurred off-line (before learning). More efficient methods exist in the literature, based on active exploration and even intrinsic motivation [77], [78], but we decided to apply exhaustive exploration in order to obtain full affordance vectors, on which the presented learning system is based. Also, we assume that full 3D models of all the considered tools are available, which would not be the case if encountering a new tool. 3D reconstruction techniques have been proposed in the literature from depth or image data [79], [80], [81], both available in the iCub. However, the low resolution of the cameras and their variable baseline make the implementation of these methods on the iCub a non trivial enterprise, out of the scope of this study. Work in this direction is already ongoing, which should in the future allow the iCub to reconstruct 3D models of the available tools online.

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 [69], 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. Moreover, at present time the iCub is unable to autonomously grasp a tool from a table with a radial tool grasp and a desired orientation, so even if it was

able to choose a tool, it should be handed correctly by the experimented. Nevertheless, a lot of work is being put into improving the grasping capabilities of the iCub, and not only, so we hope that in a near future this problem will be overcome.

VI. Conclusions

In this paper, we presented a method to learn and generalize tool affordances based on their geometry and the way in which they are grasped, implemented and validated in the iCub robot and its simulation on a large dataset of tools. The proposed learning architecture performs dimensionality reduction of the tool and affordance representations by mapping them on 2D Self-Organizing Maps, whose coordinates are in turn mapped by means of a GRNN regressor. Affordances are represented in terms of affordance vectors, which merge together action and effect. Tools, instead, are represented with three different 3D descriptors based on the tools' 3D information, which encapsulate the geometrical properties of a tool relative to the way in which it is grasped. We also compare the predictive performance of the proposed 3D descriptors against features derived from deep convolutional neural networks, showing that our description provides better generalization performance for interaction scenarios. Finally, the best tool-pose descriptor in terms of prediction is chosen to learn a model which the iCub uses to select the best action for a given task of displacing the target object. Results show that the proposed method outperforms recent studies with similar scenarios, and indeed allows the robot to select the best action for the given task with a high success rate.

References

[1] S. Brown and C. Sammut, "Tool Use Learning in Robots," in *Advances in Cognitive Systems*, 2011, pp. 58–65.

[2] T. Mar, V. Tikhanoff, G. Metta, and L. Natale, "Multi-model approach based on 3D functional features for tool affordance learning in robotics." in *Humanoids 2015*, Seoul, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[4] T. Mar, V. Tikhanoff, G. Metta, and L. Natale, "Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot," in *International Conference on Robotics and Automation*, 2015, pp. 3200 – 3206.

[5] J. J. Gibson, *The ecological approach to visual perception*. Psychology Press, 1979.

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

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

[8] A. Chemero, "An Outline of a Theory of Affordances," *Ecological Psychology*, vol. 15, no. 2, pp. 181–195, 2003.

[9] A. Murata, L. Fadiga, L. Fogassi, V. Gallese, V. Raos, and G. Rizzolatti, "Object representation in the ventral premotor cortex (area F5) of the monkey." *Journal of neurophysiology*, vol. 78, no. 4, pp. 2226–2230, 1997.

[10] M. Matelli and G. Luppino, "Parietofrontal circuits for action and space perception in the macaque monkey." *NeuroImage*, vol. 14, no. 1 Pt 2, pp. S27–32, 2001.

[11] E. Bartoli, L. Maffongelli, M. Jacono, and A. D'Ausilio, "Representing tools as hand movements: Early and somatotopic visuomotor transformations," *Neuropsychologia*, vol. 61, pp. 335–344, 2014.

[12] S. Thill, D. Caligiore, A. M. Borghi, T. Ziemke, and G. Baldassarre, "Theories and computational models of affordance and mirror systems: An integrative review," *Neuroscience and Biobehavioral Reviews*, vol. 37, no. 3, pp. 491–521, 2013.

[13] E. J. Gibson, *Principles of perceptual learning and development*. Prentice-Hall, 1969.

[14] E. J. Gibson and A. D. Pick, *An ecological approach to perceptual learning and development*. Oxford University Press, 2000.

[15] R. L. Goldstone, "Perceptual learning." *Annual review of psychology*, vol. 49, pp. 585–612, 1998.

[16] M. Viezzer and C. Nieywenhuis, "Learning affordance concepts: some seminal ideas," *International Joint Conference on Artificial Intelligence*, 2005.

[17] F. Guerin, N. Krüger, and D. Kraft, "A Survey of the Ontogeny of Tool Use: From Sensorimotor Experience to Planning," *IEEE Transactions on Autonomous Mental Development*, vol. 5, no. 1, pp. 18–45, 2013.

[18] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning About Objects Through Action - Initial Steps Towards Artificial Cognition," in *Procceedings of the IEEE Intrernational Conference on Robotics and Automation*, 2003, pp. 3140–3145.

[19] A. Stoytchev, "Toward Learning the Binding Affordances of Objects : A Behavior-Grounded Approach," in *AAAI Symposium on Developmental Robotics*, 2005, pp. 21–23.

[20] C. Geib, K. Mour, R. Petrick, N. Pugeault, M. Steedman, N. 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).*, 2006.

[21] N. Krüger, C. Geib, and J. Piater, "Object-Action Complexes: Grounded Abstractions of Sensorimotor Processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[22] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Modeling affordances using Bayesian networks," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 4102–4107.

[23] P. Osório, A. Bernardino, and R. Martinez-cantin, "Gaussian Mixture Models for Affordance Learning using Bayesian Networks," in *International Conference on Intelligent Robots and Systems*, 2010, pp. 1–6.

[24] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," *2012 IEEE International Conference on Robotics and Automation*, pp. 4373–4378, 2012.

[25] A. Gonçalves, J. Abrantes, G. Saponaro, L. Jamone, and A. 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)*, no. October, 2014, pp. 1–8.

[26] 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*, vol. 15, no. 4, pp. 447–472, 2007.

[27] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7-8, pp. 580–595, 2011.

[28] ——, "Going beyond the perception of affordances: Learning how to actualize them through behavioral parameters," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4768–4773.

[29] E. Ugur and J. Piater, "Emergent Structuring of Interdependent Affordance Learning Tasks," in *IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2014, pp. 481–486.

[30] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2014, pp. 468–473.

[31] E. Ugur and J. Piater, "Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection," *IEEE Transactions on Cognitive and Developmental Systems*, no. 99, pp. 1–13, 2016.

[32] ——, "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.

[33] 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*, 2010, pp. 5047–5054.

[34] B. Ridge, A. Leonardis, A. Ude, M. Denisa, and D. Skocaj, "Self-Supervised Online Learning of Basic Object Push Affordances," *International Journal of Advanced Robotic Systems*, vol. 12, no. 3, pp. 1–18, 2015.

[35] A. Glover, "Developing grounded representations for robots through the principles of sensorimotor coordination," Ph.D. dissertation, Queensland University of Technology, 2014.

[36] C. Wang, K. V. Hindriks, and R. Babuska, "Robot learning and use of affordances in goal-directed tasks," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2288–2294, 2013.

[37] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Krüger, and F. Guerin, "Learning Spatial Relationships From 3D Vision Using Histograms," *Icra*, 2014.

[38] A. Stoytchev, "Behavior-grounded representation of tool affordances," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April, pp. 3060–3065, 2005.

[39] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *2008 7th IEEE International Conference on Development and Learning*, 2008, pp. 91–96.

[40] V. Tikhanoff, U. Pattacini, I. S. Member, L. Natale, and G. Metta, "Exploring affordances and tool use on the iCub," in *Humanoids 2013*, 2013.

[41] R. Jain and T. Inamura, "Learning of Tool Affordances for autonomous tool manipulation," *2011 IEEE-SICE International Symposium on System Integration SII*, pp. 814–819, 2011.

[42] ——, "Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools," *Artificial Life and Robotics*, vol. 18, no. 1-2, pp. 95–103, 2013.

[43] A. Gonçalves, G. Saponaro, L. Jamone, and A. Bernardino, "Learning visual affordances of objects and tools through autonomous robot exploration," *2014 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2014*, no. May, pp. 128–133, 2014.

[44] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "Denoising Auto-encoders for Learning of Objects and Tools Affordances in Continuous Space," in *International Conference on Robotics and Automation*, 2016, pp. 1–6.

[45] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, "A Behavior-Grounded Approach to Forming Object Categories: Separating Containers from Non-Containers," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 1, pp. 54–69, 2012.

[46] D. I. Kim, "Semantic Labeling of 3D Point Clouds with Object Affordance for Robot Manipulation," in *International Conference on Robotics and Automation*, 2014, pp. 5578–5584.

[47] A. Roy and S. Todorovic, "A Multi-Scale CNN for Affordance Segmentation in RGB Images," in *European Conference on Computer Vision (ECCV2016)*, 2016, pp. 186–201.

[48] A. Myers, A. Kanazawa, C. Fermuller, and Y. Aloimonos, "Affordance of Object Parts from Geometric Features," in *International Conference on Robotics and Automation2*, 2015, pp. 5–6.

[49] A. Srikantha and J. Gall, "Weakly Supervised Learning of Affordances," *arXiv preprint*, vol. abs/1605.0, pp. 1–16, 2016.

[50] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting Object Affordances with Convolutional Neural Networks," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.*, 2016, pp. 2765–2770.

[51] M. Schoeler and F. Worgotter, "Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Autonomous Mental Development*, vol. pp, no. 99, pp. 1–1, 2015.

[52] P. Abelha, F. Guerin, and M. Schoeler, "A Model-Based Approach to Finding Substitute Tools in 3D Vision Data," in *International Conference on Robotics and Automation*. IEEE, may 2016, pp. 2471–2478.

[53] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[54] Y. Lecun, "Learning Invariant Feature Hierarchies," in *Computer VisionECCV 2012. Workshops and Demonstrations*, 2012, pp. 496–505.

[55] V. Nair and G. E. Hinton, "3D Object Recognition with Deep Belief Nets," in *Advances in Neural Information Processing Systems*, 2009, pp. 1339–1347.

[56] H. Lee, "Unsupervised Feature Learning Via Sparse Hierarchical Representations," Ph.D. dissertation, 2010.

[57] D. V. Vranić and D. Saupe, "3D model retrieval," in *Proc. Spring Conference on Computer Graphics and its Applications (SCCG2005)*, 2005, pp. 89–93.

[58] M. Madry, C. H. Ek, R. Detry, K. Hang, and D. Kragic, "Improving generalization for 3D object categorization with Global Structure Histograms," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1379–1386.

[59] Z. Wu and S. Song, "3D ShapeNets : A Deep Representation for Volumetric Shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, 2015, pp. 1–9.

[60] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and Multi-View CNNs for Object Classification on 3D Data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[61] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor, "Affordances in psychology, neuroscience and robotics: a survey," *IEEE Transactions on Cognitive and Developmental Systems*, no. August, pp. 1–1, 2016.

[62] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. 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*, vol. 23, no. 8-9, pp. 1125–34, 2010.

[63] 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*, 2014, pp. 1028–1035.

[64] 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.

[65] G. Metta, "Software implementation of the phylogenetic abilities specifically for the iCub & integration in the iCub Cognitive Architecture," Tech. Rep. 004370, 2006.

[66] Trimble, "3D modeling for everyone — SketchUp," 2016. [Online]. Available: www.sketchup.com

[67] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[68] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-Organizing Map in Matlab: the SOM Toolbox," in *Matlab DSP Conference*, 2000, pp. 35–40.

[69] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Teaching iCub to recognize objects using deep Convolutional Neural Networks," *Proceedings of The 4th Workshop on Machine Learning for Interactive Systems*, pp. 21–25, 2015.

[70] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[71] B. K. P. Horn, "EXTENDED GAUSSIAN IMAGES." *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1671–1686, 1984.

[72] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[73] L. van der Maaten, G. E. Hinton, L. van der Maaten, and G. E. Hinton, "Visualizing high-dimensional data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[74] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[75] J. Sinapov, T. Bergquist, C. Schenck, U. Ohiri, S. Griffith, and A. Stoytchev, "Proprioceptive and Auditory Feedback," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1250–1262, 2011.

[76] D. F. Specht, "A general regression neural network," *Neural Networks, IEEE Transactions on*, vol. 2, no. 6, pp. 568–576, 1991.

[77] P.-y. Oudeyer, V. V. Hafner, and F. Kaplan, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[78] A. Baranes and P. Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[79] D. Xu, J. Cai, T. J. Cham, P. Fu, and J. Zhang, "Kinect-Based Easy 3D Object Reconstruction," in *Pacific-Rim Conference on Multimedia*, 2012, pp. 476–483.
[80] C. Y. Ren, V. Prisacariu, D. Murray, and I. Reid, "STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1561–1568, 2013.
[81] Y. Zhang, G. M. Gibson, R. Hay, R. W. Bowman, M. J. Padgett, and M. P. Edgar, "A fast 3D reconstruction system with a low-cost camera accessory," *Scientific Reports*, vol. 5, pp. 1–7, 2015.

Tanis Mar is currently a senior research fellow at the iCub Facility at the Italian Institute of Technology. He received the M.Sc. degree in Telecommunication Engineering from the University of Zaragoza in 2009, the M.Sc. degree in Computational Neuroscience from the Bernstein Center - Berlin in 2013, and the PhD degree in Machine Learning for Robotics from the iCub Facility at the Italian Institute of Technology in 2017, on the topic of tool use and manipulation. He has also worked as a researcher at Fraunhofer IMPS in Dresden, Humboldt University in Berlin, ATR Laboratories in Kyoto and UKE in Hamburg, and participated in several European projects such as eSCMs, Xperience and POETICON++. His research interests focus on the mechanisms and applications of learning processes in natural and artificial agents, and include machine learning, artificial curiosity, sensorimotor integration, computer vision and developmental robotics.

Vadim Tikhanoff currently holds a Technologist position at the Italian Institute of Technology (IIT) working in the iCub Facility. He attained his Ph.D. in 2009 at the University of Plymouth UK, with a thesis on the Development of Cognitive Capabilities in Humanoid Robots and pursued his research as a Post-Doc at the Italian Institute of Technology (IIT). Vadim Tikhanoff has a background in Artificial Intelligence and Robotic Systems and is now focusing on the development of innovative techniques and approaches for the design of skills in a robot to interact with the surrounding physical world and manipulate objects in an efficient manner. He has authored numerous scientific articles including journals and book chapters in areas ranging across neural networks, language acquisition, cognitive systems and image processing. Dr. Tikhanoff served as the Program Chair of ICDL-Epirob 2014 and program committee of HAI 2017. Amongst other, he is also guest Associate Editor of the Humanoid Robotics code topic of the Frontiers in Robotics and AI.

Lorenzo Natale is presently a Researcher at the IIT, where he leads the Humanoid Sensing and Perception research group. He received his Ph.D. in Robotics in 2004 from the University of Genoa and he was later postdoctoral researcher at the MIT-CSAIL. Lorenzo Natale has contributed to the development of various humanoid robots, and he was one of the main developers of the iCub humanoid robot. His research interests range from sensorimotor learning and artificial perception to software architectures for robotics. He has authored more than 100 publications in peer-reviewed journals and conferences. Dr. Natale served as the Program Chair of ICDL-Epirob 2014 and HAI 2017. He is an Associate Editor of IEEE ROBOTICS AND AUTOMATION LETTERS, International Journal of Humanoid Robotics, and the Humanoid Robotics specialty of Frontiers in Robotics and AI.