Self-supervised learning of tool affordances from 3D tool representation through parallel SOM mapping

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

***Abstract*— Future humanoid robots will be expected to carry out a wide range of tasks for which they had not been originally equipped by learning new skills and adapting to their environment. A crucial requirement towards that goal is to be able to take advantage of external elements as tools to perform tasks for which their own manipulators are insufficient; the ability to autonomously learn how to use tools will render robots far more versatile and simpler to design. Motivated by this prospect, this paper proposes and evaluates an approach to allow robots to learn tool affordances based on their 3D geometry.**

To this end, we apply *tool-pose descriptors* to represent tools combined with the way in which they are grasped, and *affordance vectors* to represent the effect tool-poses achieve in function of the action performed. This way, tool affordance learning consists in determining the mapping between these 2 representations, which is achieved in 2 steps. First, the dimensionality of both representations is reduced by unsupervisedly mapping them onto respective Self-Organizing Maps (SOMs). Then, the mapping between the neurons in the tool-pose SOM and the neurons in the affordance SOM for pairs of tool-poses and their corresponding affordance vectors, respectively, is learned with a neural based regression model. This method enables the robot to accurately predict the effect of its actions using tools, and thus to select the best action for a given goal, even with tools not seen on the learning phase.

I. INTRODUCTION

The concept of affordances was introduced by the cognitive psychologist James J. Gibson in the late 70's as the *"latent action possibilities available to the agent"* that it can perceive directly from the environment in order to interact with. The first steps towards affordance learning robots were taken by Fitzpatrick et al. in their pioneer work [1], which showed that a robot can learn affordances by observing the effect that its actions produce on objects. Tool affordances on robotics were first studied in [2], where for each tool, affordances were learned as a list containing the actions performed and the probability of success in moving an external object. The main drawback of these early studies was that objects and tools were described by given labels, so generalization to new ones outside the initial training set was not feasible.

However, 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. Jain and Inamura applied this assumption to define a set of *functional features* that were representative of the tool's functionality [3]. In their work, a set of predefined geometric features (corners, bars, etc) were computed and linked to the action and effect, by applying the Bayesian Network (BN) framework proposed in [4]. Gonçalvez et al. improved this model by using simple 2D geometrical features extracted from vision to represent functional features, rather than predefined ones [5]. On Dehban et al. [6], the BN was substituted by a Denoising Auto-encoder, which allows for real value input and online learning [6]. Tikhanoff et al. took another approach in [7], by coupling the affordance models with a geometric reasoner in order to determine the feasibility of exploiting the learned affordances on a given scenario, although not generalizable to new tools. This shortcoming was tackled in [8], where a large set of 2D features from the contour of the tool was applied to predict grasp dependent tool affordances.

Recently, a few authors have proposed to adopt tool representations based on 3D features, which are potentially more robust to variability induced by occlusions and perspective than 2D ones. Myers et al. collected an extensive dataset of RGB-D images of tools with associated pixel-wise human labeled affordances, and applied state-of-the-art supervised classifiers to predict the affordances from the tool images [9].Abelha et al. took a different approach by estimating the suitability of a set of household objects as tools for a set of given tasks by fitting the object's superquadric model to the one of the canonical tool for that task [10]. An important drawback of these studies is that they focus only on vision, not taking in account the robot capabilities or its action repertoire.

On studies involving the robot interaction for affordance learning, it is a common practice to apply a limited number of possible outcomes. This is done either by automatically clustering the perceived effect ([11], [12], [13], [8]), or by previously defining a set of effect categories where the results of the robot actions are assigned [14], [15]. Similarly, most studies which define objects or tools in terms of features, apply clustering techniques to group them for further processing [16], [17]. However, it is frequently the case that these discretized outcome labels 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

[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, giorgio.metta@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++).

affordances.

In order to overcome these drawbacks, we propose a learning architecture based on parallel SOM mapping and SOM to SOM regression capable of learning tool affordances while avoiding clustering or predefined categories. This way, the system is able to keep a fine grain representation of all the involved elements, namely tools and effects, and produce accurate predictions beyond typical cluster resolution. Moreover, we further study the suitability of grasp-dependent 3D tool descriptors, by comparing 3 distinct variations of the Oriented Multi-Scale Extended Gaussian (OMS-EGI) feature vector introduced in [17].

II. Materials and Methods

A. Robotic Platform

All the experiments presented in this paper were carried out using the iCub humanoid robot simulator. The iCub is a full body humanoid robot with 53 Degrees of Freedom [18]. In the current experiment we only made use of the upper body, which comprises 41 DoF, including head, arms and torso. The iCub simulator simulates rigid body dynamics and collision detection by making use of ODE (Open Dynamic Engine) [19].

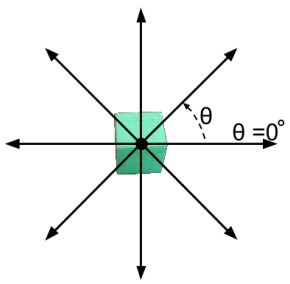
The iCub software is structured as modules that communicate with each other using YARP middleware [20]. All the tool 3D models used for the simulator and feature extraction had been modeled using Trimble's SketchUp software. For feature extraction and visualization, they were processed with the Point Cloud Library [21]. Experimental data analysis was implemented in MATLAB, employing the third party SOM toolbox for dimensionality reduction and data visualization [22], 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 openly available at `www.github.com/robotology/affordances`.

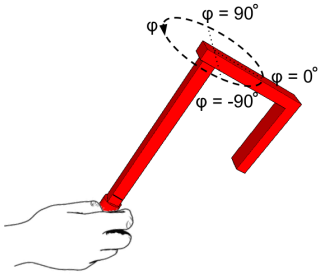
B. Experimental setup

In general, the effect that an action with a tool can achieve depends not only on the tool itself, but also on the way in which it is grasped. Thus, for convenience, we use the term tool-pose to specify a particular tool in a particular pose, following the nomenclature in [23]. That is, a tool in two different poses correspond to two tool-poses, as well as two different tools in the same pose. The experimental scenario in the present study was devised so that with a relatively simple repertoire of actions, different tool-poses achieved distinct effects.

For that end, trials consisted of series of radial drag actions upon a small target object. Specifically, on each trial a drag action of 17 cm was executed upon the object 8 times, along directions at intervals of 45 degrees (see Figure 1a). After each execution, the iCub withdrew the arm and computed the resulting effect, measured as the Euclidean distance that the object had been displaced on the table's plane. Thus, on each trial, a vector consisting of the 8 displacement values



(a) Diagram of the drag action. The tooltip is initially placed slightly behind the object, and then the tool is displaced 17 cm along the radial direction given by θ.



(b) Grasp parameter φ controls the rotation around the tool's handle axis. Orientations `right`, `front` and `left` correspond to angles $\varphi = -90, \varphi = 0$ and $\varphi = 90$, respectively.

Fig. 1: Parameters controlling action execution (a) and tool pose (b).

measured was recorded, whose indices correspond to the 8 considered actions. This vector effectively represent how well the tool-pose being held afforded dragging an object in function on the angle of the drag. Therefore, we refer to them as *affordance vector*.

The target object was a small cube of 5 cm in side, which before each action execution was placed on a table at a spot 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 position of the object was chosen so that the robot could perform the dragging action in any direction without colliding with itself (when pulling) or going out of reach limits (when dragging away).

50 different radial tools [1] , roughly divided in 5 categories, were used in to perform the experiments (see Figure 2). Each of them was hold by the iCub in 3 possible orientations (`right`, `front`, `left`, see Figure 1b for details), therefore making up a total of 150 tool-poses used for the experiments. On each trial, the 3D model of the corresponding tool-pose was simultaneously loaded on the simulator from memory and read by the processing modules for subsequent feature extraction. Also, it was used to determine the position of the tooltip with respect to the hand's 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. An image of the set up ready for an action can be observed in Figure 3.

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

The way tools are represented determines how well knowledge about them can be generalized. Therefore, we applied the Oriented Multi-Scale Extended Gaussian Images (OMS-EGI) descriptor, proposed in [17]. In a nutshell, the OMS-EGI descriptor of a tool is an ordered concatenation of

[1] Radial tool refers to tools which are typically grasped in a way such that the index and middle fingers curl around the tool with the thumb beginning to oppose and press it, and the tool effector extends along the radial side of the hand.

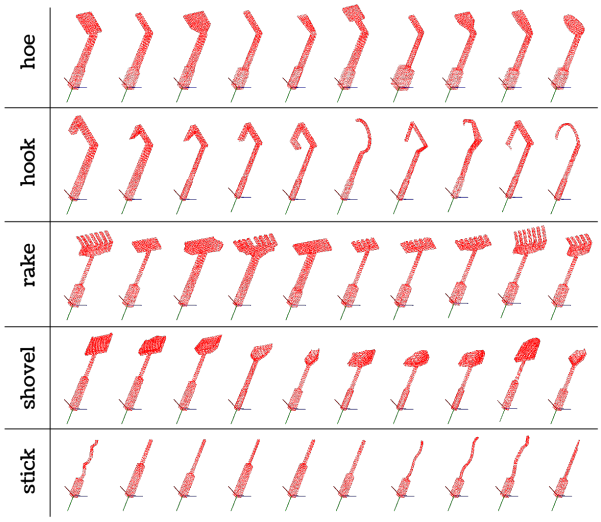


Fig. 2: Set of tool models used in this study.

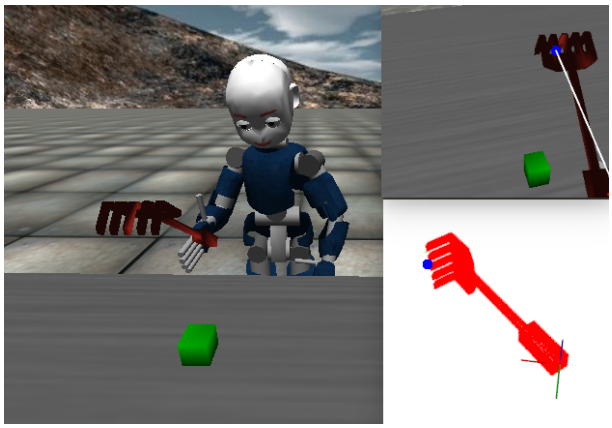


Fig. 3: Experimental setup ready for action execution. The lower right image shows the pointcloud representation of the tool, oriented w.r.t the hand reference frame, as well as the determined location of the tooltip (the blue dot). On the upper right, the iCub vision camera is shown, with the kinematic extension to the tooltip superimposed.

surface normal histograms computed from voxels obtained from iterative octree divisions of the tool's bounding box aligned with respect to the robot's hand reference frame. The resolution of these voxel-wise histograms in terms of the number of bins per dimension is given by the parameter N, while the depth of the octree, i.e. how many times the bounding box is iteratively divided into octants to form increasingly smaller voxels, is given by the depth parameter D. A detailed description of how the OMS-EGI feature vector is computed can be found in [17].

The fact that the OMS-EGI descriptor is computed from the bounding box of the tool *aligned* to the robot's hand reference frame, implies that it encapsulates not only information about the tool's 3D geometry, but also about how it is being grasped. In doing so, it provides a description of tools relative to the robot, and thus particularly appropriate for interaction scenarios.

At the same time, the OMS-EGI descriptor combines information about the surface of the object, given by its normal histograms, with spatial information about where these surfaces are present, thanks to the voxel-wise analysis of surface normals. The relevance of each one is determined in function of the parameters N and D. 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, its surface information, or a balanced combination of both. We conducted such an evaluation by comparing the predictive performance of the following 3 parameter settings:

- **Balanced information (BALAN)**: Setting $N = 2$ and $D = 2$, the feature vector corresponds to a *balanced* OMS-EGI, as applied in [17], where both surface and spatial information are represented.
- **Spatial information (OCCUP)**: If $N = 1$, all normals in each voxel are assigned to the same bin irrespective of their orientation, and therefore each voxel-wise histogram can be subsumed in a single value. On voxels where any portion of the surface is present, this value is 1, while on empty voxels the value is 0. Therefore, setting $N = 1$ transforms the OMS-EGI into a axis aligned space occupancy grid. In the present study, D is set to 3 so that the total length of the feature vector is similar to the OMS-EGI setting.
- **Surface information (EGI)**: When $D = 0$, the only voxel considered is actually the tool-pose aligned bounding box, without further subdivisions. In this case, which is equivalent to the original formulation of the EGI descriptor [24], the OMS-EGI represents a normal histogram of the tool-pose, provided a certain histogram resolution function of N. In this study, N is set to 6 so that the length of the vector is in a similar range to the other settings.

D. Parallel SOM mapping from tool-pose features to affordances

In the proposed learning architecture, which can be observed in Figure 4, tool affordances are learned as the mapping between tool-pose OMS-EGI features $X \in \mathbb{R}^L$, and their corresponding affordance vectors, $Y \in \mathbb{R}^K$. L is the length of the tool-pose feature vector and K is given by the number of directions of action application considered. Formally, affordance learning is thus implemented as finding the mapping $f : \mathbb{R}^L \rightarrow \mathbb{R}^K$.

Yet, performing such mapping directly is prone to numerical errors because of the high-dimensionality of the tool-pose feature and affordance vector spaces and the relatively small number of available samples. Therefore, instead of attempting at learning directly the regression between both spaces, we mapped them onto respective 2-dimensional Self-Organized Maps, referred henceforth as *tool-pose SOM* and *affordance SOM*. SOMs provide efficient dimensionality reduction while maintaining to a great extent the topology of the data in the original high-dimensional space [25], and a fine grained representation when compared to clustering

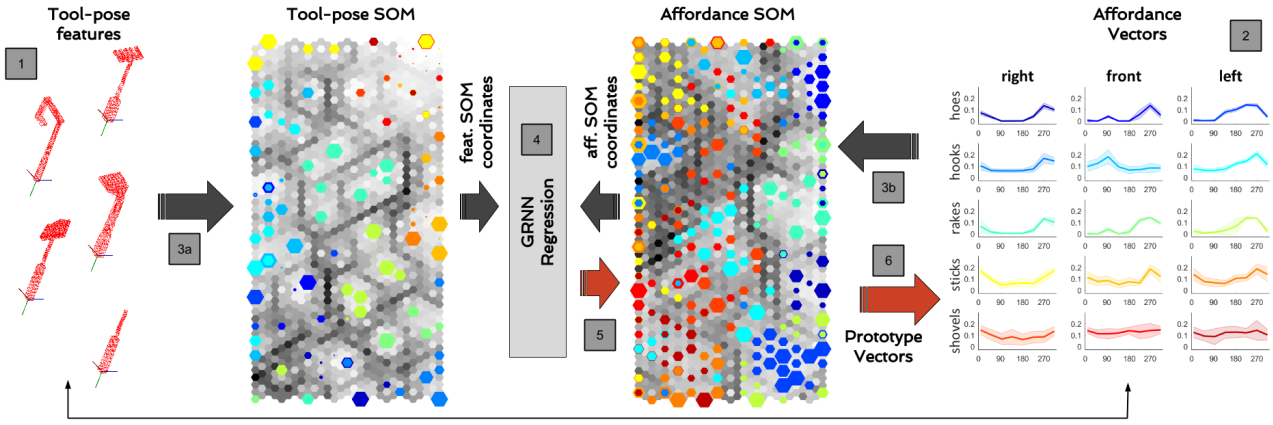


Fig. 4: Diagram of the proposed approach to discover, learn and predict tool-pose affordances. On the training phase (black arrows), a set of tool-pose features [1], and their corresponding affordance vectors [2] are available to the system. Keeping the correspondence, tool-pose features and affordance 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 vector on the **affordance SOM** [4]. On the prediction phase (red arrows), affordance SOM coordinates are estimated by the regressor from the tool-pose SOM coordinates of the given tool-pose [5]. The prototype vector of the closest neuron to the estimated coordinates is considered the predicted affordance vector for that given tool-pose [6]. For easier interpretation, each color corresponds to data generated by a particular tool type (`hoe, rake`, etc) in a particular pose (`right, front, left`), assigned following the affordance vector graphs on the right of the diagram. Tool type information, however, is only used for visualization purposes, and was never available to the system.

techniques. Indeed, similar methods involving 2 parallel SOMs have been used in [13], [26] for object affordance learning, but applying very different data modalities and learning and prediction methods.

In this particular scenario, using SOMs for dimensionality reduction has further advantages. Primarily, whereas training the regressor requires matching pairs of tool-pose features and affordance vectors, SOMs can be trained with data which does not necessarily has a corresponding pair, due to the unsupervised nature of their training. Thus, tool-pose features can be "imagined" by extracting them from slight rotations of the tool-pose pointclouds used in the experiment. We applied this *data augmentation* technique to generate more tool-pose feature samples with which to train the tool-pose SOM topology than the number for which we have corresponding affordance vectors. Another important advantage is that the prototype vectors associated to each of the SOM neurons provide a mechanism to invert the dimensionality reduction and yield a prediction for the affordance vectors in the original space.

Once the SOMs are trained to provide a reduced dimensionality representation of the original vector spaces, affordance learning can be implemented as a regression problem, where the input consists in the tool-pose SOM neuron coordinates, and the target is given by the affordance SOM neuron coordinates of the corresponding affordance vectors. In order to learn this regression, we compute first the best matching units (BMUs) for all the train tool-pose features and affordance vectors on their corresponding SOMs. We refer to the BMUs corresponding to the tool-pose features as $X_{SOM}, \in \mathbb{R}^2$, and to those corresponding the affordance vectors as $Y_{SOM}, \in \mathbb{R}^2$. The desired regression function $f(X_{SOM}) \rightarrow Y_{SOM}$ is 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 [27]. 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[2].

On the prediction phase, the first step is to extract the tool-pose feature vector $x \in X$ that represents the tool-pose being held, as described in Section II-C. The obtained vector is then mapped to the trained tool-pose SOM to obtain the coordinates x_{SOM} of its BMU. These coordinates are subsequently fed to the trained GRNN model, which in turn returns the estimated coordinates $\hat{y}_{SOM}$ of the BMU of the corresponding affordance vector for that tool-pose. Finally, the predicted affordance vector $\hat{y}$ was determined as the prototype vector of the closest neuron to the predicted coordinates, and compared to the real affordance vector $y \in Y$ to assess the accuracy of the prediction.

E. Prediction based action selection

The methods described above allow the iCub to predict the effect of performing a drag action in any of the considered directions for any radial tool, having also in account the way

[2]Recursive linesearch was conducted by evaluating the accuracy of the regressor at equally spaced values of σ with 5-fold cross validation on the training data, 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.

in which it is grasping the tool. Thereby, the robot is able to exploit its acquired knowledge of tool-pose affordances to select the action direction that provides the best expected effect for any given task.

In the present study, we tested this claim by making the iCub select the best action for the task of maximizing displacement of the target object. To ensure fairness on the test, 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. So, in order to test any tool in any given grasp, the first step was to compute the OMS-EGI feature vector of the resulting tool-pose. Then, the methods described in the previous section were applied to obtain its predicted affordance vector. Based on the predicted affordance vector, the action direction with the maximum predicted effect was selected, and its parameters sent to the robot to be performed. After the iCub executed the action in the selected direction, the actual achieved effect was measured in order to evaluate the success on the given task.

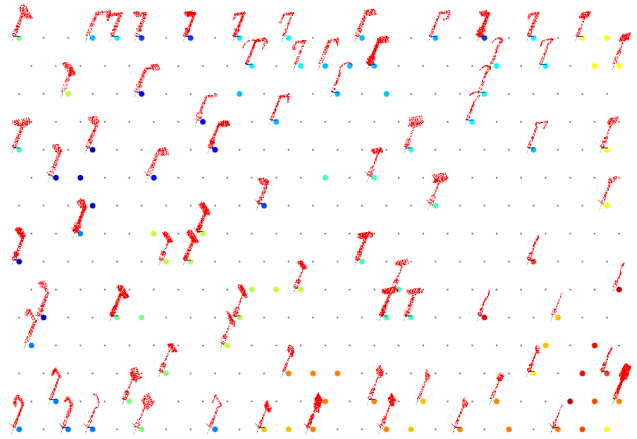
III. Results

A. Experimental data collection and separation

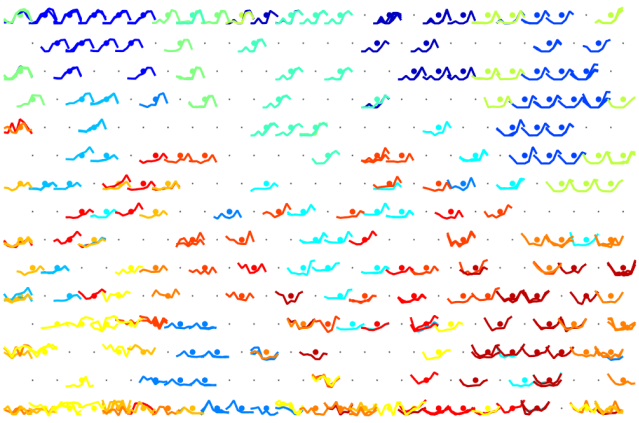
All experiments in the present study were carried out with the iCub simulator. As described in Section II-B, trials were performed with 150 different tool-poses, corresponding to 50 radial tools in 3 different poses each. For each tool-pose, 4 exploration trials were performed in order to take in account the possible variation between executions due to the slightly different location of the target object. As a result, a total of 600 trials were carried out, corresponding to 4800 action executions. On each of these trials, the data recorded consisted on the affordance vector representing the recorded effects for the 8 action directions, and the tool-pose OMS-EGI feature vector corresponding to the tool-pose being used to perform those actions. Additionally, for each tool-pose used in to perform the experiment, 30 extra OMS-EGI feature vectors without an associated affordance vector were gathered by means of the *data augmentation* method described in II-D. This number was selected in order to have a number of OMS-EGI samples considerably larger than its dimension L to perform the unsupervised training of the tool-pose SOM.

Each OMS-EGI feature vector, whether "natural" or "augmented", was computed and recorded in the 3 variants described in section II-C, that is, **OCCUP** to represent only spatial information as an axis aligned occupancy grid, **EGI** to consider only the surface as a single-voxel normal histogram, and **BALAN** for a balanced compromise between the previous 2.

In each case, the data gathered was divided into training and testing sets to evaluate the presented methods. However, in order to provide a more complete assessment of their performance, we applied two different data separation schemes. The first separation scheme serves to evaluate the general predictive performance of the proposed method, and is achieved by randomly selecting the data corresponding to 25% of the trials for testing, and keeping the rest for training. We refer to this separation scheme as **RAND**. The second separation scheme assesses the capability of the method to generalize the learned affordances to tools that have not been seen by the system during training. For that end, we perform tool-wise leave-one-out separation where on each run, the data from all the trials corresponding to a given tool (in all its poses) is kept for testing, while data from the rest of the tools used for training. This scheme is referred to as **1OUT**.



(a) Trained **tool-pose SOM**.



(b) Trained **affordance SOM**.

Fig. 5: Trained SOMs. In order to understand better how the high-dimensional data is mapped onto the SOMs, samples of the training data (tool-pose models (a) and affordance vectors (b)) are plotted onto their corresponding BMUs (represented by colored dots). To avoid clutter, only 1 tool-pose model has been plotted for every 15 samples used to train the SOM. Uncolored dots represent neurons which were not the BMU of any of their respective data samples. Colors, as in Figure 4, represent the tool type and poses that generated the data, and are used solely for visualization.

B. SOM-based unsupervised dimensionality reduction

As described in Section II-D, the first step in the proposed method for affordance learning is to map the spaces of tool-pose features and affordance vectors into their respective

SOMs. In the current study, both SOMs were chosen to have a hexagonal lattice of 15 x 20 units, which provided a good compromise between representation resolution and training time required. The tool-pose SOM was trained using all the vectors not used for testing the affordance prediction, including the ones provided by the data augmentation technique. The results of this mapping process can be observed on Figure 5a. 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 5b.

C. Prediction of tool-pose affordances

In order to evaluate the performance of the method for affordance prediction described in Section II-D, we compared the affordance vectors predicted for the test trials with the corresponding real affordance vectors previously recorded, for the different data separation schemes and OMS-EGI features parameter settings. In each case, a baseline performance was also computed by randomly shuffling all the vector indices before training the GRNN, effectively removing all correlation information between tool-pose feature and affordance vectors. This allows the comparison of the prediction results for the trained system against the results obtained in the absence of learning. The absolute performance of the system 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. Additionally, the improvement of the performance over the baseline MAE, denoted by MAE_{BL}, due to the learning process, was quantified by the percentage of improvement (PI), so that if error was not improved after learning, PI would be 0% while if error was reduced to 0, PI would be 100%. Formally:

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

where N is the number of test trials, and

$$PI = \frac{MAE_{BL} - MAE}{MAE_{BL}} \tag{2}$$

Table I displays the prediction error in each of the evaluation scenarios, expressed in terms of the MAE computed as the average from 50 runs in **RAND** and **1OUT** data separation modes, where in the latter each run corresponded to one left-out tool. In Figure 6 the comparison between prediction and recorded data can be observed graphically.

D. Action Selection

In the last evaluation step, we employed the learned models to select, given a tool-pose not observed during training, the best action direction for the task of achieving maximum displacement of the target object, as explained in Section II-E. This test was run 2 times for all the tool-poses. In order to ensure fair evaluation, we applied the **1OUT** separation schema for this test, so that the data corresponding to the tested tools had never been used to train the models used to predict its affordances.

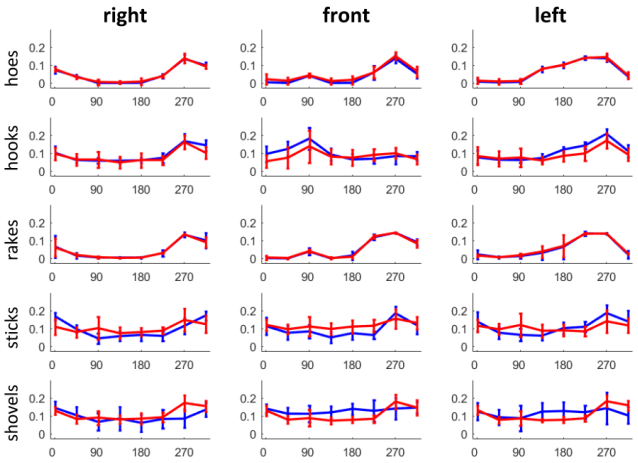


Fig. 6: Predicted effect (red) against recorded effect (blue) with variance (red vertical lines), using the 1-OUT data separation scheme. Each graph row corresponds to the aggregated data of all tools in a tool category (`hoe, rake, etc`), separated by pose (on columns). In all graphs, X axis corresponds to the angle of the drag action, from $\theta = 0°$ to $315°$, and the Y axis to the displacement (predicted or measured) in meters.

In this test, the baseline was defined for each tool-pose as the median among the effects achieved for all action executions recorded previously for that tool-pose. In principle, if actions had been selected at random, the achieved effect would be over this baseline 50% of the times. Figure 7 shows the displacement that the robot achieved using the predicted best action direction for each of the tested tool-poses, alongside the corresponding baseline. The overall degree of task success was measured in 2 ways, similar to [15], 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 hand, 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 of this test can be observed in Table II, where they are separated by tool type.

IV. Discussion

The prediction results, displayed in Figure 6 and Table I, evince that even in the presence of tools with unknown affordances, the proposed methods are able to successfully generalize the knowledge learned for similar tool-poses, and apply it to correctly predict the effect the tool will generate for any action in the repertoire. This approach drastically improves the performance our previous method [17] in terms

	BALAN			EGI			OCCUP		
	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$	MAE	MAE_{BL}	$PI(\%)$
RAND	2.77	5.00	44.6	2.85	4.99	42.9	2.66	5.02	47.0
1OUT	3.28	5.12	31.4	3.28	5.09	35.6	3.26	5.09	35.9

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

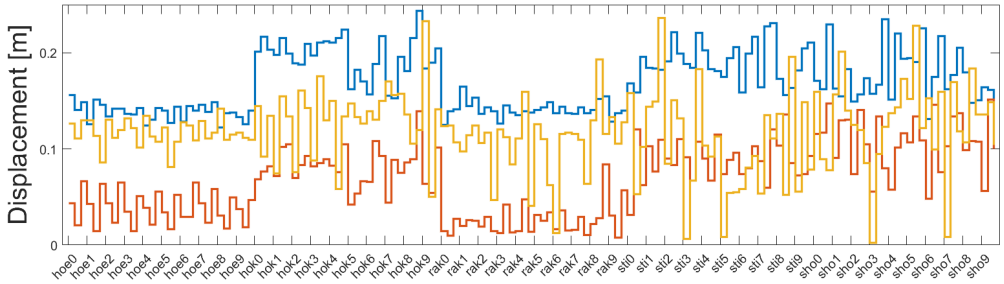


Fig. 7: Results of the Action Selection experiment by tool-pose. The effect measured from execution of best action for each test tool-pose (orange) is displayed against the baseline for that tool-pose (red), and its maximum (dark-blue).

	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

TABLE II: Action selection performance results.

of the PI achieved on the MAE over the baseline, which soars from 29.7% in [17] to 47% in the present study.

Also, during this study we compared 3 variations of the OMSEGI descriptor; EGI, OMSEGI and OCCUP, described in Section II-C. Interestingly, the best results have been consistently obtained with the OCCUP parameter setting, which is equivalent to a binary occupancy grid cell, while the traditional EGI yielded the lower performance also in all scenarios. Our presumption is that the coarse spatial information of the tool's extension and position with respect to the hand provided by the occupancy grid directly matters for interaction. The surface normals, on the other hand, would be better at describing curvature and smaller detail, less relevant for the performance in the current scenario.

On the action selection results displayed in Figure 7 and Table II it can be observed that the accurate affordance prediction achieved enables the iCub to select the best action direction to accomplish the displacement task with a high rate of success. Yet, in some cases the selected action produces a displacement much smaller than expected. We believe that these errors are generated when certain tool-poses generate different effects for the same action, which prevents proper learning of these tool-pose affordances.

By observing the experiment, we noticed that this happened mostly because during action execution, due to errors in collision calculations, when a tool pushed the target object down against the table there were some cases where the object would "jump" a few centimeters away in an unpredictable direction. This situation happened with tool-poses where the tooltip was situated in the same vertical axis as the handle, namely `sticks`, `hooks` oriented to the front, and some `shovels`. As a result, the mapping between actions and effects for these tool-poses has a large amount of variance, which renders the prediction of their affordances a much more error prone task, and thus 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, the selected actions led to successful effects with a high degree of accuracy (up to 100% in the case of `rakes`). This suggest that in this scenario, results in the real robot would actually be more robust, since physics glitches do not occur in real life. In fact, this seems to be the case in some preliminary experiments already performed.

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. [15], 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 gets seriously penalized by the inaccuracies on the prediction of the *sticks* affordances, and therefore is on average lower than on their study. On the other hand, the overall accuracy is nevertheless around 6.5% higher in our study. An important factor to take in account, however, is that in our study we consider 8 possible directions and 150 different tool-poses, while in [15], only 4 tools and 4 push directions are considered.

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

Concerning the target object, we have only considered its location, disregarding any properties such as geometry or material. We acknowledge that these properties do influence the effect of actions on the object, but as in the previous literature [7], we assume that it can or has been learned in previous stages of the robot development.

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

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

V. Conclusions

In this paper, we presented set of methods 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 simulator on a large dataset of tools. The presented method learns affordances as a regression between tool-pose and action-effect representations on a SOM, which allows the system to keep a fine grain representation of both elements. Moreover, we further studied the suitability of 3D tool descriptors for interaction learning, revealing that coarse spatial information seems to be more relevant than detailed surface one to describe tools in affordance learning scenarios. Results show that the combination of the proposed features and learning architecture enables the system to produce accurate predictions, which can be used to select the best action for a given task with a high degree of accuracy.

References

[1] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, G. Sandini, L. Natalet, S. Raot, and G. Sandinit, "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.

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

[3] R. Jain and T. Inamura, "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, sep 2013.

[4] 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, oct 2007, pp. 4102–4107.

[5] 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.

[6] 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.

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

[8] 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.

[9] A. Myers, C. L. Teo, C. Ferm, and Y. Aloimonos, "Affordance Detection of Tool Parts from Geometric Features," in *International Conference on Robotics and Automation*, 2015, pp. 1374–1381.

[10] P. Abelha, F. Guerin, and M. Schoeler, "A Model-Based Approach to Finding Substitute Tools in 3D Vision Data," 2015.

[11] 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*. Ieee, aug 2008, pp. 91–96.

[12] 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, jul 2011.

[13] 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.

[14] 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, feb 2008.

[15] 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. 13–16.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

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

[24] B. K. P. Horn, "Extended Gaussian Images." *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1671–1686, 1984.

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

[26] B. Ridge, A. Leonardis, A. Ude, M. Deniša, and D. Skočaj, "Self-Supervised Online Learning of Basic Object Push Affordances," *International Journal of Advanced Robotic Systems*, no. November, p. 1, 2015.

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