Exploring affordances and tool use on the iCub

V. Tikhanoff, U. Pattacini, *IEEE Member*, L. Natale, *IEEE Member*, G. Metta, *IEEE Senior Member*

***Abstract*— One of the recurring challenges in humanoid robotics is the development of learning mechanisms to predict the effects of certain actions on objects. It is paramount to predict the functional properties of an object from "afar", for example on a table, in a rack or a shelf, which would allow the robot to select beforehand and automatically an appropriate action (or sequence of actions) in order to achieve a particular goal. Such sensory to motor schemas associated to objects, surfaces or other entities in the environment are called affordances [1, 2] and, more recently, they have been formalized computationally under the name of object-action complexes [3] (OACs). This paper describes an approach to the acquisition of affordances and tool use in a humanoid robot combining vision, learning and control. Learning is structured to enable a natural progression of episodes that include objects, tools, and eventually knowledge of the complete task. We finally test the robot's behavior in an object retrieval task where it has to choose among a number of possible elongated tools to reach the object of interest which is otherwise out of the workspace.**

I. Introduction

Humanoids robots are becoming increasingly complex and, to a certain extent, they can imitate human behavior. One of the great challenges of interacting with a humanoid robot is to develop cognitive architectures with an interface that allows humans to collaborate, communicate and teach robots as naturally and efficiently as they would with other human beings [4]. This line of enquiry follows a two-fold approach by drawing on our knowledge of natural cognition and, simultaneously, by instantiating plausible models of cognitive skills on humanoid robots [5, 6]. The hallmark of cognition, according e.g. to developmental psychologists [7], is the ability to predict the functional behavior of objects and their interaction with the body, simulating and evaluating the possible outcomes of actions before they are actually executed. In the brain, this is thought to happen through the activation of appropriate sensorimotor schemas [8] that effectively function to couple sensory and motor signals in planning action. Sensorimotor schemas have been long posited to be a functional explanation of the brain direct perception of object properties and their role in action execution [1, 9]. The literature abounds of computational and robotic models of such sensorimotor representations as for example the work of Oztop et al. [10], the already cited Montesano et al. [2], Metta et al. [11] and Krüger and colleagues [3] to name a few. In this context the term affordances and object-action complexes (OACs) can be used interchangeably. Tool use is also a direct consequence of the ability to understand affordances and it is believed to share a common representation in the brain [12]. Neuroscience thus suggests an interpretation of the concept of objecthood, which involves measuring and structuring sensorimotor experiences, while associating them with symbolic representations. Roboticists have followed a similar line of enquiry [3]. Affordances can be used in predicting the course of action (and planning) and thus they constitute one of the important ingredients of artificial cognition [2].

In summary, to equip a humanoid robot with the machinery required to learn affordances and tool use we need to prepare a number of basic sensory and motor "innate" skills, e.g. object recognition, shape estimation, etc. We also need to endow the robot with the ability to exploit the interaction with objects to learn affordances. This paper in particular focuses on the description of learning of affordances and tool use. Learning from action demonstration is considered elsewhere [13] as well as the possibility of acquiring new actions via kinesthetic teaching [14, 15]. We further rely on an architecture for teacher-robot interaction that was developed previously [4].

More specifically, our experimental scenario consists of a fully instantiated system integrating perception and learning, capable of interacting in the real world, in real time, and engaged in an object retrieval task which requires choosing among a number of possible tools to reach the object of interest. The choice of the correct tool requires understanding affordances. A longer term goal of this research is to embed the robot with enough knowledge of affordances to be able to apply sophisticated planners [16] to achieve the task no matter what tools are available (or not available) to the robot.

The paper is organized as follows. Section II contains a brief description of the experimental setup. Section III provides a detailed description of the basic components that make up the experiments (see Fig. 1). We divided them into three main areas being these the visual, the motoric and the learning algorithms respectively. Section IV describes the affordance learning methods: i.e. what signals and how they are extracted and combined. Section V presents the experimental results and, finally, we draw the conclusions and discuss future work in section VI.

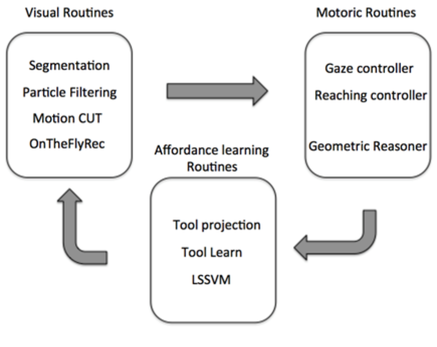


Figure 1: General overview of the iCub modules used in the experiments.

*This work was supported by the European FP7 ICT project No. 288382 (POETICON++), No. 270490 (EFAA) and No. 270273 (Xperience).

V. Tikhanoff, U. Pattacini L. Natale and G. Metta are with the iCub Facility, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy (email: vadim.tikhanoff@iit.it, ugo.pattacini@iit.it, lorenzo.natale@iit.it and giorgio.metta@iit.it)

II. Experimental setup

The experimental setup is the Open Source iCub humanoid robot [5]. For the scenario described in this paper we utilize only movements of the upper body, i.e. 41 degrees of freedom. The robot is equipped with cameras, microphones, force-torque sensors, inertial sensors, joint angle sensors and a full-body skin (2000 sensing points on the upper body). A small cluster of about 30 cores and a few GPUs is used to parallelize sensory processing. In the experiments described in the following, vision is the main source of sensory information and therefore it is detailed more extensively. Several basic features are always in use on the iCub, as for example, impedance control of the arms, contact detection (e.g. with the table), etc. which use various combinations of its sensors.

The robot interacts with objects placed on a table whose height is estimated online via tactile/force exploration. Knowing that the objects are on the table allows reaching and touching objects using 2D visual information only (a single camera).

On the software side, the iCub employs YARP, a thin middleware that enables inter-process communication across a network of machines [17]. YARP defines modules (executables) which collaborate in implementing a given behavior on the iCub. In the following we often name the modules explicitly (they are marked in *italic*). These can be found in the iCub SVN repository and downloaded from:

https://svn.code.sf.net/p/robotcub/code/trunk/iCub/contrib/src/misc/[modulename]
https://svn.code.sf.net/p/robotcub/code/trunk/iCub/contrib/src/[modulename]
https://svn.code.sf.net/p/robotcub/code/trunk/iCub/main/src/modules/[modulename]

III. iCub behavior Routines

A. Visual routines

1) Segmentation with fixation

The ability to automatically segment an object of interest or parts of salient locations of various sizes from its background is a valuable asset for a humanoid robot to interact effectively in the real world. Attention plays an important role in the human visual system and has been extensively considered and investigated in the computer vision literature [18]. The human eyes make a series of fixations at distinctive locations in the scene (about three per second). These saccadic movements typically bring the fovea inside some particular region of interest whether on an entire object or on part of it. The fixation point becomes the seed of the segmentation procedure. We center a polar conversion at the fixation point and apply an approximate graph partitioning method [19] to search a closed contour that includes the fixation point. The algorithm consists of two steps: (1) a probabilistic boundary edge map is generated by merging the available visual information (e.g. intensity, color) and (2) the optimal contour is constructed by selecting the closed contour that minimizes a cost based on the continuity and overall shape of the candidate contours in the edge map. The segmentation algorithm is described in details in the paper by Mishra et al. [20] and an implementation is available in the iCub repository as the *activeSeg* module.

We use the *activeSeg* module to segment the object of interest from its background and use the segmented region as a template to initialize a particle filter (see next section). Tracking is particularly useful in consideration of analyzing the behavior of the objects while the iCub interacts with them. In addition, the module provides information such as the orientation of the object and its principal axes.

An example of this segmentation can be seen in the following Fig. 2.

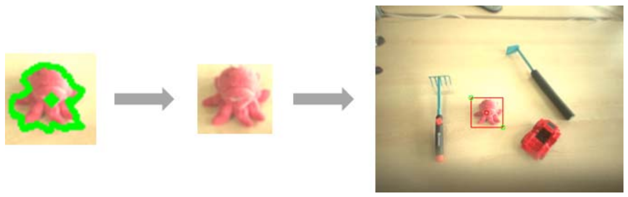


Figure 2: Example of segmentation of an object and corresponding bounding box. Left: segmented contour remapped in the original image (from polar); middle: the seed of the particle filter; right: image tracking.

2) Tracking using particle filtering

Visual object tracking is crucial for an interactive robot and lately the use of particle filtering has been particularly successful. Once initialized on an object, a particle filter tracker maintains a probability distribution over the condition of the tracked object (location, size, etc.). This distribution is a set of weighted samples representing a hypothesis of the possible location of the object. A set of Bayesian filters is used to propagate the particles through time (and consequently the probability distribution). This probability distribution through time is used to compute the location and trajectory of the tracked object by estimating its moments at each time step. The suggested tracking algorithm is described fully in [21] and an implementation is available in the iCub repository as the *templatePFTracker* module. The trajectory of the object is one of the input signals to the affordance learning modules (cf. section III.C).

3) Independent motion detection

In order to autonomously acquire and extract information about the tool dimensions, the iCub needs to explore the modification of the kinematics due to the addition of the tool (the tool is always in the iCub hand). This procedure is based on the detection of the movement of the tool in the images and in particular it is based on the *motionCUT* module [22]. The *motionCUT* algorithm is an elaboration of the Lucas-Kanade optical flow algorithm [23] to detect moving objects irrespective of the egomotion produced by the camera movement. *MotionCUT* is purely visual and therefore it does not require additional information apart from the images coming from the cameras. The output of *motionCUT* is then utilized in a post-processing stage that detects the tip of the moving tool as described in section IV.B.1.

4) Object recognition

In a natural robotic setting we cannot expect data for object recognition to be fully labeled and accurately prepared beforehand. We seek thus methods that work well in real-time, where learning happens incrementally as new samples become available. We collect training data through weak supervision provided by a human teacher who indicates through speech the class label to each object and shows the objects to the robot by either pointing or moving it. The iCub then uses the *motionCUT* and the *activeSeg* modules for segmenting the object and associating the class label provided.

Visual recognition is implemented after Yang et al. [24], which employs local features (SIFT) augmented through a sparse coding stage, and classified with a linear support vector machine. The sparsification stage ensures high quality results in terms of accuracy even if simple linear models are used in the classification stage. The full implementation is described further in Fanello et al. [25]. The module in the iCub repository is called *scspmClassifier*. A full characterization of the method performance is available in the papers of Fanello et al. [25, 26].

The recognition of objects and their 2D shape extracted from segmentation are one of the basic components of the representation of affordances. Clearly the iCub needs to distinguish the available tools, associate their functionalities to each object in order to later "use" them appropriately. An example of the object recognition results is shown in the following Fig. 3.

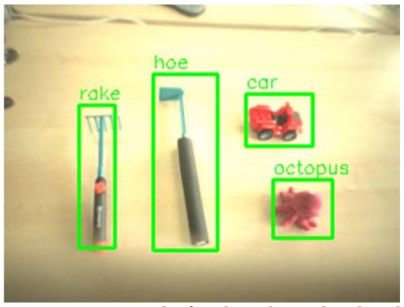


Figure 3: Object recognition results for object located in the table in front of the robot.

B. *Motoric routines*

1) *The reaching controller*

Reaching is implemented by determining the 3D position of the target object and subsequently computing a suitable trajectory to bring the hand in contact with the object. We do not use visual servoing at the moment and thus the system relies on accurate calibration of the camera rig and of the iCub kinematic parameters. The overall position accuracy is within 10mm which is sufficient for the purpose of this work. The iCub module that implements the inverse kinematics and trajectory generation is called *Cartesian Interface* [27]. The inverse kinematics computation is implemented as follows:

$$q^* = \arg\min_{q \in \mathbb{R}^n} \|x_d - F(q)\|^2 \qquad (1)$$

taking as input the target $x_d \in \mathbb{R}^6$ in Cartesian space, comprising three coordinates for the position and three coordinates for the orientation in axis-angle notation. The output is the joint configuration $q^* \in \mathbb{R}^n$, being F the forward kinematic map of the manipulator and n the number of degrees of freedom. Equation (1) is optimized by using the *Ipopt* solver [28], a fast and reliable state-of-the-art nonlinear optimization algorithm. At this stage, a number of linear and/or nonlinear constraints such as the joint limits and the prioritization between reaching in position with respect to reaching in orientation are added to equation (1) (see [27] for the complete formulation utilized in the experiments).

Having determined the final configuration, the trajectory is generated following the *multi-referential* approach of Hersch et al. [29]. Two trajectories – one in joint space and another in task space – are generated and synchronized allowing a smooth combination of joint space movements when close to singularity with straight path whenever possible (e.g. far from singularities).

At the user level, reaching can be evoked with a simple call to **goToPose**(x_d) method, where x_d represents the desired pose of the robot's hand. A call to the **attachTipFrame**(x_tip) method with $x_{tip} \in \mathbb{R}^6$ allows the user to specify a new end-effector for the *Cartesian Interface* module (roto-translated from the hand of the iCub). This is used each time a tool is in the hand. The kinematic description of the robot effector is extended by including the tool dimensions and orientation and, consequently, also the inverse kinematics takes it into account.

2) *Geometric reasoning*

The *Cartesian Interface* can be employed to access the functionality of the inverse kinematics solver. The user can query the interface to find the most suitable solution of a reaching task without actually moving the robot. This is encoded by the **(x,q)=askForPose**(x_d) method; the returned pair (x, q) is such that $x = F(q)$ and $D = \|x_d - x\|^2$ is minimum as per (1). The evaluation of the residual distance D can be used to reason about the reachability of the target using various tools. Let $T_1, T_2 \dots T_m$ be the set of m tools that the iCub is provided with, whose dimensions $x_1^T, x_2^T \dots x_m^T$ with respect to the hand frame of reference have been learned according to the algorithm discussed in section IV.B.1. The procedure for selecting the correct tool is thus:

```
$k$=selectTool $(x_d, x_1^T \dots x_m^T)$
for i=1:m
    attachTipFrame $(x_i^T)$;
    $(x_i, \sim)$= askForPose$(x_d)$;
    $D_i = \|x_d - x_i\|^2$;
end
$k = \arg\min_{i=1\dots m} D_i$
```

Tool Selection algorithm.

Essentially, given the desired final position of the object x_d, a test is run for each tool T_i to retrieve the resulting pose x_i of the tool tip as solution of the inverse kinematics problem, considering the frame x_i^T as an extension to be attached to the robot hand. The corresponding cost measure D_i is computed, accounting for how close the robot can perform reaching with that tool. Finally, the tool T_k with the lowest cost measure is chosen. As *Ipopt* is fast in solving the problem in (1), the tool selection algorithm takes on average about 30ms per cycle. The whole geometric reasoning procedure is practically instantaneous in comparison to the average task duration.

3) *The gaze controller*

Another important motor component of this work is the gaze controller, which is responsible for driving the robot's head joints to track visual stimuli or to saccade to locations of interest in the scene. For example, during the exploration of the objects, the robot is instructed to gaze the object as it moves due to the application of a random pushing action; on the other hand, when the robot is asked to learn the tool dimensions, attention will focus on the optical flow generated by the tool tip as it is moved by the robot. To achieve tracking effectively and robustly, we use the *iKinGazeCtrl* module available from the iCub repository [30]. *iKinGazeCtrl* uses the same inverse kinematics formulation of equation (1) customized to the head kinematics. Interestingly, this controller can realistically mimic human-like movements with rapid eyes movements and compensation of neck rotation while, at the same time, providing adequate tracking performance of fast moving objects. Similarly to what discussed in the previous section, this module exposes an interface that allows the user to control the gaze both in operational space via a call to **lookAtFixationPoint(x_{d})** and in image planes through the **lookAtPixel(u,v)** service. Importantly *iKinGazeCtrl* computes also the Cartesian position of objects detected in the image planes by triangulation or by resorting to the simpler monocular approach of the **x=get3DPointOnPlane(u,v,eqplane)** service, which assumes that the object lies on a table defined by **eqplane**.

C. Learning affordance routines

Learning affordances from the ground up, without taking into account any *a priori* model, can be a daunting task due to the high dimensionality of the space to be explored. This is due to the fact that affordances incorporate simultaneously relations between motoric and perceptual signals which in themselves are large. We impose certain priors to simplify the problem and for the time being work on a well-defined subset of possible affordances related to the action of pushing, pulling and moving. Objects are also constrained to lie on the table.

In the following we use a simple Least Square Support Vector Machine (LSSVM) to regress the applied action (e.g. push object x from direction θ) to the object displacement (change of state). More details are provided in section IV.B.1. In particular, we employ an incremental variant of the LSSVM that approximates the Gaussian kernel with a large but finite number of Fourier components [31]. This allows the construction of an incremental kernel-based machine whose computational cost depends only on the number of Fourier components which can be set *a priori* depending for example on the computational budget. A standard kernel-based machine employing the exact Gaussian kernel would increase its computational cost as a function of the number of examples. LSSVM has few hyper-parameters (regularization term λ and the inverse of the variance of the Gaussian kernel γ) which can be optimized beforehand on a number of test runs using e.g. grid search. We refer the reader to [31] for a complete treatment of the method.

IV. Learning Affordances and Tools

A. Learning rolling affordance of objects

The rolling affordance is meant to represent the ability of an object to react to a tapping action by rolling over or simply translating of a certain distance. Thereby, objects generally tend to respond differently with respect to the direction of the tap: for example a toy car will move considerably farther if hit by the robot from its front or its rear rather than from the side; conversely, a cylindrical object lying on a table will cover a larger distance if pushed from a direction perpendicular to its principal axis. The idea is thus to let the iCub freely explore this affordance by tapping the object from a random direction. The intensity of the tap is controlled by the robot and the resulting behavior of the object is observed visually. Action and effects can thus be used for learning a map that relates the tap angle to the translation of the object. The angle θ is calculated between the line connecting the tap point selected by the robot (at random) with the object center and one of the principal axes of the object as given by the *activeSeg* module (cf. section III.A 1), which extracts the moments of the corresponding segmentation (Fig. 4). The tap is generated by using the *Cartesian Interface* as two consecutive reaching movements, whose relative spatial displacement and hand orientation are predefined parameters.

The effect of the tap is measured as the distance d traveled by the object – in meters – once it has been pushed; d can be easily obtained by comparing the initial 3D position x_i of the object located on the table with its final position x_f. The object is tracked by the gaze controller, which constantly keeps it in fixation by receiving the object location in the image planes from the *templatePFTracker* module; finally triangulation (**get3DPointOnPlane()**) is used to compute the Cartesian positions of the object.

Having a number of samples of θ and d, the learnt map $d = M_{\mathrm{roll}}(\theta)$ codes the rolling affordance of an object when a push is exerted from a given direction. M_{roll} is learnt by means of the LSSVM described in III.C.

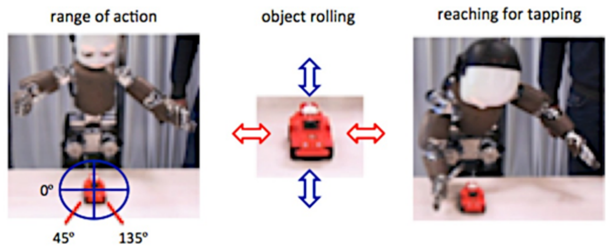


Figure 4: Experimental setup for the learning of the object rolling affordances.

B. Learning pulling affordance of tools

The *pulling affordance* refers to the property of a tool to be used to pull an out-of-reach object and eventually allow grasping it successfully. The use of a tool for dragging objects intrinsically depends on the shape of the tool tip and how it gets in contact with the object, and therefore entails two main aspects: (1) the ability of the robot to discover the relative position of the tool tip with respect to its standard effector in order to use the tip in place of the hand to execute actions and (2) the possibility to learn the pulling affordance of the tool tip by exploring the action-effect relation similarly to what has been done in section IV.A.

1) Discovering the tool size

The iCub estimates the 3D position of the tool tip relative to its effector by waving the tool held in the hand, thus generating movement in the cameras images. Movement is used both as a cue for tracking (gaze controller) and as input to the *motionCUT* module (cf. section III.A.3). The measure of the tool tip in the image plane (Π) is reasonably precise (Fig.5) and it is retrieved as the topmost corner of the motion blob principal axis. The final accuracy benefits significantly from the fact that *motionCUT* produces blobs that are consistent in space and time [22].

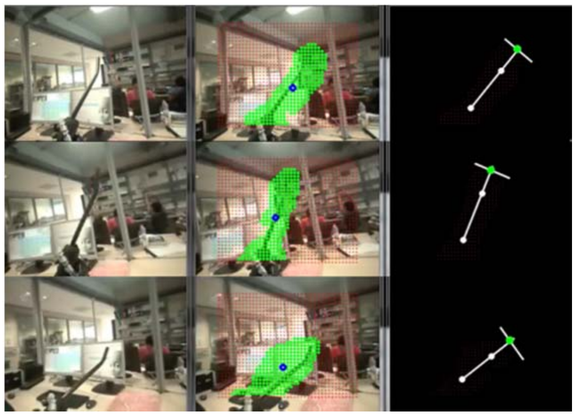


Figure 5: Explorative behavior for discovering the tool dimensions. Left: images acquired from the robot's camera; Middle: the output of the *motionCUT* module depicting the independent motion blobs; Right: extraction of the tool tip projection (green dot).

Given $\widehat{P}\,(x,y,z)$ the estimate of the tool tip position, we seek (Fig. 6) to collect a number of observations (k) during the waving motion and compute the error $e_k = \pi_k(\hat{P}) - \Pi_k$, where $\pi_k(\cdot)$ represents the pin-hole camera projection of a 3D point in the image plane taking also into account the camera position in space obtained from the kinematic configuration of the robot sampled at the instant t_k. We determine the tool size as the solution to the following minimization problem:

$$P^* = \arg\min_{x,y,z}\left(\frac{1}{2N}\sum_{k=1}^{N}\|e_k(x,y,z)\|^2\right), \qquad (2)$$

where N is the total number of observations. This minimization can be conveniently performed as before with the *Ipopt* optimizer.

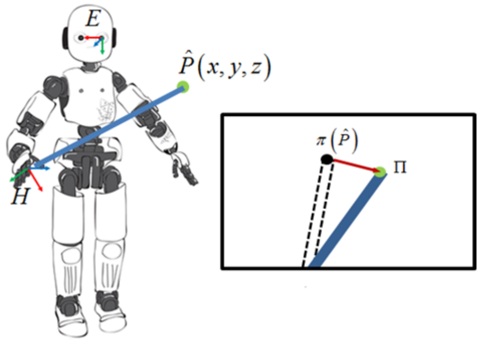


Figure 6: Left: the pictorial representation of the robot holding a tool in its hand. Right: a sketch showing the projection of the tool in the camera image plane and the error displacement (red line) between the estimation and the measurement.

2) *Exploring Tool Use through Pulling Action*

The exploration of the pulling affordance of a specific tool (and for a specific object) consists of finding the correct position (relative to the object) to place the tool tip with the purpose of enabling a successful dragging action. It is clear that the desired position is a function of the tool dimension (especially its length if we consider elongated tools) and the shape of the tool tip. For instance, a rake can be properly placed with its tip aligned with the object center due to its symmetry with respect to longitudinal axis (Fig. 8), whereas a hoe with the paddle parallel to the table has to be put slightly to the right (left) if held in the right (left) hand due to its L-shaped tip (Fig. 8). The learning procedure simply executes the steps described in section IV.A with some minimal modifications. To reduce the complexity of the exploration, we sample the space of the initial Cartesian positions by testing points located on a circle centered on the object with a radius r. The input is the angle θ and the dragged distance d represents the output (the consequences) of the pulling test. Also in this case, we can finally learn (through the LSSVM algorithm) the map $d = M_{\text{pull}}(\theta)$ accounting for the predicted distance covered by the object when dragged with the tool tip initially placed in the relative angular configuration θ. The pulling affordance will be found as that particular configuration $\theta_{\max}$ for which M_{pull} is maximized, which essentially tells us how to arrange the tip to obtain the largest object displacement.

V. Experiments

In the following we report on the experiments carried out to validate the methodologies discussed in the previous sections.

A. *Learning rolling affordance of objects*

We put to test the procedure to learn the rolling affordance with the three objects shown in Fig.7: a toy car, a cylindrical tube of colored pins and a small ball.

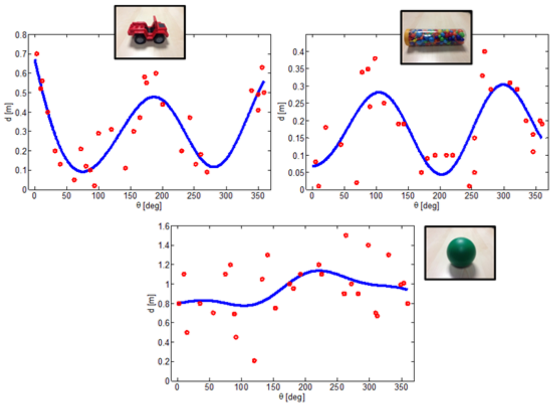
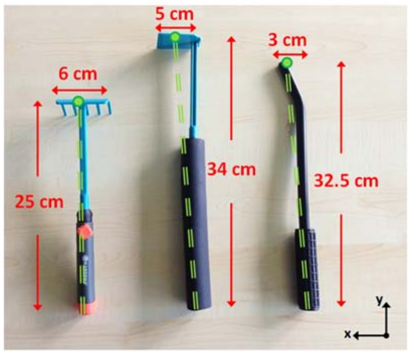


Figure 7: The maps of the rolling affordances for the toy car, the cylindrical tube and the small ball: red dots represent the sampled pairs of input angle θ and the output distance d; in blue the predictions provided by the learned maps M_{roll} over the whole input space.

These objects are special as for example the affordance of the car and the tube are dual: the toy car rolls over along the front-rear axis while the latter is more sensitive to the left-right orientation. Finally, the ball does not have a preferred direction of movement and thus can be used as a challenging test. About 30 trials per object are sufficient to approximate the expected affordances quite well, as shown in Fig. 7: the whole span of 360 degrees of the angular input θ (computed relatively to the orientation of the principal axis) is randomly sampled and the resulting rolled distances d are acquired online. The LSSVM hyper-parameters λ and γ used throughout all of the experiments were set to 100 and 10, respectively. The maps M_{roll} built to fit these data (Fig. 7-right) shows that the car has two peaks centered around 0 and 180 degrees accounting for the front-rear direction, whereas the second map correctly indicates that the tube would most naturally roll over if hit approximately along the directions specified by $\theta_1 = 90$ and $\theta_2 = 270$ degrees, i.e. orthogonal with respect to the car's affordance. Differently, data collected for the ball are much more sparse and affected by more noise since the friction with the table's surface is reduced compared to the first two cases and it is considerably difficult to provide a consistent repeatability of the tapping actions; for this reasons, the related map does not reveal maxima that are clearly separated in distance, but rather approximates a flat distribution. Nonetheless, this turns out to be a good hint of the fact that, as anticipated, the ball object does not show any preferred rolling direction. Once learning has been achieved, the robot can be asked to tap the objects from the correct orientation (and does it without error unless object recognition fails).

B. Discovering tools sizes

The tools we employ in our experiments are a rake, a longer hoe and finally a stick-like tool, as illustrated in Fig. 8. By doing this, the functionality of the tools will differ in the scenario where the pulling affordance will be learned. Thereby, the robot is asked to autonomously discover the positions of the tool tips given in terms of coordinates in the frame attached to the hand. By applying the method described in section IV.B.1, the iCub can acquire not only the actual shape of the tool, but also the location of the centroid of the tool tip (highlighted with a green dot in Fig. 8). It turns out that the exploration works well since the tip positions retrieved by the algorithm correspond to the real tool size. The variance in the table is computed over 10 trials per tool, as summarized in Fig. 8.



	x [cm]	**y** [cm]	**z** [cm]
rake	0.3±0.6	25.9±2.1	2.2±0.3
hoe	3.1±0.7	36.1±2.8	3.1±0.2
stick	1.3±0.6	33.5±1.8	2.9±0.3

Figure 8: Top: the rake, the hoe and the stick employed by the robot as tools. Bottom: the average positions along with the standard deviations of the tools tip relative to the robot hand as determined by the exploration.

Interestingly, as a proof of concept, we replicate the rolling affordances experiment by letting the robot hold the stick-like tool (resorting to the `attachTipFrame()`) and do the required tapping action with the learned maps. As in the previous case, iCub performs the task correctly.

1) Exploring Tool Use

With the tool tightly in the hand, the tool's dimensions precisely estimated, the iCub is now equipped with the information needed to extend its kinematics, considering the tip as the new effector. It is then possible to explore the functionality of the tip's shape when the tool is brought in contact with an out-of-reach object in order to pull it closer to the robot body. The results of this exploration described in section IV.B.2 are reported in Fig. 9. The pulling maps $M_{\text{pull}}^{\text{rake}}$and $M_{\text{pull}}^{\text{hoe}}$ of the rake and hoe tools, respectively, are still recovered from a very few samples, representing how the related tips should be put with respect to the object centers, to eventually obtain the best drag: as expected, the maps predict that the rake will work better when put roughly on top of the object ($\theta_{\max}^{\text{rake}} = 95$ deg), whereas the hoe is required to be placed a bit to a side (on the right, $\theta_{\max}^{\text{hoe}} = 83$ deg) to exploit its paddle, when it is held with the right hand and finally the stick-like tool shows a significantly lower performance because the tip does not have any hooking capabilities (as expected).

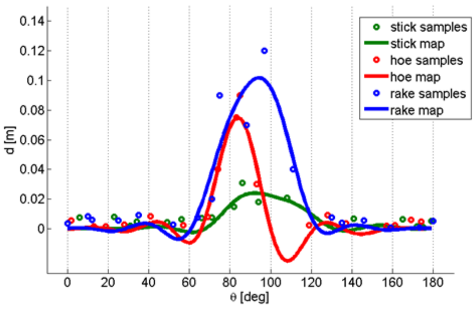


Figure 9: The maps retrieved for the pulling affordance with tools.

At this point we can take advantage of the geometric reasoning (section III.B.2) to employ the maps M_{pull}^{*} with the aim to let iCub grasp far-off objects by pulling them near via tool use. The task is accomplished according to the steps exemplified in Fig.10. Once the object is segmented and recognized to be too distant for a direct grasp (to this end the reasoner can be run without a tool), the robot can identify the presence of known tools on the table by means of the *scspmClassifier* module, and then start analyzing "mentally" whether it can pursue the action by using the rake or the hoe. To do that, for each tool i, the corresponding pulling affordance θ^i is determined such that $M_{\text{pull}}^{i}(\theta^i)$ is maximum. The spatial position x_d^i can be then related to the angular quantity θ^i based on the fixed distance r from the object. Finally, the geometric reasoner executes the tool selection algorithm with x_d^i as input data, providing the tool selection by assessing the error measures D^i.

C. Putting everything to use

The testing phase, reported in this section, consists of presenting a few objects to the iCub, such as the previously learned tools, rake and hoe, and a plush toy (octopus). The tools are positioned in front of the robot but the octopus was carefully positioned far enough from his reaching space see Fig.11 (a). The goal at this stage is to grasp the octopus by using all the available objects/tools at the robot's disposal.

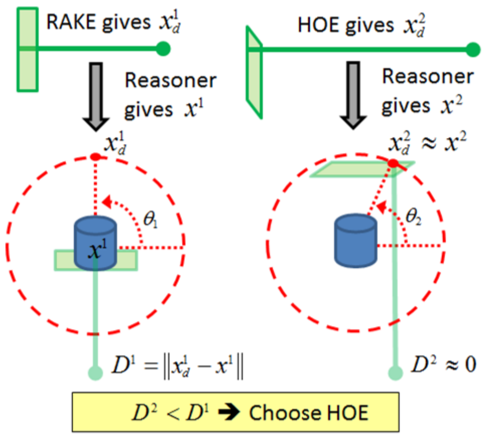


Figure 10: Example of the geometric reasoner selection procedure.

After the instruction is given, the robot recognizes that the octopus is far from his reach, and reasons about which object/tool allows accomplishing the task. At this stage the robot asks the user to provide the rake Fig.11 (b). Once the tool is in the robot's hand, it will reach with the correct orientation for the octopus Fig.11 (c), and do the pulling action until the object is within grasping distance Fig.11 (d). At this stage, which is outside of the scope of this paper, the robot returns the tool, and reaches for the octopus with its hand Fig.11 (e) and finally grasps it correctly Fig.11 (f).

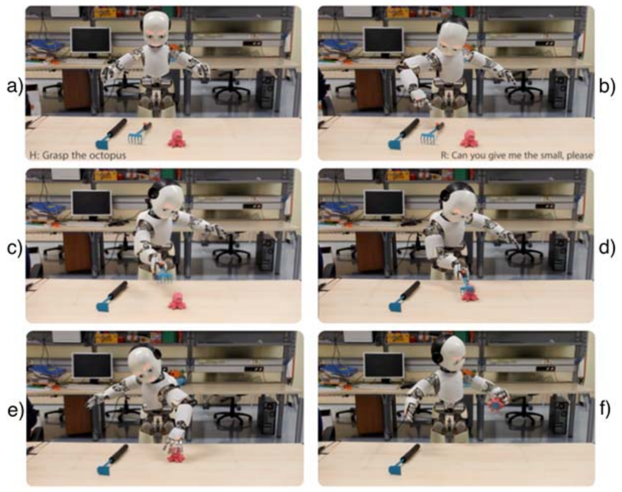


Figure 11: Tool use experiment: (a): The instruction to grasp the octopus is given to the robot. (b): After reasoning about which tool to use, the robot asks to be given the required one. (c): The iCub reaches for the octopus with the tool. (d): The iCub pulls the octopus towards itself (e): The iCub reaches for the octopus with its hand. (f): The iCub is able to successfully grasp the object of interest.

VI. Conclusions

This paper described a complete framework for the iCub robot to learn rolling affordances of object, explore handheld tools, learn how to use them and finally put the learned skill to use. The presented experiments give an emphasis to the hypothesis that learning to act on objects is a twofold exercise: one aspect is to sharpen the robot's skills in order to interact in a natural way with objects and the second one is to acquire enough knowledge in order to interpret the actions of others [32].

So far, we focused on the realization of the complete system trading the generality of the single modules. Some of them are particularly simple and clearly too limited for future uses. In other cases, we were missing altogether the possibility of teaching the robot directly (let us imagine a set of non-trivial affordances). The latter problem can be quickly fixed since the initial trajectory can be taught to the iCub via kinesthetic teaching and represented using e.g. DMPs [15]. Furthermore, policy search reinforcement learning methods seem particularly apt to attack the problem of refining the controller of the motoric part of the affordance representation [33]. Similarly, we did not address the problem of generic affordance exploration although there are various possibilities for generalizing the approach as for example in the work of Montesano et al. [2].

In addition, the iCub in this experiment performed only simple stereotyped grasping of the object. This can be improved and in fact there is on-going work toward the implementation of a full-fledged grasping pipeline [34].

Although not particularly stressed in this paper, it is worth noting that the interaction of the iCub with the teacher becomes fundamental to learn new objects and tools as well as to receive useful "hints" on the plausible affordances or combination of them. The method of Lallee et al. [4] is a good candidate to manage the acquisition of complex tasks on the iCub. Finally, the reasoning abilities of the iCub presented here are too limited (almost trivial). They were in replacement of more sophisticated methods which are under investigation for this specific case of pulling, pushing, moving affordances [16].

References

[1] J. J. Gibson, "The theory of affordances," in *Perceiving, acting and knowing: toward an ecological psychology*, R. Shaw and J. Bransford, Eds., ed Hillsdale: Lawrence Erlbaum, 1977, pp. 67-82.

[2] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory Motor Maps to Imitation," *IEEE Transactions on Robotics, special issue on Bio-Robotics,* vol. 24, 2008.

[3] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrcen, A. Agostini, and R. Dillman, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems,* vol. 59, pp. 740-757, 2011.

[4] S. Lallee, E. Yoshida, M. A., F. Nori, L. Natale, G. Metta, F. Warneken, and P. F. Dominey, "Human-Robot Cooperation Based on Interaction Learning," in *From Motor Learning to Interaction Learning in Robots*. vol. 264, O. Sigaud and J. Peters, Eds., ed: Springer-Verlag, 2010.

[5] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, and J. Santos-Victor, "The iCub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks,* vol. 23, pp. 1125-1134, 2010.

[6] D. Vernon, G. Metta, and G. Sandini, "A Survey of Cognition and Cognitive Architectures: Implications for the Autonomous Development of Mental Capabilities in Computational Systems,"

IEEE Transactions on Evolutionary Computation, special issue on AMD, vol. 11, April 2007 2007.

[7] C. von Hofsten, "An action perspective on motor development," *Trends in cognitive sciences,* vol. 8, pp. 266-272, 2004.

[8] V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti, "Action recognition in the premotor cortex," *Brain,* pp. 593-609, 1996.

[9] M. A. Arbib, "Perceptual Structures and Distributed Motor Control," in *Handbook of Physiology*. vol. II, Motor Control, V. B. Brooks, Ed., ed: American Physiological Society, 1981, pp. 1449-1480.

[10] E. Oztop, M. Kawato, and M. A. Arbib, "Mirror neurons and imitation: A computationally guided review," *Neural Networks,* vol. 19, pp. 254-271, 2006.

[11] G. Metta and P. Fitzpatrick, "Early Integration of Vision and Manipulation," *Adaptive Behavior,* vol. 11, pp. 109-128, 2003.

[12] S. T. Grafton, L. Fadiga, M. A. Arbib, and G. Rizzolatti, "Premotor cortex activation during observation and naming of familiar tools," *Neuroimage,* vol. 6, pp. 231-236, November 1997 1997.

[13] I. Gori, S. R. Fanello, F. Odone, and G. Metta, "A Compositional Approach for 3D Arm-Hand Action Recognition," presented at the IEEE Workshop on Robot Vision (WoRV), Clearwater, Florida, USA, 2013.

[14] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering Optimal Imitation Strategies," *Robotics and Autonomous Systems, Special Issue: Robot Learning from Demonstration,* vol. 47, pp. 69-77, 2004.

[15] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems,* vol. 60, pp. 1327-1339, 2012.

[16] K. Mourao, L. Zettlemoyer, R. Petrick, and M. Steedman, "Learning STRIPS operators from noisy and incomplete observations," presented at the 28th Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, 2012.

[17] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: yet another robot platform," *International Journal on Advanced Robotics Systems,* vol. 3, pp. 43-48, March, Special Issue on Software Development and Integration in Robotics 2006.

[18] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active Vision," *International Journal of Computer Vision,* vol. 1, pp. 333-356, 1988.

[19] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," in *Energy minimization methods in computer vision and pattern recognition*, ed: Springer, 2001, pp. 359-374.

[20] A. Mishra, Y. Aloimonos, and C. Fermuller, "Active segmentation for robotics," presented at the IEEE/RSJ international conference on intelligent robots and systems, St. Luis, MO, USA, 2009.

[21] R. Hess and A. Fern, "Discriminatively trained particle filters for complex multi-object tracking," presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, Florida, USA, 2009.

[22] 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," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2011), San Francisco, CA, USA, 2011.

[23] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," presented at the 7th International Joint Conference on Artificial Intelligence (IJCAI), Vancouver, BC, Canada, 1981.

[24] J. J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," presented at the International Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA, 2009.

[25] S. R. Fanello, C. Ciliberto, L. Natale, and G. Metta, "Weakly Supervised Strategies for Natural Object Recognition in Robotics," presented at the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013.

[26] S. R. Fanello, N. Noceti, G. Metta, and F. Odone, "Multi-Class Image Classification: Sparsity Does It Better," presented at the International Conference on Computer Vision Theory and Applications (VISAPP), Barcelona, Spain, 2013.

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

[28] Y. A. Wätcher and L. T. Biegler, "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical-Programming,* vol. 106, pp. 25-27, 2006.

[29] M. M. Hersch and A. G. Billard, "Reaching with multi-referential dynamical systems," *Autonomous Robots,* vol. 25, pp. 71-83, Nov 14, 2007 2008.

[30] U. Pattacini, "Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub," PhD dissertation, RBCS, Italian Institute of Technology, Italian Institute of Technology and University of Genoa, Genoa, 2011.

[31] A. Gijsberts and M. G., "Real-Time Model Learning using Incremental Sparse Spectrum Gaussian Process Regression," *Neural Networks,* vol. 41, pp. 59-69, 2013.

[32] P. Fitzpatrick and G. Metta, "Grounding vision through experimental manipulation," *Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences,* vol. 361, pp. 2165-2185, 2003.

[33] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks,* vol. 21, pp. 682-697, 2008.

[34] N. Vahrenkamp, M. Kröhnert, S. Ulbrich, T. Asfour, G. Metta, R. Dillmann, and G. Sandini, "Simox: A Robotics Toolbox for Simulation, Motion and Grasp Planning," presented at the International Conference on Intelligent Autonomous Systems (IAS), Jeju Island, Korea, 2012.