Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot

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

***Abstract*— The ability to learn about and efficiently use tools constitutes a desirable property for general purpose humanoid robots, as it allows them to extend their capabilities beyond the limitations of their own body. Yet, it is a topic that has only recently been tackled from the robotics community. Most of the studies published so far make use of tool representations that allow their models to generalize the knowledge among similar tools in a very limited way. Moreover, most studies assume that the tool is always grasped in its common or *canonical* grasp position, thus not considering the influence of the grasp configuration in the outcome of the actions performed with them.**

In the current paper we present a method that tackles both issues simultaneously by using an extended set of functional features and a novel representation of the effect of the tool use. Together, they implicitly account for the grasping configuration and allow the iCub to generalize among tools based on their geometry. Moreover, learning happens in a self-supervised manner: First, the robot autonomously discovers the affordance categories of the tools by clustering the effect of their usage. These categories are subsequently used as a teaching signal to associate visually obtained functional features to the expected tool's affordance. In the experiments, we show how this technique can be effectively used to select, given a tool, the best action to achieve a desired effect.

I. INTRODUCTION

A practical way to approach tool use is by modeling it as the relationship between the agent's actions, the external object effector being used (the tool), and the resulting effect in the agent's environment. These relationships can in principle be learned and used to predict future effects or to choose appropriate tools or actions. The concept of affordances, first proposed by Gibson [1], has proven to provide an adequate theoretical framework to study of these interactions, also within the field of robotics [2], [3].

The first approach to robotic tool use from a developmental perspective was conducted by Stoytchev [4], in which affordances were learned for each available tool by populating a list with behavioral parameters that led to success in a sliding task when performed with that tool (labeled by color). Using a very similar setup, Sinapov [5] improved Stoytchev's method by hierarchically clustering the effects, which allowed the robot to discover new categories of affordances. A more recent study by Tikhanoff et al. [6] combined exploratory behaviors and geometrical feature extraction in order to determine which tool of a small given set affords better to pull into reach a toy otherwise unreachable by the robot with its own hand. The major drawback of the studies mentioned above is that the affordances were learned for a given set of labeled tools, so the knowledge gained remained limited to the tools in the original set.

A few studies have tried to overcome this problem by using other forms of tool representation that could enable the generalization of the knowledge to previously unseen tools if they shared some characteristics. Some authors focused in the effector part of the tool, such as in [7], where the functionality is assumed to be on the tip of the tool, which might be shared by similar tools, or in [8], where the degree of matching between the tool and the target object contours is assessed to predict the tool affordance possibilities on that object. A similar concept was employed in [9] and [10] where the authors propose to substitute the *tool* element of the tool affordance tuple by the existence or not of a set of *tool functional features* which combined with certain actions result in predictable effects. The advantage of this kind of features over previous approaches is that they allow the inference of possible affordances of previously unknown tools by observing them visually. Nevertheless, the functional features considered are very few and specifically related to the task at hand, so they would need to be redefined to look for their relationship with other kind of affordances. Other studies have in fact applied large sets of non-specific or even non-specified functional features in object affordance learning scenarios, where the robot interacted directly with a target object [11], [12], [13], but never so far in tool use scenarios.

Apart from tool representation, the other topic investigated in the present paper is the effect of different grasps on the tool's affordances. Robotic tool grasping has indeed been extensively studied but almost always assessed from the perspective of stability of the grasp or the suitability of the joint configuration [14], [15], [16], [17]. To our best knowledge, only [18] considered tool-pose (equivalent to grasping configuration) in the context of tool affordances. Indeed, considering the grasping position as a variable includes additional complexity to the problem. However, we believe that if it is conveniently approached, the problem of grasping for tool use can not only be efficiently tackled, but also lead to more robust tool use behaviors, as suggested in [19].

This paper proposes a method which simultaneously tackles the aforementioned drawbacks. On one hand it allows the

[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++)

robot to autonomously discover the set of distinct affordances that a group of tools provide, taking in account both how the tool is grasped, and how the action is performed. On the other hand, it also enables the robot to predict which will be the affordance of a grasped tool based on the tool's functional features, defined as those features (visual or otherwise) that can potentially influence tool's functionality. In this manner, it can set the parameters of its action towards maximizing a particular goal, even if the grasped tool has not been observed before.

II. MATERIALS AND METHODS

A. Experimental Setup

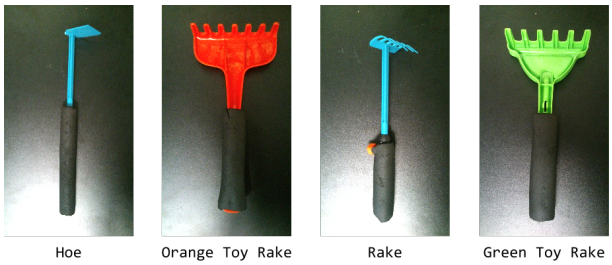
The experiments carried out in the present study were performed using the iCub Humanoid Robot [20] as well as its simulator [21]. The iCub has 53 degrees of freedom, although in the current scenario we mainly use the 41 of the upper torso. It is also equipped with force-torque, joint angle and inertial sensors, binocular vision, and full body skin sensing capabilities, although we do not use the latter in this scenario. The software on the iCub is built based on modules which communicate with each other using YARP, enabling multi-machine and multi-platform integration [22].

For the current experiment, all modules concerning robot motor actions as well as sensory extraction and processing are written in C++, with extensive use of the OPENCV library for feature extraction. The data analysis and learning part has been programmed in MATLAB, making use of the third party libSVM library [23] to implement SVM learning algorithms.

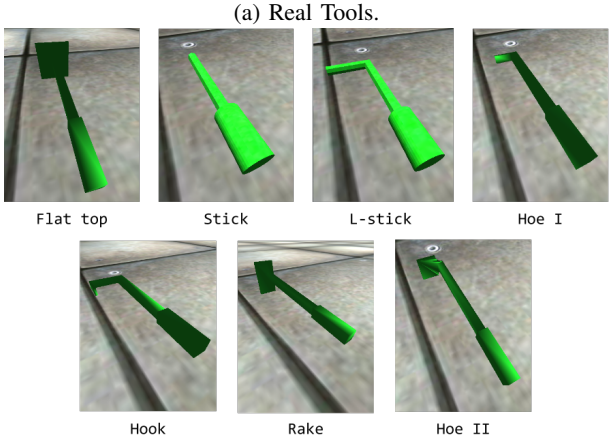
We have used 7 different tools for the experiments on the simulator and 4 for those on the real robot, which can be observed in Figures 1a and 1b, respectively. In order to study how the way in which a tool is grasped affects what it affords, the iCub held each tool with the end-effector oriented in three different ways, as shown on Figure 2: either to the front, to the right, or to the left. Our method does not use tool labels in any step of the process but nevertheless tools have to be given to the robot in a particular configuration. Therefore, we refer to each of the combinations between the available tools and the given orientations of the end-effector as a *tool-pose*, following the nomenclature used in [18]. The effect of the tool use was measured as the displacement of a small cube placed on a table in front of the robot, whose approximate size is known.

B. Tool-pose based functional features

The first part of the method consists in the extraction of 2D geometrical features of the grasped tool. We argue that these features will relate to the functionality of the tool-pose, and hence could be later used to predict its effects. Therefore, the first step is to determine the position of the tool. This is done differently on the simulator and the real robot. On the simulator the tooltip is set at a fixed position relative to the hand center while on the real robot we used the method developed in [6]. This method involves a discovery procedure in which the robot swings the grasped tool in different



(a) Real Tools.



(b) Simulated Tools.

Fig. 1: Tools used on the experiment on the real robot (a) and simulation (b).

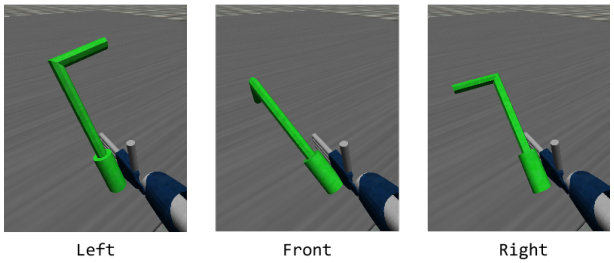


Fig. 2: Tool Orientations applied: Left, front and right

positions, while the generated motion is detected by a variant of the Lucas-Kanade optical flow algorithm extended to compensate for ego-motion [24]. An optimization technique analyzes the motion data to reliably establish the tooltip position based on the minimum error between the predicted position of the tooltip and the observed one. When the position of the tooltip with respect to the hand is found, it is annexed to the arm's forward kinematics so it can be used as an end-effector and retrieved at any moment.

In parallel, another module is running a graph-based segmentation (GBS) algorithm [25] on the images from the left eye to split them into uniform regions. The 2D projection of the tooltip in the image plane is used as the seed for the segmentation to extract the blob that corresponds to the tool. It should be noted that while the pose that the robot adopts in order to observe the tool and extract its features is always such that the whole tool is within its visual field, the specific position and orientation of the hand are slightly

randomized so that no two sets of extracted features will be the same, even if the robot is holding the same tool in the same orientation in several observations.

After the blob representing the view of the tool has been isolated from the rest of the image, its orientation is normalized with respect to the angle between the tooltip and the hand center (computed from the robot's kinematics). Additionally, only a region of interest between the tooltip and about $3/4$ of the distance between the tooltip and the hand center is selected, in order to keep only the part of the tool corresponding to the end effector, which is the one that differentiates among tools and poses. Finally, a set of 75 geometrical features is extracted from the contour of the resulting blob. Before being applied for learning, each feature value is normalized by being subtracted the mean and divided by the variance of that feature computed from the training subset of the data. The resulting normalization constants (training mean and variance of each feature) are then used to normalize any incoming test data. An example of the image processing at different stages of the feature extraction pipeline can be observed in Figure 3.

The list of features considered is detailed below, citing which previous works have used them, if any. The number in parenthesis corresponds to the number of values used to represent that feature:

- Based on convex hull
  - Depth of the 5 larger convexity defects[1] (5).
  - Histogram of bisector angles at convexity defects (8).
  - Area of the convex hull.
  - Solidity: Contour area / convex hull area.
- Based on thinning (from [26])
  - Number of skeleton bifurcations to the left, right, under and above the blob's center of mass, respectively (4).
  - Number of skeleton endings to the left, right, under and above the blob's center of mass, respectively (4).
- Moments:
  - Normalized central moments *nu11*, *nu02* and *nu02* (as implemented in OPENCV) (3).
- Shape descriptors (inspired by [27] and [28]).
  - Area.
  - Perimeter.
  - Compactness.
  - Length: major principal axis.
  - Width: minor principal axis.
  - Aspect ration: width/height.
  - Extension to the right w.r.t. the center of mass.
  - Extension to the left w.r.t. the center of mass.
  - Elongation.
  - Rectangularity.
- From the angle signature: outwards pointing normal angle between each two points in the contour (based on [28]).
  - Bending energy: sum of squares of the angle variation along the contour), divided by the number of points in the contour.
  - Angle signature histogram (8).
- Domain transformations from the distance to the centroid signature (from [28]).
  - Fourier coefficients (15).
  - Wavelet coefficients (15).

[1]A convexity defect is defined as any continuous deviation of the object's contour from its convex hull.

C. Discovering and Learning Pull Affordances

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

1) Gathering tool use data through interaction and observation of the effects.
2) Discovering affordances by clustering the observed effects.
3) Mapping functional features to the discovered affordances.

The execution of explorative pulls action and the computation of their effects, required to gather tool use data, rely on 3 processes running in parallel modules that allow the iCub to know where the tooltip is with respect to its own hand, where the object is on the table, and how to bring the former close to the latter and perform the pull, respectively. The modules in charge of finding the tooltip and tracking its location relative to the robot hand have been already described in Section II-B as they are also involved in segmenting the tool blob for contour extraction. The second task is achieved by a tracker module that is able to constantly locate the target object within the field of view, by making use of a particle filter which is initialized at the beginning of the experiment with a user provided template of the desired target object [29]. This way, it is able to reliably provide at any time the position of the pixel in the robot camera where the center of the target object is located. The transformation from the 2D coordinates in the image to the 3D coordinates in the robot's coordinate frame is achieved by the iKinGaze module [30], which takes advantage of the robot's kinematics to determine the direction of the gaze and project the point on the plane of the table, whose height w.r.t the robot is known. Finally, once the tooltip and the object are located, the control of the motor actions is performed by the Cartesian Interface module [31], which handles the inverse kinematics problem on the iCub and enables high level control of the robot directly in the operational space. Therefore, in order to perform a pull action, the only required parameter is the approach position with respect to the object, as the object position is tracked automatically and the actions required to reach and pull are estimated and performed by the iCub's Cartesian Interface module.

Each training trial of the present study consists on 11 pull actions, corresponding to approaches from -5 to 5 cm to either side of the object, on integer centimeter values. When reaching for the object, the tooltip is not directed exactly to the coordinates where the object is detected, but 3 cm behind, so that the end-effector is able, if the approach parameter is adequate, to grab the object during the drag action, which has a constant length of 20 cm. For each approach, the effect of the tool use is computed as the distance between the position of the object before and after the pull action, as provided by the tracker. If the object has been displaced, it is put back after the computation of the effect on approximately the same position as it was before the action. The 11 pairs action-effect obtained by this process represent an affordance vector which describes how well a particular tool-pose affords pulling as a

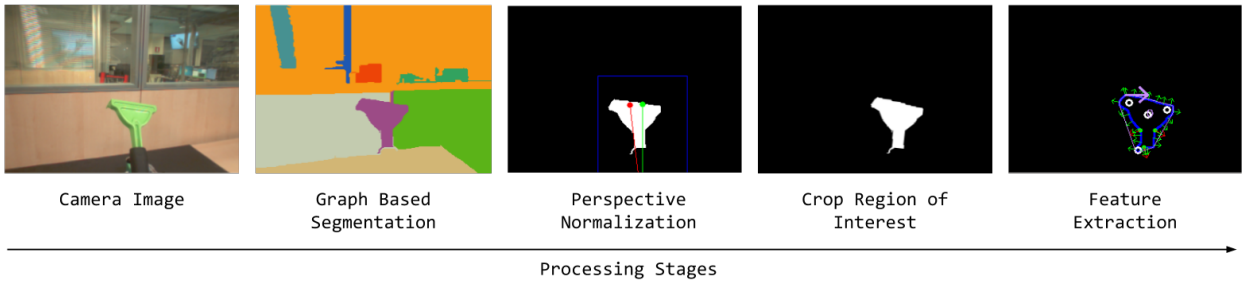


Fig. 3: Visual Feature Extraction processing pipeline.

function of the approach position w.r.t the object. An instance of the pull action can be observed in Figure 4.

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

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

To that end, we performed a series of K-means clustering runs among all the affordance vectors, regardless of the tool-pose that generated them, varying the number of clusters from the minimum ($K = 2$) to half of the number of the considered tool-poses, that is, to $K = 11$ for simulation data and $K = 6$ for real robot data. For each run, the quality of the clustering is assessed with the Davies-Bouldin clustering index [32], which measures how well separated the clusters are as the ratio between the average of the mean scatter of the points in each cluster and the average distance among their centroids. Thus, the number of clusters that better separate the affordance vectors can be determined as the K which produces the lowest DB-index, K_{best}.

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

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

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

A schematic diagram representing the flow of information and the modules involved in the process of discovering and learning affordances described above is shown on Figure 5.

D. Affordance Prediction and Action Execution

After the training process, the robot has learned the mapping between the geometrical features of the different tool-poses and the category of affordance that they are likely to achieve. By retrieving the prototype vector of the predicted affordance category, the robot is able to compute the expected effect for each of the possible values of the approach parameter. Based on this information, the parameter value that is expected to maximize the displacement of the object is chosen and the respective pulling action subsequently performed. This process is shown on Figure 5 superimposed to the diagram of the training process, on which it relies. The effect caused on the object by the robot's action is measured as in the training trials, by retrieving the coordinates of the object before and after the action has been performed, and measuring the euclidean distance between them. A video presenting a condensed version of the whole experiment can be watched at: `https://www.youtube.com/watch?v=neiX_eP4qq4`.

III. RESULTS

A. Discovering Affordances

Due to the random initialization of the K-means clustering algorithm, the K_{best} number of clusters which minimizes the DB-index can vary significantly from run to run. In order to cope with this inconsistency, we ran 1000 times the procedure to find K_{best} described above. The histograms on Figure 6 show that the number of clusters which was more likely to be K_{best} was 3. This means that the effects that the different tool-poses achieve on the object naturally group into 3 affordance categories, both in real and simulated experiments. Moreover, if we observe the prototype vectors

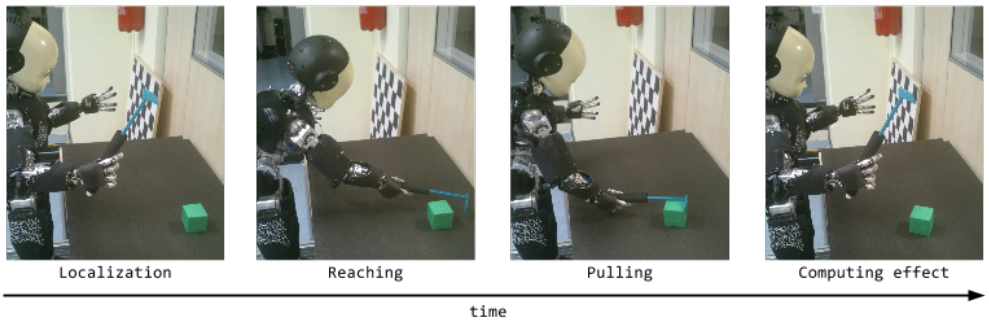


Fig. 4: Pull action sequence.

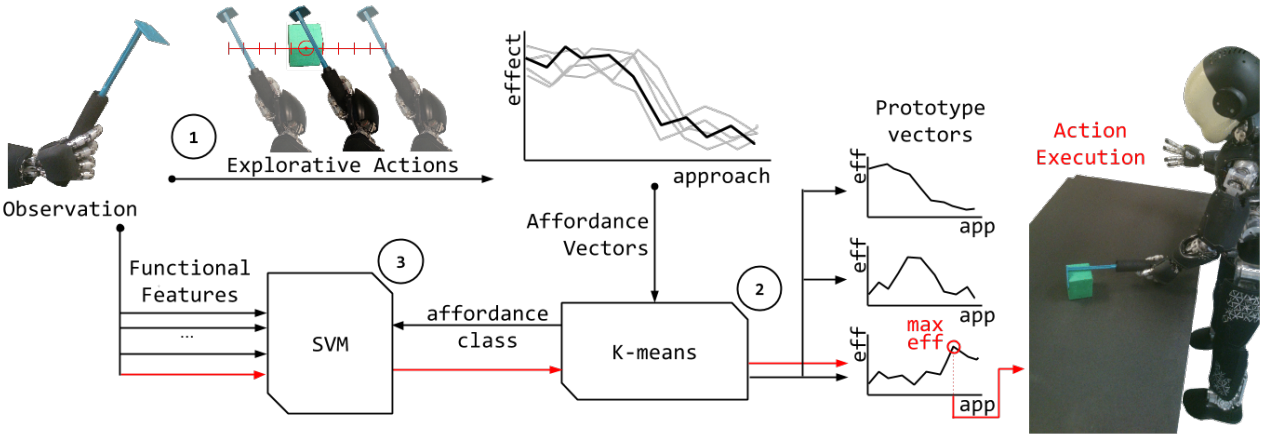


Fig. 5: Diagram of the proposed affordance learning and prediction method. Black and red arrows represent the flow of information in the training and prediction phases, respectively. The three stages of the training procedure are indicated by the circled numbers as follow: (1) Gathering of tool use data through explorative actions and observation of the effects; (2) Discovering affordance types by clustering the observed effects; (3) Mapping functional features to the discovered affordance classes.

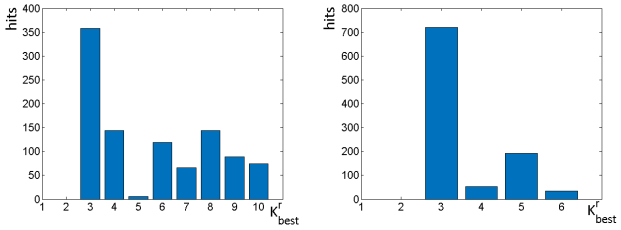
of each affordance cluster, as represented by their centroids (shown on Figure 7), they resemble the different effects that one would expect to find: large effect when the approach is either on the right, the center or the left of the object [2].

B. *Affordance Prediction and Generalization*

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

Accordingly, on the first test the training subset is built by randomly gathering $3/4$ of the vectors obtained with each tool-pose, while the test subset contains the remaining $1/4$ of the vectors. These data is subsequently fed to the SVM classifier to learn the mapping between the set of tool-pose functional features of each training trial and the affordance category to which their corresponding affordance vectors had been clustered. For testing, the SVM classifier is presented only the feature vectors of the testing subset, for which it returns the indexes of the predicted affordance categories.

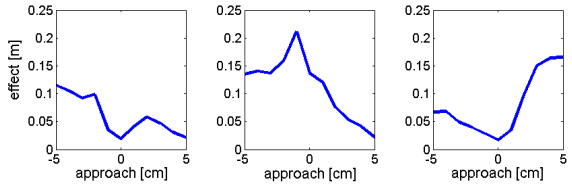
By comparing the indexes predicted by the SVM classifier with the ones that the recorded affordance vectors were assigned to, we can quantify the prediction accuracy. Moreover,



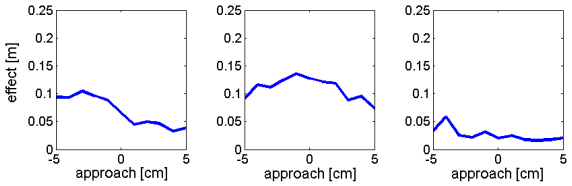
(a) Simulated data (b) Robot data

Fig. 6: The histogram shows the number of times that each possible K was K_{best} based on the Davies-Bouldin index (left: simulated data, right: real robot).

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



(a) Simulated data



(b) Robot data

Fig. 7: Prototype Vectors from the affordances discovered by K-means.

Test	Env.	Class. Acc. (%)	rMSE [m]
Prediction	Sim.	81.9 %	0.064
Prediction	Robot	64.1 %	0.051
Leave-One-Out	Sim.	56.9 %	0.077
Leave-One-Out	Robot	53.9 %	0.054

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

by confronting the prototype affordance of the predicted cluster with the real affordance vector recorded during the experiment, we can graphically assess if the predicted affordance resembles the recorded one, and compute the mean square error from the distance between them. Table I shows the resulting values of accuracy and rooted mean square error (rMSE) obtained on this test with simulated data as well as with data from the real robot.

The second test determines how well the system is able to predict the affordance of previously unseen tool-poses. To this end, we performed a leave-one-out test, whereby data from one tool-pose at a time is used to test an SVM classifier which has been trained with all the data of all the remaining tool-poses. Therefore, this test involves training and evaluating as many SVM classifiers as tool-poses are being considered, so the overall performance displayed in Table I is measured by averaging across them.

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

Results yielded by the leave-one-out test are slightly worse, but still about about 20% over chance, which means that although there is plenty of room for improving the method, it already allows the robot to generalize its knowledge to previously unseen tools.

One of the issues that drove down the performance of the classifiers, other than inaccuracies in the blob extraction and clustering procedures, is class imbalance, meaning that one of the affordance clusters usually dominated over the remaining. This led the classifiers to disproportionally classify feature vectors as corresponding to the dominating affordance category.

Environment	Goal Acc. (%)	Avg. Diff [m]
Simulation	86.51 %	0.064
Robot	86.11 %	0.056

TABLE II: Online Experiment Performance

C. *Affordance based action execution*

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

In this scenario there is no ground truth or labels against which to compute the performance accuracy, so instead we used the 2 following measures to assess the performance. On one hand, the amount by which the goal of displacing the object is achieved is measured by the percentage of trials in which the object was actually displaced (i.e. the distance measured by the tracker is more than 5 cm, to account for tracking error). On the other hand, the error in prediction performance was computed as the average of the absolute difference between the predicted effect of the action and the measured object displacement after the robot's action was performed.

In order to carry out this evaluation, 6 trials were performed with each tool-pose on simulation, while the real robot performed 3 trials per tool-pose, thus having a total of 126 trials in simulation and 36 trials on the real robot. Results, which can be observed in Table II, show that for a large majority of trials, the robot achieves its goal of moving the object with the tool, and does so with an average error very similar to the one achieved in the off-line experiment for affordance prediction on known tools. It is noteworthy that the accuracy in achieving the displacement of the object is higher than the affordance prediction accuracy. This can be supported by the fact that, sometimes, even if the affordance category is wrong, it is usually close enough to the correct one that the maximum values of its prototype vector leads to an action that still achieves the goal.

IV. CONCLUSION

This paper presents a novel approach to the study of tool affordances in robotics, introducing several contributions to the state of the art. In it, we expanded the application of functional features for tool representation beyond the limited sets used before. We applied a comprehensive set of geometrical features extracted from vision which is not task-specific and whose features might relate to a number of different functions, as well as enable generalization among geometrically similar tools. Furthermore, affordances are discovered by evaluating the effects of the robot actions, considering a set of effects generated by a parametrized action as a single affordance entity. This allows the robot to predict not only a single effect for a particular action, but a whole set of them which can be used then to select the best action. Moreover, it accounts for the way in which the tool is grasped when determining its affordances. This is achieved due to the fact that both the affordance vectors and the functional features vary in function on the way in which the tool is grasped, so the learned mapping between them implicitly retains information about the grasp configuration. Finally, the proposed system learns tool affordances without the need for external labeling or supervision for classification, based only on the observation of the effects of tool use.

Yet, we are aware of the many shortcomings of the experiment, which will need to be studied carefully in the future. One of the main drawbacks of the present approach is that whereas it permits the iCub to take in account the grasp pose to select the action that maximizes the desired effect, it does not yet encode the grasp pose explicitly, and so it needs to start with a tool placed in the hand. Future work will tackle this issue as well as introducing features based on 3D representations of the tool, which shall greatly reduce the variability due to perspective.

REFERENCES

[1] J. J. Gibson, "The theory of affordances," in *Perceiving acting and knowing Toward an ecological psychology*, R. Shaw and J. Bransford, Eds. Lawrence Erlbaum, 1977, vol. Perceiving, ch. 3, pp. 67–82.

[2] 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.

[3] 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.

[4] A. Stoytchev, "Robot Tool Behavior: A developmental approach to autonomous tool use," Ph.D. dissertation, Georgia Institute of Technology, 2007.

[5] 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.

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

[7] C. C. Kemp and A. Edsinger, "Robot Manipulation of Human Tools: Autonomous Detection and Control of Task Relevant Features," in *IEEE International Conference on Development and Learning*, 2006.

[8] T. E. Horton, "A Partial Contour Similarity-Based Approach to Visual Affordances in Habile Agents." Ph.D. dissertation, 2011.

[9] 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.

[10] ——, "Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools," *Artificial Life and Robotics*, pp. 1–9, Sep. 2013.

[11] 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.

[12] 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.

[13] T. Hermans, J. M. Rehg, and A. Bobick, "Affordance Prediction via Learned Object Attributes," in *IEEE International Conference on Robotics and Automation (ICRA 2011)*, 2011, pp. 1–8.

[14] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele, "Learned Affordance Cues," Tech. Rep., 2008.

[15] L. Montesano and M. Lopes, "Learning grasping affordances from local visual descriptors," in *2009 IEEE 8th International Conference on Development and Learning*. IEEE, 2009, pp. 1–6.

[16] R. Detry, E. Baseski, M. Popovic, Y. Touati, O. Kroemer, N. Krüger, J. Peters, and J. Piater, "Learning Continuous Grasp Affordances by Sensorimotor Exploration," in *From Motor Learning to Interaction Learning in Robots*. Springer-Verlag Berlin Heidelberg, 2010, pp. 451–465.

[17] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3260–3267, Oct. 2012.

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

[19] 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.

[20] G. Metta, L. Craighero, L. Fadiga, A. Ijspeert, K. Rosander, G. Sandini, D. Vernon, and C. von Hofsten, "A Roadmap for the Development of Cognitive Capabilities in Humanoid Robots," Tech. Rep. 004370, 2010.

[21] 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.

[22] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *International Journal of Advanced Robotic Systems*, vol. 3, pp. 43–48, 2006.

[23] C.-c. Chang and C.-j. Lin, "LIBSVM : A Library for Support Vector Machines," Tech. Rep., 2013.

[24] C. Ciliberto, U. Pattacini, L. Natale, F. Nori, and G. Metta, "Reexamining Lucas-Kanade method for real-time independent motion detection: Application to the iCub humanoid robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4154–4160, Sep. 2011.

[25] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[26] T. Zhang and C. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," *Image Processing and Computer Vision*, vol. 27, no. 3, pp. 236–239, 1984.

[27] D. Zhang and G. Lu, "A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures," Tech. Rep., 2001.

[28] ——, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, Jan. 2004.

[29] R. Hess and A. Fern, "Discriminatively trained particle filters for complex multi-object tracking," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, 2009, pp. 240–247.

[30] U. Pattacini, "Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub," Ph.D. dissertation, 2011.

[31] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1668–1674, Oct. 2010.

[32] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE transactions on pattern analysis and machine intelligenceP*, vol. 1, no. 2, pp. 224–227, 1979.