A framework for tool cognition in robots without prior tool learning or observation

Keng Peng Tee, Samuel Cheong, Jun Li, Gowrishankar Ganesh

A framework for tool cognition in robots without prior tool learning or observation

Keng Peng Tee[1*], Samuel Cheong[1], Jun Li[1], Gowrishankar Ganesh[2*]

[1]Institute for Infocomm Research, A*STAR, Singapore 138632
[2]LIRMM, Univ. Montpellier, CNRS, France
*To whom correspondence should be addressed.
Email: teekengpeng@gmail.com, ganesh.gowrishankar@lirmm.fr

Abstract

Human tool use prowess distinguishes us from other animals. In many scenarios, a human is able to recognize objects, seen for the first time, as potential tools for a task, and use them without requiring any learning. Here we propose a framework to enable similar abilities in robots. We first characterized human tools to identify a special category of tools that humans are able to use immediately through a process of skill transfer from their limbs, rather than tool learning. Motivated by the tool characterization and our neuroscientific studies on human tool use and embodiment, we then developed a tool cognition framework that enables a robot to recognize a previously unseen object as a tool for a task, plan how to grasp and wield the tool in the face of constraints and obstacles, before finally performing the task with the tool. Furthermore, the framework allows for flexibility in tool use, where the same tool can be adapted for different tasks, and different tools for the same task, all without any prior learning or observation of tool use. We demonstrate the possibilities offered by our tool cognition framework in several robot experiments with both toy and real objects as tools.

1 Introduction

In his seminal work on apes in the early 1900s, Wolfgang Koehler wrote about his tool-using chimpanzee: *"When the bananas are hung out of reach on the smooth wall of the house, he (the chimpanzee) takes a green plant –stalk, then a stone, a stick, a straw, his drinking bowl, and finally a stolen shoe, and stretches up towards the fruit; if he has nothing else at hand, he takes a loop of the rope to which he is attached and flaps it at the bananas"* [1].

The reason this account of the chimpanzee amazes us is that we realise that the chimpanzee could recognize objects around it as potential reaching tools, just like a human would. But how do humans and the chimpanzees do this? How can we intuitively recognize and use objects in our environments as tools to accomplish tasks? Similar tool cognition can be of tremendous benefit for robots, and enable them to become truly autonomous and successful in unstructured environments.

For this reason, several studies have attempted to develop tool use in robots. However, first time tool use is popularly seen as a learning problem in robotics, where a robot learns the movements and strategies possible with a tool by exploration [2]-[10], or demonstration/observation [11, 12, 13], similar

to the learning of tasks without tools. On the other hand, humans often use objects immediately as tools, without requiring to learn how to use them (from scratch). When a coin rolls under the sofa, you are able to intuitively recognize an elongated object like a *cricket bat*, that you may have never seen before, as a potential tool to retrieve the coin. And are also able to pick it up and immediately use it as such.

Arguably, to perform this tasks, we utilize the knowledge we have accumulated about the task from our previous skills with and without tools. Again previous studies have suggested tool affordance learning [14]-[17] or transfer learning [18, 19, 20, 21, 22, 23, 24, 25, 26, 27] to utilise learning from *some* instances of tools, before extending to other tools. However, here we show that even this is not required for a large category of tools. We propose a tool cognition framework that enables robots to use tools with absolutely *zero* previous experience in tool use or tool use observation. It allows robots to extend their prior skills *without tools* for both, *recognizing* as well as *using* previously unseen objects in their environments as tools, without any new learning.

The first intuition for our tool cognition framework was derived our recent characterization of human tools into 3 categories [30] depending on the tasks they serve and the actions required to use them.

We proposed as *Category I* tools, tools which help amplify/augment kinematic or dynamic features of functions that are already in a human individual's repertoire (that is, functions he/she is able to perform without the tools). And furthermore, these tools require an individual to perform the same general action with the tool as that without. Category I tools include tools like hammers, rakes and tongs. A hammer enables a human to hit a wooden board, something he can do without tools, and with his fist. Furthermore, using a hammer requires an individual to make the same hitting action with his arm, as that when using his fist. Similarly, a rake is used for reaching, something a human can already do with his arm, and using a rake requires him/her to perform a reaching action with his/her arm as performed without the rake (though he may adjust is reach to rake's shape). Another case in point: a tong helps a human lift up hot or distant objects, an while he has to hold the tongs in his hand and adjust for its length, using a tong also requires a pinching action, something a human would have used to pick up the objects with his/her bare hands.

This is not so for Category II tools, which include tools like a traditional bow drill, a car jack or a power hammer. These tools augment functions in a human's repertoire (poking, lifting or hammering respectively), but using these tools requires an individual to make actions different from what he/she would have performed to achieve the same functions without the tool. For example, poking/drilling using a bow drill does not require one to produce a poking action with his/her arm, but to-and-fro movements instead; lifting a car with a car jack does not require an individual to produce a lifting action, but rather a pumping/rotary action with his hand.

Finally, Category III tools provide new functions that a human cannot perform without a tool. These include most modern tools ranging from a vacuum cleaner and chain saw, to computers.

Interestingly, note that all the tools that humans are able to use immediately are Category-I tools. We can intuitively recognize and use a stick as a rake, or tongs to pick up objects. Our previous examples of using a cricket bat, paper sheet and the tree branch are also examples of Category I tools, which (augment but) afford actions that we can do even without the tools, and with actions similar to those when using the tools (see Fig. 1a). This intuitiveness is however, visible to a lesser extent with Categories II and III tools. A person who has never seen a bow drill will not be able to immediately use it. He would have explore and learn how to use it. Similarly, a person who has never seen a power hammer (a Category II

[1]Please see the supplementary video accompanying this manuscript. Also available online https://youtu.be/gGCAWJ40K6I

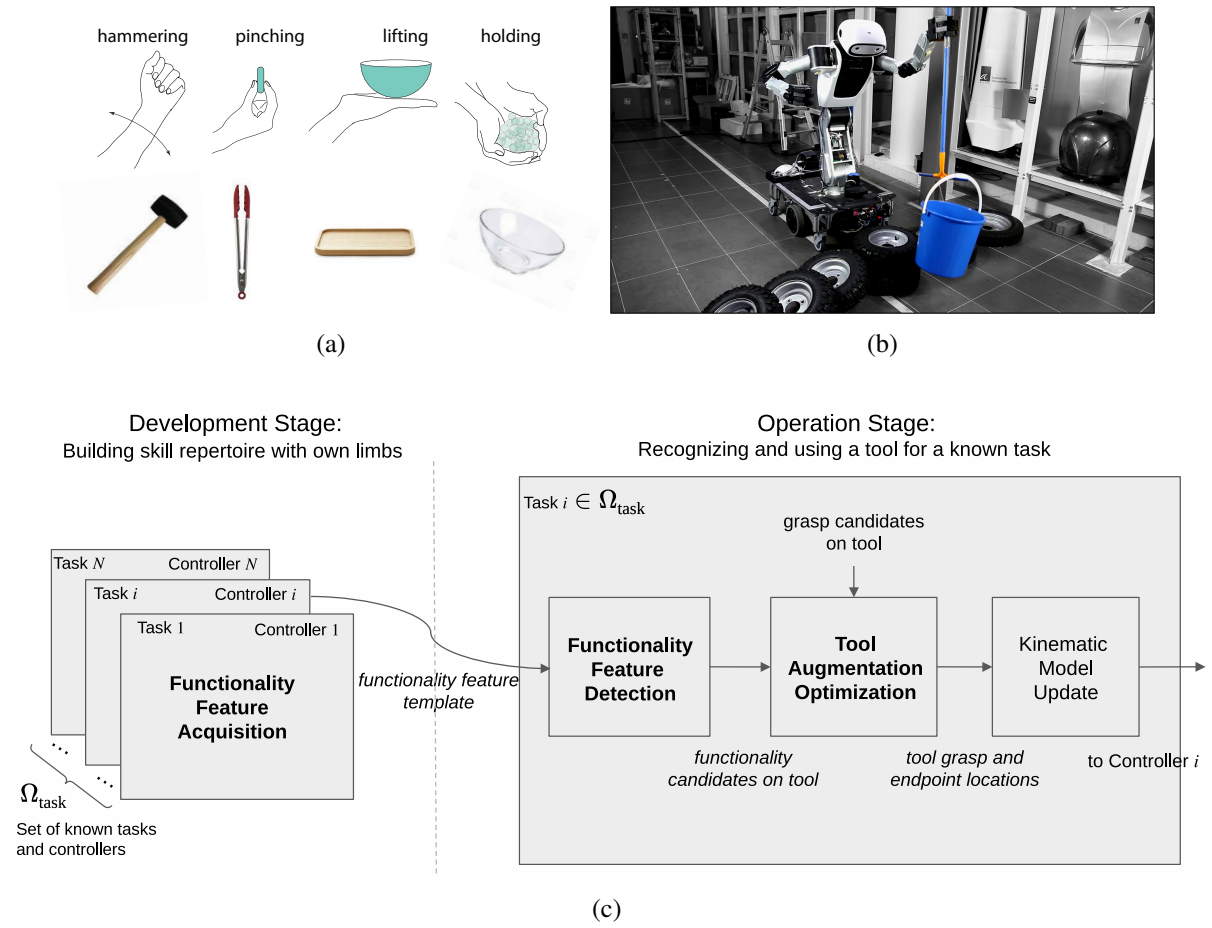


Figure 1: (a) Robot tool cognition concept: Note the physical similarity between the hand while performing a hammering, pinching, lifting and holding task, and the corresponding objects that we recognize as tools for the same tasks. We propose that robots can compare 'functionality' features, crucial for a task, on their limbs to recognize unknown objects as potential tools for the task. (b) Example of in situ tool use in which our robot retrieved a distant pail with an object it recognized as a tool, and picked up from its environment[1]. The robot had no prior learning with the tool or observation of the tool being used. (c) Overview of the proposed tool cognition framework enabled by key components comprising functionality feature acquisition, functionality feature detection, and tool augmentation optimization. Grasp candidates on the tool are assumed to be available in this paper (see e.g. [28, 29] for grasp generation techniques).

tool) or a vacuum cleaner (a Category III tool) will not be able to use it unless he finds out that he needs to press a button to start it and learns (either by himself or by reading instructions) what happens when the button is pressed.

The above observations of Category 1 tools enable known tasks with actions one already possesses, suggested to us that like humans, robots should be able to utilize available non-tool skills (controllers in case of robots) to use Category 1 tools without any learning. But how can robots recognize category 1 tools? The intuition for the answer to this question came from our Neuroscience studies on the issue of *embodiment*.

In psychology, it is believed that human tool skill stems from their ability to *embody* tools [31]-[34]. Our recent investigations of limb and body embodiment by humans have highlighted that the ability of our brain to embody an entity is critically dependent on whether the functionality offered by it [35, 36, 37] matches that to the limb it replaces. [35]. This result suggests that human's may recognize embodied objects (tools in our case) by comparing the functionally relevant physical features of the object with that of their own limb. Correspondingly, here we propose to use the limbs of a robot (or certain features of it) as a template to recognize (Category 1) tools for a task (see Fig. 1a).

Utilizing these intuitions, here our framework proposes the following to enable tool recognition and use by robots without any learning.

We propose (see Fig. 1C) that:

a) A robot can identify an object as a potential tool for a task by detecting *functionality* features similar to those on the robot's limbs crucial for performing the same task.

b) Once recognized, the robot can search for a suitable pair of grasp and functionality feature locations using a standard planner in order to optimize the required *augmentation* to performing the task.

c) It can then update its own kinematic and/or dynamic model (we deal with only kinematic tools in this work) to account for the gripped tool and use the tool with the same task controller as without tools.

For illustrative purposes, consider a robot with anthropomorphic hands that can catch a tennis ball in its hand. The robot posseses a *catching controller* to first reach the falling ball, and then ensure that it enters into its cupped hands. Then, when catching a volley ball (which it cannot do with its hands), our framework enables the robot to acquire and use the shape of its cupped hands (the functionality feature) to recognize that a bucket is a tool for catching, even though it may have never seen or used a bucket before. The robot can then use the same catching controller, with an updated robot kinematic model (to account for the bucket) to catch the volley ball with the bucket, without requiring any learning.

Taking an example from our experiments, in which our robot uses a *reach controller* to pull objects using its curved (hooked) hand like a rake. When the robot needs to rake in an object beyond its reach, it searches for similar curved parts (the functionality for this task) on objects in its environment, and so can recognize an umbrella as a rake tool, even though it has never seen it before. The grasp location on the umbrella is decided according to how far beyond its reach the object is, while the use of the umbrella is enabled by the use of the same reach controller that it already possesses, albeit with updated kinematics to account for the umbrella .

The specific procedures for the acquisition and detection of functionalities is described in Sec. 4.1 and Sec. 4.2, while results of these procedures are provided below in Sec. 2.1 and Sec.2.2. The procedure

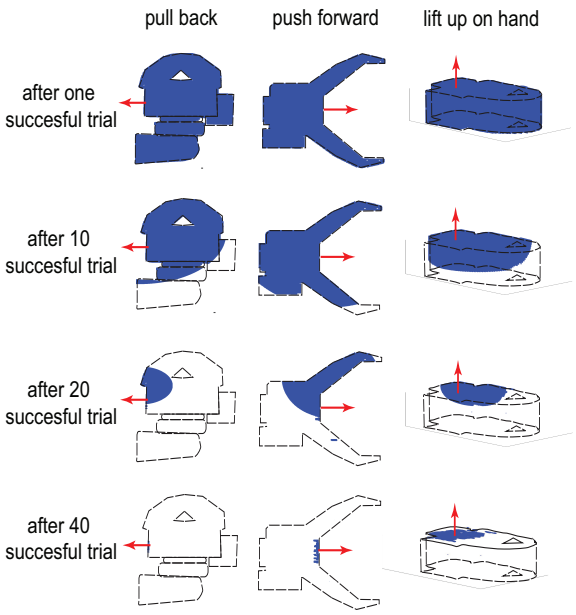


Figure 2: The functionality feature learnt by the robot by pulling objects towards itself (left column), pushing objects away from its body (center column) and lifting objects on its hand (right column), each across 1-40 successful trials (rows). The blue volume shows the contact feature representation starting from a naive prior and updating through successful trials. The red arrow shows the task feature representation, which is the average direction of the object velocity during the task. The functionality features have been superimposed on the end-effector image for reference.

utilized to select the grasp, movement plan considering required augmentation, and the final tool execution are explained in Sec. 4.3 and Sec. 4.4 respectively, while these results are shown below in Sec.2.3. Due to space constraints, a detailed comparison of these procedures to the those proposed in literature, is provided in the supplementary materials, while we provide a list of limitations of the framework in the discussion section.

2 Results

2.1 Functionality Features acquisition from the Robot's Limb

Our framework accumulates contact points over successful trials of the task to infer the features of the limb crucial for the task, that is, the functionality features. Fig. 2 shows the three dimensional functionality features learnt for three tasks. Each task involved 40 successful exploration trials where the robot manipulated cubic or cylindrical objects of different sizes (see Sec. 4.1 for experiment details). The blue volumes in Fig. 2 show the contact features, represented as *3D voxels* superimposed on the 3D outline of the shape of the end-effector. At the beginning, the entire end-effector is considered to represent the functionality feature, but during each successful trial, the robot updates the functionality feature by accumulating and analyzing the spatial frequency of the contact points. Eventually, after 40 successful trials, the crucial contact features required for each task remain on the surface of the limb.Our

functionality feature representation consists of the *task feature* (remaining blue surface) and *contact features* (red arrows), which encode the average direction of the object movement across the 40 trials.

2.2 Functionality Detection on Objects

After the functionality features (see Sec. 4.1) required for completing a task have been acquired, the robot is able to use them as a visual template to recognize objects that can potentially serve as a tool for the same task.

Results on segmented tool point clouds, computed voxel surfels, detected candidate functionality features, and grasp candidates for pushing/raking from Sec. 4.1 are shown in Fig. 3 for a 3D-printed L-shaped tool as well as real-world objects. The figure also shows 'lift' functionalities detected on 2 dustpans and a frying pan. As seen previously in the functionality learning results (Fig. 2), the lift functionality is a surface patch with a direction vector. More examples of functionality detection on real objects are shown in Fig. S2 and S3.

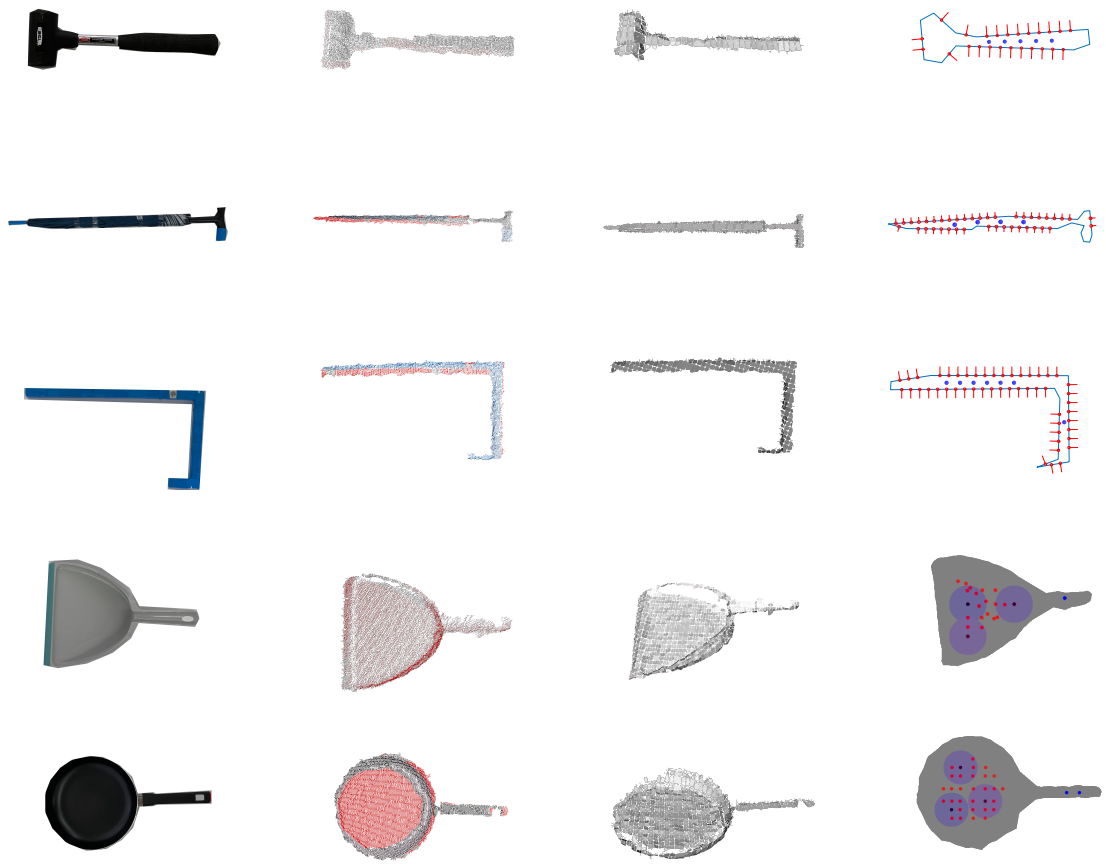
2.3 Planning Grasp and Movement for Augmentation

From the many candidate functionality features that were detected on a potential tool, we use a Monte Carlo Tree Search (MCTS) to find a solution, in terms of a pair of grasp and functionality candidates, that provides the required augmentation to satisfy the task goal and constraints while optimizing for arm manipulability and configuration changes. Details of the algorithm are explained in Sec. 4.3. To demonstrate the tool use enabled by our framework, we first present results for a simulated experiment in a 2D world with a 3 degree-of-freedom (DoF) planar robot, followed by real experiment results on a 10-DoF articulated robot as further validation.

2.3.1 Simulated Experiment

The simulated experiment (Fig. 4) involved 3 object pushing tasks, namely direct point-to-point push (Task 1), push around an obstacle (Task 2 with the red circular obstacle), and push into a channel (Task 3). For each task, 3 different tools were available, which the robot observed for the first time, and had to choose one to perform the task. The choice was made based on the *utility score*, which is a measure of how well the utility function consisting of a measure of the task error, configuration changes, considering the task constraints, are met. Details of the utility score and simulation setup can be found in Sec. 4.3 and 4.5 respectively.

The heat maps in the figure is the visualization of the utility scores of all the solution candidates. It is observed that the heat maps for Task 1 have a greater density of high scoring regions compared to Tasks 2 and 3, whose heat maps are much sparser. This is due to the fact that Task 1 is a relatively easier with many feasible combinations of grasp and push locations, and our algorithm chooses the highest-scoring combination as the optimal solution. Task 2 and 3, on the other hand, is the most constrained as it involves 2 movements that also need to avoid collision with the obstacle, yielding smaller pockets of sufficient grasp and push locations pairs. For Task 3, the heat map for Tool 1 has two bands while that for Tools 2 and 3 have only a single band (at the top of the heat map). This is because both tips of Tool 1 can be used to push the object into the channel, while only a particular tip is feasible for Tools 2 and 3. Though specific in the push location choices, Task 3 offers high flexibility in the choice of where to



(a) (b) (c) (d)

Figure 3: Perception of functionality features on objects for a pushing/raking task (top 3 rows) and a lifting task (bottom 2 rows). (a) Observed object. (b) Segmented point clouds. (c) Voxel surfels provide local planar approximation of the object surface. (d) Detected functionality features are shown as red spikes for the pushing/raking task, and red dots/blue patches for lifting. To avoid clutter in the visualization, functionality candidates are shown simply as red dots representing the center of the circular patch. Circular patches are shown for representative samples only. Grasp candidates (manually specified) are shown as blue dots.

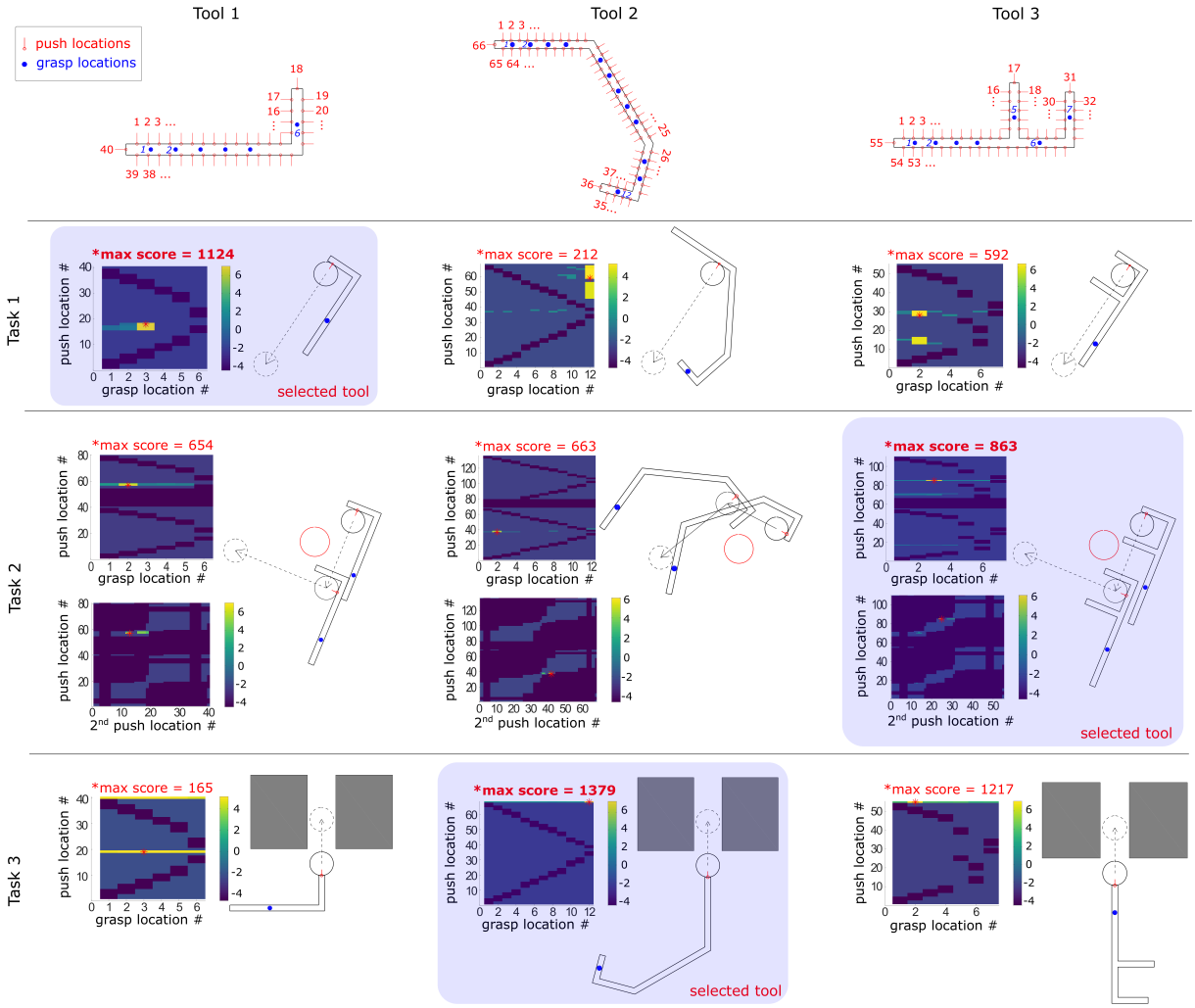


Figure 4: Selection of the best tool for a given task based on optimal utility scores obtained after 3200 MCTS simulations. The heat maps show the utility scores for each task. The robot compared the maximum scores for the 3 different tools and chose the tool with the highest maximum score. In the top row, index labels for the grasp and push functionality candidates are indicated for each tool in blue and red respectively. For tasks 1-3, black circles represent the target (dotted) and object (solid line) respectively, and large red circles for task 2 represent obstacles. Dotted arrows indicate the desired movement direction. In the heat maps, warmer colors indicate higher log scores, with red asterisk '*' denoting the solution with maximum score. Task 2 involves 2 heat maps because a second round of push location selection is required before the second leg of movement to circumvent the obstacle.

grasp the 3 tools, as most of the grasp locations have relatively high scores when matched with the few feasible push locations (see also Supplementary figure S1).

These results show that the proposed framework enables the robot to adapt the use of different tools to achieve the same task, as well as, use a same tool for different tasks. Evolution of the heat maps with the number of simulations was also studied. We found that (see Supplementary Fig. S6) that with sufficient number of MCTS simulations, the optimal solution tends to converge. While the results are for a particular combination of task and tool, similar results can be obtained for other combinations.

2.3.2 Real Experiment

Further validation of our framework was performed in experiments with a physical robot to perceive, recognize and use objects in the real world as tools. Other than 3D-printed tools, we also tested on real objects like an umbrella[2]. Details about the robot and the experiment setup can be found in Sec. 4.6

The top row of Fig. 5 shows the heat maps and optimized tool augmentation solutions corresponding to the tasks shown in Figure 5. It can be seen from the heat maps that the solution regions are sparse and concentrated, which reflects the difficulty of finding real world inverse kinematics solutions given the joint limits, restricted workspace of the arm, as well as the constrained end effector orientation necessitated by the task. The 2D stick diagrams beside the heat maps show the perceived 2D tool outlines, the identified optimal grasp and push points on the tools, and the corresponding initial tool pose to push the objects (yellow circles) to their respective targets (red circles). This initial tool pose, along with the desired final tool pose when the object is at the target, provide the key inputs for robot motion planning, before the tool action is finally executed.

Trials with obstacles were also performed, but not shown here due to space constraints. We provide snippets of these trials in the two supplementary videos accompanying this manuscript.

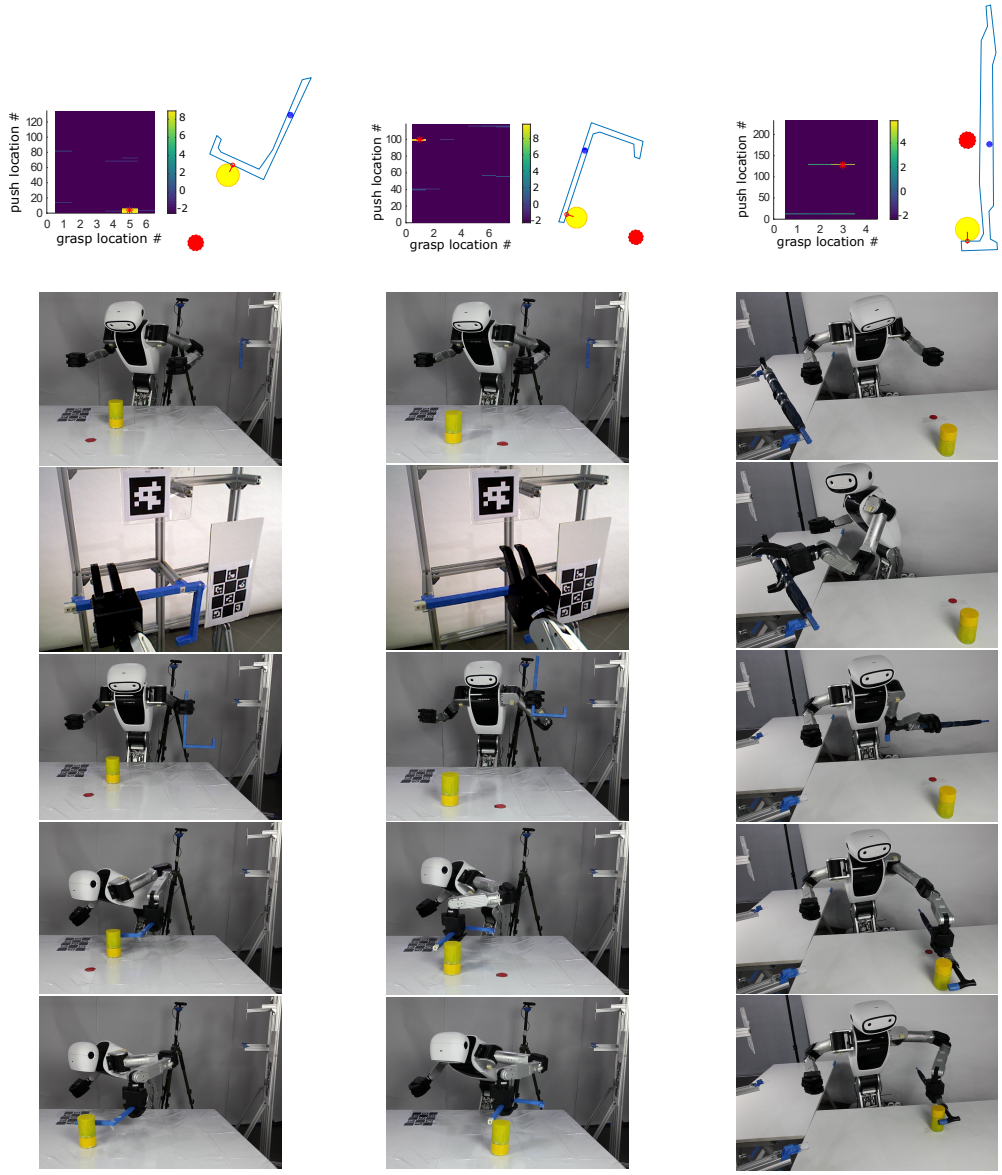
3 Discussion

Controllers in robots provide it with skills to achieve specific tasks. Utilizing intuitions from Neuroscience studies of human tool use and embodiment, here we propose a framework that can allow a robot to immediately transfer this skill to tool-use in the same task. The framework enables robots to recognize and utilize tools in the task without requiring any supervised tool learning, which has been the norm with all robot tool affordance and tool-use frameworks in literature.

We believe the tool use capability provided by our framework can provide robots with the ability to be innovative and autonomous in unstructured environments where obstacles can impede the completion of a task (see Supplementary Table S1 for comparison with existing work). However, to understand the full scope of the framework, its also important to understand its limitations, which we discuss below.

First, this framework was designed to enable robots to recognize and use specifically Category I tools. This will allow it to use available objects as tools for tasks it knows. To use Categories II and III tools, which mostly include man-made devices, the robot needs to experience and learn them, using techniques such as those previously suggested [6, 12, 13]. Learning can of course also aid the use of Category 1 tools and hence techniques of tool learning and tool skill transfer may be used in addition to our framework to further innovate robot tool use.

[2]Please see videos accompanying this manuscript. Also available online at https://youtu.be/yCgocGncPrg



(a) (b) (c)

Figure 5: Snapshots of real robot performing (a) forward push using an L-shaped tool, (b) sideways push using the same L-shaped tool, and (c) backwards pull using a real umbrella. The robot first perceived the scene and, on determining that the task was infeasible with its end effector, turned to the tool rack. After selecting the optimal pair of grasp and functionality candidates on the previously unseen tool, it used the tool to complete the push task with both torso and arm motion. The chosen solution is shown as a '*' on the heat map, while the kinematics corresponding to the chosen solution is shown adjacent to the heat map.

Second, our framework proposes to recognize the functionality and augmentation in objects using vision. This limits the current implementation to the recognition of tools that extend the kinematic abilities of robots. To use dynamic augmentation tools like a hammer, a robot has to recognize its dynamic properties, like its mass, which is not directly possible through vision. Future work on developing visuo-dynamic association, to enable a robot to associate textures and shape to mass for example, can enable extension of our framework to identify and utilize tools that provide dynamic augmentations.

In any case, our proposed framework is heavily reliant on 3D visual perception for the recognition of a tool, and the visual abilities of the robot are the third and probably the single most critical constraint for the use of the framework. In the current manuscript, we proposed to use a voxel surfel representation and point-pair feature matching as a means of recognizing functionality features on tools (Sec.4.2), but optimizing this technique for tool recognition is not the focus of this manuscript. Robot tool recognition can potentially be improved in the future by using more sophisticated 3D template matching techniques such as [38, 39, 40].

Similarly, our functionality learning algorithm can also be improved. In the current study we provide an 'object-independent' functionality learning algorithm that is specific for manipulation tasks. This method enables a robot to identify tools that are valid for a particular manipulation of all (or maximum) objects. On the other hand, this method can miss tools that are valid only for specific objects. Functionality learning can be optimized separating task specific (object independent) and object specific functionality learning. This can help expand the repertoire of tools a robot can recognize and use.

Although our current functionality learning, being studied in simulation, has performed robustly in limited real experiments, it may encounter sim-to-real issues in other varied and complex scenarios. To improve robustness in real applications, techniques that randomize dynamics in simulation (e.g. [41]) is a promising avenue for future investigation.

Finally, While many animals (including apes, crows, jays, elephants among others) have been documented to be able to use tools [42], only apes and New Caledonian crows are known to able to make tools [43, 44, 45]. Inspired by the tool shaping ability of New Caledonian crows, we conjecture that our framework can be extended to at least rudimentary tool-making. We provide the video and explanation of the first preliminary tool making experiment in the supplementary materials.

4 Materials and Methods

Our tool cognition framework (Fig. 1c) consists of 3 key components, namely i) functionality feature learning (sans tools), ii) functionality feature detection on previously unseen tool, and iii) tool augmentation optimization, which work together to enable a robot to recognize and use a tool without any previous experience on the tool itself. During a 'development' stage, the robot builds its skill repertoire by acquiring functionality features of different tasks with its limbs. Subsequently, when faced with a known task that needs a tool, the robot detects task-relevant functionality features on a tool, and then optimizes the tool grasp and endpoint locations to achieve the required augmentation. Finally the robot motion is planned and performed with an incremental update of the end effector kinematics with the tool in hand. Grasp planning/generation is not studied in this paper, but interested readers can refer to [28, 29] for related techniques.

4.1 Acquiring Functionality Features

Functionality features relevant to an object manipulation task can be acquired by performing the task with the robot's limb only (without any tools). In this paper, a simulated model of the Olivia III robot (Figs. 6a,6b) was used to perform object manipulation trials, using its right arm end effector, in ROS Gazebo environment. Contact sensors covering the end effector were activated (using Gazebo contact sensor plugin) to register contact points with the object. Data of the contact positions on the robot end effector was collected at 100Hz from the start to end of each manipulation motion. Then, the batch of data for each task was used to acquire the functionality features F for the task, which will be used as a template/model for finding similar features on a potential tool.

We considered single-hand manipulation tasks which included pushing and lifting of objects. For the pushing task, the environment consisted of a tabletop in front of the robot, and an object on the tabletop. The robot was required to manipulate the object over a distance of 10cm in 3 different directions: north, west, and south. For northward pushing (Fig. 6a), the robot spread its fingers and pushed the object with the inside of its palm. For westward (Fig. 6b) and southward sliding, the robot used the side of its palm. This push style was *a priori* knowledge given to the robot before acquiring the functionality features. For each direction, the robot performed 60 trials, with random initial object position (uniformly distributed about nominal position with 2cm maximum deviation in x and y directions). For the lifting task, an object is supported on a flat upward-facing surface of the robot's hand, and moved upwards by 10cm. That the flat upward-facing surface was chosen was *a priori* knowledge given to the robot. Both cylindrical and cubic objects of random sizes varying from 2cm to 8cm were used.

The main idea for acquiring functionality features is to use accumulated limb-object contacts during successful task performance to update the functionality feature model iteratively, starting with the entire limb as an initial model. Details of the algorithm are described in the following.

Let $\mathbf{B} = \{b(i_x, i_y, i_z)\}$, $\mathbf{O} = \{o(i_x, i_y, i_z)\}$, and $\mathbf{C} = \{c(i_x, i_y, i_z)\}$ be voxel representations of the robot's limb, manipulated object, and limb contact points, respectively, where (i_x, i_y, i_z) are the voxel coordinates with respect to a reference frame $\mathcal{O}_B$ on the limb. For simplicity, we define the manipulation task by the error vector $\mathbf{T} = p_{tar} - p_{obs}$ between the initial object position p_{obs} and the target position p_{tar}. Let the robot try to learn the task $\mathbf{T}$ over N movement trials.

Define the functionality feature as $F = \{F_t, F_c\}$, consisting of *task feature* F_t and *contact features* F_c. The task feature captures information about the manipulation task requirement, and is specified by $F_t = \mathbf{T}/\|\mathbf{T}\|$, the normalized direction vector between the object and the target, originating from $\bar{F}_c$, the centroid of the contact points in the learnt contact feature F_c. The contact feature F_c are the points of contact between limb and object during a successful execution of the task manipulation.

The total set of accumulated contact points is given by:

$$\mathbf{C}_N = \{c_N(i_x, i_y, i_z)\} = \bigcup_{k=1}^{N} \bigcup_{j=1}^{n_k} \mathbf{B} \cap \mathbf{O}_j^k \tag{1}$$

where n_k is the number of time steps in the kth trial, and $\mathbf{O}_j^k$ the object voxel representation, with respect to reference frame $\mathcal{O}_B$ on the limb, at the jth time step during the kth trial.

Let $\mathbf{S}_\mu^N(r)$ be the voxel representation of a sphere of radius r about a voxel $\mu = (\mu_x, \mu_y, \mu_z)$ if the

Nth trial is successful, i.e.

$$\begin{aligned} \mathbf{S}_{\mu}^{N}(r) &= \{s_{\mu}^{N}(i_x, i_y, i_z, r)\} \\ s_{\mu}^{N}(i_x, i_y, i_z, r) &= \begin{cases} 1 & \text{if } \sqrt{(i_x-\mu_x)^2+(i_y-\mu_y)^2+(i_z-\mu_z)^2} \leq r \text{ and } N\text{th trial successful,} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Through exploration of the task with its limb, the set of contact features after N trials is obtained as the intersection of limb voxels $\mathbf{B}$ and spheres centered on accumulated contact voxels $\mathbf{S}_{\mu}$, as follows:

$$F_c^N = \bigcup_{\mu \in \Omega_C^N} \mathbf{S}_{\mu}^{N}(r_N) \bigcap \mathbf{B} \tag{3}$$

where $\Omega_C^N = \{i_x, i_y, i_z \mid c_N(i_x, i_y, i_z) = 1\}$ is the set of voxels that registered at least one contact over the N trials, and r_N is a contracting sphere radius given by

$$r_N = \begin{cases} \max(0,\ \theta - \gamma\, n_C) & \text{if } r_N > r_{min} \\ r_{min} & \text{otherwise} \end{cases} \tag{4}$$

which decreases with the total number of accumulated contact points $n_C = \sum_{i_x}\sum_{i_y}\sum_{i_z} \mathbf{C}_N(i_x, i_y, i_z)$, and θ, γ are positive constant parameters that determine the learning rate.

After the contact feature is learnt, a reference frame for the functionality feature F is placed at $\bar{F}_c$, the centroid of the contact points, so that it can be used as a template to search for functionality candidates on a potential tool. To allow generalization of this learnt feature representation to different object sizes, we multiply a scaling constant k_s for manipulating any new object:

$$F_{c,new} = k_s F_{c,learnt} \tag{5}$$

where

$$k_s = \frac{A_{\perp}(\mathbf{O}_{new}, F_t)}{\bar{A}_{\perp}(\mathbf{O}_{new}, F_t)} \tag{6}$$

with $A_{\perp}(\mathbf{O}_{new}, F_t)$ the area of the projection of object $\mathbf{O}_{new}$ on a plane perpendicular to F_t, and $\bar{A}_{\perp}(\mathbf{O}_{new}, F_t)$ the average projection from the trained objects. For example, the functionality features learnt from interacting with a tennis ball with the limb can be scaled up to recognize a potential tool for interacting with a basketball, by using the ratio of projected object area estimated by visual perception.

4.2 Detecting Functionality Features

To find a potential tool which can perform the specified task, tool surfaces need to be examined for the required functionality features. In this paper, we use voxel surfels, originally proposed in [46], to rapidly perform visualization and meshification of point clouds with the advantages of reduced data size and faster processing speed compared to point clouds. We simplify the voxel surfel representation by dividing the 3D space into a set of non-overlapping voxels $V = \{v_i, i = 0, ...1\}$ with dynamic voxel size $[v_x, v_y, v_z]$. Then, a voxel surfel is described by the set:

$$\nu_i = \{\omega_i, \eta_i, \xi_i\} \tag{7}$$

where ω_i is the centroid of all points in the voxel, η_i the normal of the dominant plane, and ξ_i a measure of flatness for the dominant plane. Essentially, the voxel surfels are locally linear approximations of the object surface.

Based on the voxel surfel representation, we detect functionality features on the tool by using point-pair features (PPF) [47], which has been found to be efficient, accurate and robust for 3D object matching [48]. We construct a PPF vector from a pair of voxel surfels ν_i, ν_j as follows [47]:

$$\mathcal{P}_{ij} = [\|d_{ij}\|,\ \angle(\eta_i, d_{ij}),\ \angle(\eta_j, d_{ij}),\ \angle(\eta_i, \eta_j)] \tag{8}$$

where $d_{ij} = \omega_j - \omega_i$. We form global descriptors, for the functionality feature template F, as the full set of PPFs $\{\mathcal{P}_{ij}^f\}$, and a hash table $\mathcal{H}^f$ that quantizes PPFs such that similar PPFs share the same key (see Algorithm 1). The same is done for the tool candidate to obtain $\{\mathcal{P}_{ij}^t\}$ and $\mathcal{H}^t$. Then, the template PPFs are compared with PPFs for each voxel surfel on the tool. For every matched hash table key, a 'local transformation' α that maps (ω_i^f, ω_j^f) to (ω_i^t, ω_j^t) is obtained by solving [47]:

$$\begin{aligned} \omega_j^t &= T_{t\to g} R_x(\alpha) T_{f\to g} \omega_j^f \\ T_{t\to g}\eta_i^t &= T_{f\to g}\eta_i^f = [1,0,0]^T \end{aligned} \tag{9}$$

where $R_x(\bullet)$ is the rotation matrix about the x-axis of a reference frame g. A vote is cast for the 'local coordinate' (ν_m^t, α) in a discrete accumulator space. The most-voted local transformation α_* within the set of matched template PPFs Ω_m is then used to form a hypothesis, which is checked by applying α_* to other voxel surfels in Ω_m. If the hypothesis is not rejected, then a functionality feature is detected at tool voxel surfel ν_*^t. The above procedure for detecting functionality features on a tool candidate is summarized in Algorithm 2.

Algorithm 1 Generating global descriptors for functionality template

Input: Functionality template F

Output: Global descriptors: point pair features $\mathcal{P}^f$ and hash table $\mathcal{H}^f$

```
Compute set of voxel surfels for F, {ν_i^f}_{i=1,...,N_f}, from (7)
Create H^f with discrete cells covering a range of distance and angle values between point pairs
for i = 1 to N_f do
    for j = 1 to N_f do
        if i == j then continue
        Compute P_ij^f for ν_i^f and ν_j^f based on (8)
        Place P_ij^f in the corresponding cell in H^f and register its key
return P^f and H^f
```

4.3 Optimizing Tool Augmentation

We formulate the tool augmentation optimization problem as follows. Define the *functionality pose* as the position and orientation of the patch on the tool surface that contacts the object in order to perform the task. Let the forward kinematics of the arm be $X = f(q)$, where q is the vector of arm joint angles,

Algorithm 2 Detecting functionality features on a tool candidate

Input: Tool point cloud $\mathcal{C}^t$. Functionality template global descriptors $(\mathcal{P}^f, \mathcal{H}^f)$ and voxel surfels $\{\nu_i^f\}_{i=1,\dots,N_f}$

Output: $\mathcal{T} = \{\nu_*^t, \alpha_*\}$: list of tool voxel surfels ν_*^t with the required functionality features, along with the corresponding transformations α_*^t

set $\mathcal{T} = \emptyset$
Compute tool voxel surfels $\{\nu_i^t\}_{i=1,\dots,N_t}$ from $\mathcal{C}^t$ and (7)
Create $\mathcal{H}^t$ with discrete cells covering a range of distance and angle values between point pairs
for $i = 1$ to N_t **do**
    for $j = 1$ to N_t **do**
        if i==j **then** continue
        Compute point pair feature $\mathcal{P}_{ij}^t$ from ν_i^t, ν_j^t and (8)
        Place $\mathcal{P}_{ij}^t$ in the corresponding cell in $\mathcal{H}^t$ and register its key
        $\Omega_m \leftarrow$ set of PPFs in $\mathcal{H}^f$ that matches the key of $\mathcal{P}_{ij}^t$
        $N_m \leftarrow$ size(Ω_m)
        for $k = 1$ to N_m **do**
            Compute local transformation α_k from $\mathcal{P}_k^f$ to $\mathcal{P}_{ij}^t$ by solving (9)
            Vote for the local coordinate (ν_k^f, α_k)
    Obtain hypothesis transformation α_* from the most-voted local coordinate (ν_*^f, α_*) in Ω_m
    Map ν_*^f to ν_*^t using α_* and (9)
    Similarly, map other matching template surfels $\{\nu_k^f\}_{\Omega_m}$ to corresponding tool surfels $\{\nu_k^t\}_{\alpha_*}$
    $N^* \leftarrow$ number of correspondences between $\{\nu_k^t\}_{\alpha_*}$ and $\{\nu_i^t\}$
    if $\frac{N^*}{N_m} >$ threshold **then**
        Append ν_*^t, α_* to $\mathcal{T}$
return $\mathcal{T}$

and X the end effector pose. For optimization of tool augmentation, we find the solution $\theta = (\theta_g, \theta_f)$ consisting of the grasp pose, θ_g, and functionality pose on the tool, θ_f, both in the local tool frame, that maximizes the utility function $\mathcal{L}$ as follows:

$$\mathcal{L} = \det\left(J(q)J^T(q)\right) \tag{10}$$

$$\theta = \underset{\theta_g \in \Omega_g, \theta_f \in \Omega_f}{\operatorname{argmax}} \mathcal{L} \tag{11}$$

where $J = \frac{\partial f(q)}{\partial q}$ denotes the Jacobian, $\Omega_g = \{\theta_{g_1}, ..., \theta_{g_m}\}$ and $\Omega_f = \{\theta_{f_1}, ..., \theta_{f_n}\}$ the sets of grasp and functionality candidates respectively. The utility maximization is subject to the constraints:

C1. The functionality candidate θ_f on the tool aligns with each desired task functionality:

$$T^i_{w\to t}\theta_f = X^i_{task}, \quad i = 1, 2, ..., n_f \tag{12}$$

where $T^i_{w\to t}$ is the transformation from the world frame to the ith tool frame, X^i_{task} the ith desired functionality pose for the task, expressed in the world frame, and n_f the number of desired functionality poses in the task. An example is an object pushing task with $n_f = 2$ where X^1_{task} and X^2_{task} are the desired initial and final functionality poses for the tool. More complicated tasks like scooping can also be specified as a sequence of functionality poses.

C2. The tool grasp poses are reachable:

$$\exists\, q^i = f_{inv}(T^i_{w\to t}\theta_g), \quad \forall i = 1, 2, ..., n_f \tag{13}$$

where f_{inv} is the inverse kinematics of the arm, and q^i the joint angle solution for the grasp pose corresponding to the ith desired functionality pose.

C3. The tool does not collide with the environment.

C4. The robot joint limits are satisfied.

To solve the tool augmentation optimization problem, we use Monte Carlo Tree Search (MCTS), an anytime, heuristic search algorithm which uses stochastic simulations to find the most promising set or sequence of decisions in a search tree [49].

The action space $\mathcal{A}$, in the tool frame, can be represented by a Cartesian product of 4 layers of decision candidates, for the case without obstacle handling:

$$\mathcal{A} = G \times \Phi \times S \times P \tag{14}$$

where $G = \{g_1, ..., g_{n_g}\}$ is the set of candidates for grasp locations, $\Phi = \{\phi_1, ..., \phi_{n_\phi}\}$ grasp orientations, $S = \{s_1, ..., s_{n_s}\}$ segments on the tool surface, and $P = \{p_1, ..., p_{n_p}\}$ functionality locations on the tool surface segment.

The structure of the search tree is shown in Fig. S4, where the decision layers are organized in a manner intuitive for understanding, e.g. for each grasp location, there is a set of grasp orientations, and for every grasp orientation within this set, there is a set of valid tool segments, and, in turn, for every tool segment within this set, there is a set of valid functionality locations.

The selection step of the MCTS uses the Upper Confidence Bound for Trees (UCT) [50] to select the best child node while maintaining a balance of exploitation and exploration. For a node where one or more child nodes have not been visited, a random sequence of actions is made until a functionality candidate on the surface of the tool is selected (i.e. terminal node). Then, evaluation is performed to determine the **utility score** (or reward) $r_{terminal}$, based on the manipulability cost (10) and constraints C1-C4. This score is then backpropagated up the tree to the root node:

$$r_{terminal}(k) = r_{terminal}(k-1) + \Delta r(k) \tag{15}$$

$$\Delta r(k) = \begin{cases} c_1\mathcal{L} + c_2, & \text{if constraints C1-C4 are satisfied} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$r_{parent}(k) = r_{parent}(k-1) + r_{child}(k) \tag{17}$$

for the kth simulation, and where c_1, c_2 are positive coefficients.

After N_{sim} simulations, the terminal node with the highest utility score, along with the trace of parents up the search tree, constitute the solution to the augmentation optimization problem, $\{g^*, \phi^*, s^*, p^*\}$, in the tool frame. This is rewritten in world frame pose forms for grasp X_g and functionality X_f as follows:

$$\theta_g = \begin{bmatrix} R(\phi^*) & g^* \\ 0 & 1 \end{bmatrix} \quad , \quad \theta_f = \begin{bmatrix} R(n_{s^*}) & p^* \\ 0 & 1 \end{bmatrix} \tag{18}$$

$$X_g^i = T_{w\to t}^i \theta_g \quad , \quad X_f^i = T_{w\to t}^i \theta_f \tag{19}$$

for $i = 1, 2, ..., n_f$, where $R(\bullet)$ is the rotation matrix based on orientation $\bullet$, and n_{s^*} the unit normal of segment s^*.

Algorithms 3-4 illustrates the MCTS algorithm for finding the best combination of grasp and functionality candidates. Backpropagation$(root, r_{terminal})$ backpropagates utility score (17) to the root node. BestAction(r_{best}) selects the terminal node with the highest utility score r_{best} over N_{sim} simulations, and traces its parents up the search tree to yield the best action $a^* = \{g^*, \phi^*, s^*, p^*\}$.

Algorithm 3 MCTS for Augmentation Optimization

Input: Action space $\mathcal{A}$, maximum number of iterations N_{sim}, desired task functionality poses $\{X_{task}^i\}_{i=1:n_f}$.

Output: Best action $a^* = \{g^*, \phi^*, s^*, p^*\} \in \mathcal{A}$.

$r_{best} \leftarrow 0$
$r_{terminal} \leftarrow 0$
for $i = 1$ to N_{sim} **do**
  $leaf \leftarrow$ Selection&Expansion$(root)$
  $a \leftarrow$ Simulation$(leaf)$
  $r_{terminal} \leftarrow r_{terminal} +$ RewardEvaluation $\left(a, \{X_{task}^i\}_{i=1:n_f}\right)$
  $r_{best} \leftarrow \max(r_{best}, r_{terminal})$
  Backpropagation$(root, r_{terminal})$
$a^* \leftarrow$ BestAction(r_{best})
return a^*

Algorithm 4 RewardEvaluation

Input: Action $a = \{g, \phi, s, p\}$, desired task functionality poses $\{X^i_{task}\}_{i=1:n_f}$.
Output: Reward Δr

$\theta_f \leftarrow$ ComputeLocFuncPose(s, p)
$\theta_g \leftarrow$ ComputeLocGraspPose(g, ϕ)
$P_r \leftarrow$ true
$\mathcal{L}_{min} \leftarrow$ large_number
for $i = 1$ to n_f **do**
    $T^i_{w\to t} \leftarrow X^i_{task}\theta_f^{-1}$
    $q^i \leftarrow$ InvKin$(T^i_{w\to t}\theta_g)$
    $\mathcal{L}_{min} \leftarrow \min\left(\mathcal{L}_{min}, \det\left(J(q^i)J^T(q^i)\right)\right)$
    $P_r \leftarrow P_r \wedge$ CheckCollision$(T^i_{w\to t}) \wedge$ CheckJointLim(q^i)
if P_r is true **then**
    $\Delta r \leftarrow c_1\mathcal{L}_{min} + c_2$
else $\Delta r \leftarrow 0$
return Δr

We can extend the above algorithm to deal with obstacles that impede tool or object motion, by modifying the utility function and the action space. The detailed formulation and algorithm can be found in S4.

4.4 Tool Use Controller: Same Controller as Without Tool

After identifying the best grasp and functionality poses on the tool, the robot needs to use the tool to perform the task. Since we are dealing with Category I tools, tool use involves similar actions as what the robot would perform with its end effector. Thus, the robot can utilize the same controller to perform the task with the tool as without, but with an update to the kinematic model of its end effector to include the tool. This is similar to the popular notion of tool embodiment in humans [51]. When the robot is holding the tool, the new kinematics of the arm is updated to:

$$X_{tool} = T_{ee\to tool} f(q) \tag{20}$$

where $T_{ee\to tool}$ is the transformation from the original end effector to the tool. Based on this new arm kinematics, motion planning is executed to ensure that the arm is also safe from collision with the environment when moving between desired functionality poses.

4.5 Simulation Setup for Tool Use Experiments

The simulation study is based on a planar 3-DoF robot moving on a horizontal plane, and the purpose is to study the augmentation optimization algorithm extensively with different tool forms. The 3-DOF planar robot arm with link lengths $(0.35, 0.35, 0.05)$m, a cylindrical object with diameter 0.08m, a cylindrical obstacle with diameter 0.1m for one of the tasks, and a channel 0.16m wide for one of the tasks. The base of the robotic arm is fixed at the origin.

The primary task studied involves pushing a cylindrical object to a goal position on a horizontal plane, and the secondary task is to avoid any obstacle in the environment. An illustration of the tasks is shown in Fig. S5. Specifically, the robot is required to perform 3 tasks as follows:

- **Task 1**: Moving the object at p_{obj}=$(0.8, -0.2)m$ to a target at p_{tar}=$(0.5, 0)m$.
- **Task 2**: Moving the object at p_{obj}=$(0.6, -0.4)m$ to a target at p_{tar}=$(0.5, 0)m$ with an obstacle blocking the shortest path between the object and target.
- **Task 3**: Moving the object at p_{obj}=$(0.5, 0)m$ into a channel to reach a target p_{tar}=$(0.65, 0)m$ in a 0.096m-wide channel.

The desired functionality pose for the task, X_{task}, is given by:

$$\begin{aligned} X_{task} &= \begin{bmatrix} R_{task} & p_{task} \\ 0 & 1 \end{bmatrix} \\ R_{task} &= \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \quad p_{task} = p_{obj} - r_{obj}\frac{e}{\|e\|} \\ e &= \begin{bmatrix} e_x \\ e_y \end{bmatrix} = p_{tar} - p_{obj}, \quad \phi = \arctan\frac{e_y}{e_x} \end{aligned} \tag{21}$$

where r_{obj} is the radius of the object.

We generated pairs of via point candidates flanking both sides of the obstacle when going along the object-to-target vector e:

$$p_v^{i,j} = p_{obs} + \text{sgn}(j - 1.5)d_i(r_{obj} + r_{obs})\hat{e}_\perp, \quad i = 1, ..., N_c, \quad j = 1, 2 \tag{22}$$

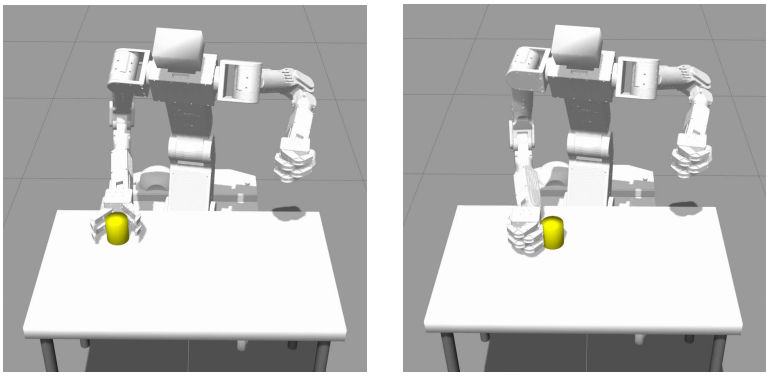
where sgn($\bullet$) is the signum function, N_c the number of candidate pairs, $\hat{e}_\perp$ a unit vector perpendicular to e, $d_i > 1$ a parameter determining the distance of the candidate pair from the obstacle, and r_{obs} the radius of the obstacle respectively. For simplicity, we set $N_c = 1$.

Three different 2D tools, illustrated in Fig. 4, were considered. For simplicity, perception of push and grasp candidates was omitted in this simulation study. Push candidates were placed at $0.02m$ intervals along the 2D tool edges, and grasp candidates $0.045m$ apart. At each grasp candidate, the neighboring push candidates were considered invalid, since they were covered by the robot end effector (during the grasp) and could not be used for pushing. These invalid push candidates were not given rewards if visited during the MCTS simulations branching from that grasp candidate.

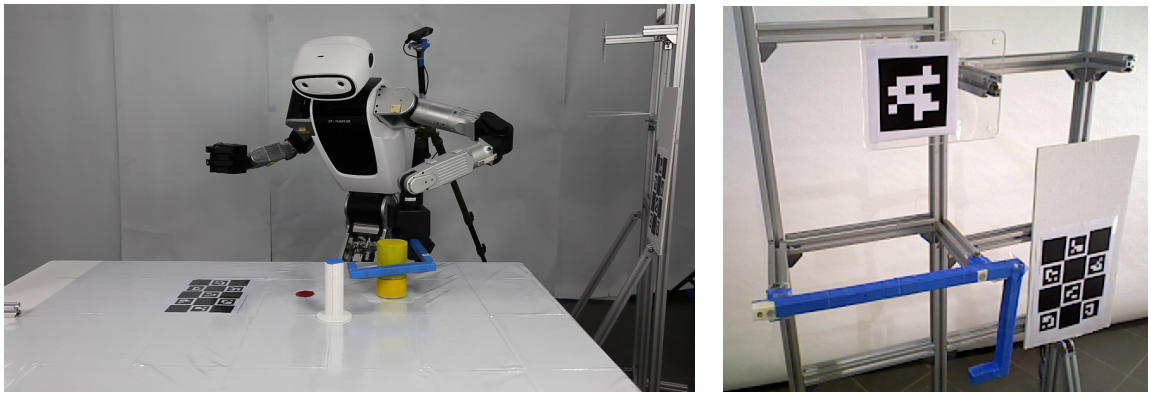
4.6 Real Experiment Setup for Tool Use

The manipulator platform used in the real experiment is the *Olivia III* robot (Fig. 6c) comprising a 3-DOF torso, 2-DOF head with an RGBD sensor, and dual articulated 7-DOF arms with 4-fingered grippers. We used a combination of arm and torso for manipulating objects and tools.

The experiment environment (Fig. 6c) included a $2m \times 1m$ table surface and a tool rack beside the robot. The apparatus on the table included a cylindrical object (diameter $0.08m$, height $0.1m$), a flat goal disc (diameter $0.05m$), and an obstacle (diameter $0.04m$, height $0.15m$). The robot was tasked with pushing the object from its initial position to the target position on the table, with and without an obstacle, and worked with a 3D-printed L-shaped tool.



(a) (b)



(c) (d)

Figure 6: For *learning functionality features*, the simulated *Olivia III* robot uses (a) an open hand to push forward, and (b) the side of its palm to push to its left. For the *real robot experiment* on tool recognition and use, (c) shows the robot and its environment including a table top with Charuco marker, tool rack, and external cameras for assisting perception, and (d) shows the front view of the tool rack showing Aruco marker for tool segmentation and Charuco marker view registration.

For ease of implementation, 3 RGBD cameras were used. An external camera was mounted above the table for detecting the object, target and obstacles on the table. Another external camera was placed behind the robot to detect the tool pose on the rack as well as the functionality features on the tool. The onboard camera registered the poses of all entities in the robot's frame with the help of 2 *Charuco* markers on the table and tool rack. Object segmentation was based on Random Sample Consensus to extract the table plane and then clustering points above the plane. The object and goal clusters were identified by color (yellow and red respectively), and remaining clusters were considered obstacles. The tool was segmented from the tool rack scene by extracting the cluster of points in front of a vertical plane marked by an *Aruco* marker (Fig. 6d). As shown in Supplementary Fig. S7, the robot perception module was able to localize the object, goal, obstacle, and tool.

The segmented tool point cloud was used to detect the functionality candidates using cube voxels with initial size of $1cm^3$. Since the tabletop pushing task is 2D in nature, we used a simplified 2D representation of the tools (Fig. 3) by extracting the intersection between the cross-sectional plane and the voxel surfels, and then taking the convex hull. Grasp candidates were placed at $0.03m$ interval along the medial axis of the 2D tool, excluding those near the tool ends and those that cannot fit the end effector (Supplementary Fig. S1). After tool augmentation was optimized, motion planning for grasping and moving the tool was performed using the ROS Moveit! library. To increase reach, we appended the torso joints to the arm kinematic chain.

The object, target marker, and obstacle were randomly presented by a human experimenter, and their positions estimated by the robot's perception (Supplementary Tables S2-S3). To complete the task, the robot first located the object, target and any obstacle. Then, it evaluated the feasibility of using its gripper to complete the task, by planning the motion. If no feasible plan is found, the robot turned towards the tool rack, and perceived the scene in order to segment the tool, estimate the tool pose, and detect the functionality and grasp candidates. After tool augmentation is optimized, the robot planned and performed the tool grasp and movements.

Besides the L-shaped tool, we also experimented with an umbrella to test the robot's ability to recognize and use real world objects as tools. Here, the setup is slightly different in that the umbrella rack is to the front of the robot (Fig. S8), and the robot relies on its onboard camera only, i.e. no external cameras. The task procedure is the same as that for the L-shaped tool. Perception of the umbrella is similar to that for the L-shaped tool, except that it relies only on the Aruco marker on the umbrella rack for segmentation. Charuco markers were not used because view registration with external cameras was not required. We placed grasp candidates at $0.09m$ interval along the medial axis of the 2D umbrella, excluding those near the tool ends and those that cannot fit the end effector. The object, target, and obstacle positions for the umbrella trial are shown in Supplementary Table S4.

5 Data availability

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

6 Code availability

All codes details will be made available by the corresponding authors on request.

References

[1] Köhler, W. *The mentality of apes (2nd rev. ed.) (E. Winter, Trans.)* (Routledge & Kegan Paul, 1927).

[2] Nabeshima, C., Kuniyoshi, Y. & Lungarella, M. Towards a model for tool-body assimilation and adaptive tool-use. In *2007 IEEE 6th International Conference on Development and Learning*, 288–293 (2007).

[3] Stoytchev, A. Learning the affordances of tools using a behavior-grounded approach. In *Proceedings of the 2006 International Conference on Towards Affordance-Based Robot Control*, 140–158 (Springer-Verlag, Berlin, Heidelberg, 2006).

[4] Kemp, C. C. & Edsinger, A. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *5th IEEE International Conference on Development and Learning* (2006).

[5] Brown, S. & Sammut, C. Tool use learning in robots. *Advances in Cognitive Systems: Papers from the 2011 AAAI Fall Symposium (FS-11-01)* (2011).

[6] Nishide, S., Tani, J., Takahashi, T., Okuno, H. G. & Ogata, T. Tool-body assimilation of humanoid robot using a neurodynamical system. *IEEE Transactions on Autonomous Mental Development* **4**, 139–149 (2012).

[7] Gonçalves, A., Saponaro, G., Jamone, L. & Bernardino, A. Learning visual affordances of objects and tools through autonomous robot exploration. In *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 128–133 (2014).

[8] Nguyen, K. N., Yoo, J. & Choe, Y. Speeding up affordance learning for tool use, using proprioceptive and kinesthetic inputs. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (2019).

[9] Saito, N., Ogata, T., Funabashi, S., Mori, H. & Sugano, S. How to select and use tools? : Active perception of target objects using multimodal deep learning. *IEEE Robotics and Automation Letters* **6**, 2517–2524 (2021).

[10] Kawaharazuka, K., Okada, K. & Inaba, M. Adaptive robotic tool-tip control learning considering online changes in grasping state. *IEEE Robotics and Automation Letters* **6**, 5992–5999 (2021).

[11] Arsenio, A. Learning task sequences from scratch: applications to the control of tools and toys by a humanoid robot. In *Proceedings of the 2004 IEEE International Conference on Control Applications, 2004.*, vol. 1, 400–405 Vol.1 (2004).

[12] Li, W. & Fritz, M. Teaching robots the use of human tools from demonstration with non-dexterous end-effectors. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 547–553 (2015).

[13] Wicaksono, H. & Sammut, C. Relational tool use learning by a robot in a real and simulated world. In *Australasian Conference on Robotics and Automation (ACRA)* (2016).

[14] Jain, R. & Inamura, T. Learning of tool affordances for autonomous tool manipulation. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, 814–819 (2011).

[15] J. Brawer, M. Q. & Scassellati, B. A causal approach to tool affordance learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8394–8399 (2020).

[16] Kokic, M., Stork, J. A., Haustein, J. A. & Kragic, D. Affordance detection for task-specific grasping using deep learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 91–98 (2017).

[17] Nguyen, A., Kanoulas, D., Caldwell, D. G. & Tsagarakis, N. G. Detecting object affordances with convolutional neural networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2765–2770 (2016).

[18] Fitzgerald, T., Short, E., Goel, A. & Thomaz, A. Human-guided trajectory adaptation for tool transfer. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '19, 1350–1358 (International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2019).

[19] Fang, K. *et al.* Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research* **39**, 202–216 (2020).

[20] Gajewski, P. *et al.* Adapting everyday manipulation skills to varied scenarios. In *2019 International Conference on Robotics and Automation (ICRA)*, 1345–1351 (2019).

[21] Hoffmann, H. *et al.* Adaptive robotic tool use under variable grasps. *Robotics and Autonomous Systems* **62**, 833–846 (2014). URL `https://www.sciencedirect.com/science/article/pii/S0921889014000244`.

[22] Kroemer, O., Ugur, E., Oztop, E. & Peters, J. A kernel-based approach to direct action perception. In *2012 IEEE International Conference on Robotics and Automation*, 2605–2610 (2012).

[23] Chu, V., Fitzgerald, T. & Thomaz, A. L. Learning object affordances by leveraging the combination of human-guidance and self-exploration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 221–228 (2016).

[24] Fitzgerald, T., Goel, A. & Thomaz, A. Modeling and learning constraints for creative tool use. *Frontiers in Robotics and AI* **8** (2021).

[25] Nair, L., Srikanth, N. S., Erickson, Z. M. & Chernova, S. Autonomous tool construction using part shape and attachment prediction. In *Proceedings of Robotics: Science and Systems (RSS '19)* (2019).

[26] Nair, L. & Chernova, S. Feature guided search for creative problem solving through tool construction. *Frontiers in Robotics and AI* **7** (2020). URL `https://www.frontiersin.org/article/10.3389/frobt.2020.592382`.

[27] Sinapov, J. & Stoytchev, A. Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *2008 7th IEEE International Conference on Development and Learning*, 91–96 (2008).

[28] Park, S., Baek, J., Kim, S. & Park, J. Rigid grasp candidate generation for assembly tasks. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 589–594 (2020).

[29] Mousavian, A., Eppner, C. & Fox, D. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)* (2019).

[30] Tee, K. P., Li, J., Chen, T. P., Wan, K. W. & Ganesh, G. Towards emergence of tool use in robots: Automatic tool recognition and use without prior tool learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6439–6446 (2018).

[31] Head, H. & Holmes, G. Sensory disturbances from cerebral lesions. *Brain* **34**, 102–254 (1912).

[32] Berti, A. & Frassinetti, F. When far becomes near: Remapping of space by tool use. *J. Cogn. Neuroscience* **12**, 415–420 (2000).

[33] Jacobs, S., Bussel, B., Combeaud, M. & Roby-Brami, A. The use of a tool requires its incorporation into the movement: Evidence from stick-pointing in apraxia. *Cortex* **45**, 444 – 455 (2009). URL http://www.sciencedirect.com/science/article/pii/S0010945208000397.

[34] Maravita, A. & Iriki, A. Tools for the body (schema). *Trends in Cogn. Sci.* **8**, 79–86 (2004).

[35] Aymerich-Franch, L. & Ganesh, G. The role of functionality in the body model for self-attribution. *Neuroscience Research* **104**, 31 – 37 (2016). URL http://www.sciencedirect.com/science/article/pii/S0168010215002655. Body representation in the brain.

[36] Aymerich-Franch, L., Petit, D., Ganesh, G. & Kheddar, A. Object touch by a humanoid robot avatar induces haptic sensation in the real hand. *Journal of Computer-Mediated Communication* **22**, 215–230 (2017).

[37] Aymerich-Franch, L., Petit, D., Ganesh, G. & Kheddar, A. Non-human looking robot arms induce illusion of embodiment. *International Journal of Social Robotics* **9**, 479–490 (2017). URL https://doi.org/10.1007/s12369-017-0397-8.

[38] Vock, R., Dieckmann, A., Ochmann, S. & Klein, R. Fast template matching and pose estimation in 3d point clouds. *Computers & Graphics* **79**, 36–45 (2019). URL https://www.sciencedirect.com/science/article/pii/S0097849319300081.

[39] Park, K., Patten, T., Prankl, J. & Vincze, M. Multi-task template matching for object detection, segmentation and pose estimation using depth images. In *International Conference on Robotics and Automation (ICRA)*, 7207–7213 (2019).

[40] Hinterstoisser, S., Lepetit, V., Rajkumar, N. & Konolige, K. Going further with point pair features. In Leibe, B., Matas, J., Sebe, N. & Welling, M. (eds.) *Computer Vision – ECCV 2016*, 834–848 (Springer International Publishing, Cham, 2016).

[41] Peng, X. B., Andrychowicz, M., Zaremba, W. & Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810 (2018).

[42] Shumaker, R. W., Walkup, K. R. & Beck, B. B. *Animal tool behavior: The use and manufacture of tools by animals (Rev. and updated ed.)* (Johns Hopkins University Press, 2011).

[43] Boesch, C. & Boesch, H. Tool use and tool making in wild chimpanzees. *Folia primatologica* **54**, 86–99 (1990).

[44] Hunt, G. R. & Gray, R. D. The crafting of hook tools by wild new caledonian crows. *Proceedings. Biological sciences* **271**, S88–S90 (2004).

[45] Weir, A. A. S., Chappell, J. & Kacelnik, A. Shaping of hooks in new caledonian crows. *Science* **297**, 981 (2002).

[46] Ryde, J., Dhiman, V. & Platt, R. Voxel planes: Rapid visualization and meshification of point cloud ensembles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Tokyo, Japan, 2013).

[47] Drost, B., Ulrich, M., Navab, N. & Ilic, S. Model globally, match locally: Efficient and robust 3d object recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 998–1005 (2010).

[48] Hodaň, T. *et al.* BOP: Benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV)* (2018).

[49] Chaslot, G., Bakkes, S., Szita, I. & Spronck, P. Monte-carlo tree search: A new framework for game ai. In *The Fourth AAAI Conference on Artifical Intelligence and Interactive Digital Entertainment (AIIDE)*, 216–217 (AAAI Press, 2008). URL `http://dl.acm.org/citation.cfm?id=3022539.3022579`.

[50] Kocsis, L. & Szepesvári, C. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, 282–293 (Springer-Verlag, Berlin, Heidelberg, 2006). URL `http://dx.doi.org/10.1007/11871842_29`.

[51] Ganesh, G., Yoshioka, T., Osu, R. & Ikegami, T. Immediate tool incorporation processes determine human motor planning with tools. *Nature Communications* (2017).

[52] Saponaro, G. *et al.* Learning at the ends: From hand to tool affordances in humanoid robots. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 331–337 (2017).

[53] Jamone, L. *et al.* Affordances in psychology, neuroscience, and robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems* **10**, 4–25 (2018).

[54] Mar, T., Tikhanoff, V. & Natale, L. What can i do with this tool? self-supervised learning of tool affordances from their 3-d geometry. *IEEE Transactions on Cognitive and Developmental Systems* **10**, 595–610 (2018).

[55] Abderrahmane, Z., Ganesh, G., Crosnier, A. & Cherubini, A. Haptic Zero-Shot Learning: Recognition of Objects Never Touched Before. *Robotics and Autonomous Systems* **105**, 11–25 (2018).

[56] Lampert, C. H., Nickisch, H. & Harmeling, S. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**, 453–465 (2014).

[57] Yu, X. & Aloimonos, Y. Attribute-based transfer learning for object categorization with zero/one training example. In Daniilidis, K., Maragos, P. & Paragios, N. (eds.) *Computer Vision – ECCV 2010*, 127–140 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010).

[58] Holladay, R., Lozano-Pérez, T. & Rodriguez, A. Force-and-motion constrained planning for tool use. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7409–7416 (2019).

[59] Chen, H., Wan, W. & Harada, K. Combined task and motion planning for a dual-arm robot to use a suction cup tool. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 446–452 (2019).

[60] Labbe, Y. *et al.* Monte-carlo tree search for efficient visually guided rearrangement planning. *IEEE Robotics and Automation Letters* (2020).

[61] Silver, D., Huang, A. & Maddison et al., C. Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

[62] Paxton, C., Raman, V., Hager, G. D. & Kobilarov, M. Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6059–6066 (2017).

[63] Toussaint, M. & Lopes, M. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4044–4051 (2017).

[64] M.Toussaint, Allen, K. R., Smith, K. A. & B.Tenenbaum, J. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems* (2018).

[65] Chen, Z. *et al.* An intermediate point obstacle avoidance algorithm for serial robot. *Advances in Mechanical Engineering* **10** (2018).

[66] Bruijnen, D., van Helvoort, J. & van de Molengraft, R. Realtime motion path generation using subtargets in a changing environment. In *American Control Conference*, 4243–4248 (2006).

Supplementary Materials for:

A framework for tool cognition in robots without prior tool learning or observation

S1 Related Works and Chosen Implementation

In our previous work [30], we introduced these concepts. However, our previous work and a work by Saponaro and colleagues [52] presented very basic algorithms which considered the overall shape of the objects and not specific features. Furthermore, the tool grasp and tool tip (functionality) locations were assumed to be restricted in certain regions of the tool, and demonstration of the robot's tool use ability was limited to specific toy tools. In this paper, we provide a more complete framework for tool cognition, which enables a robot to not only recognize task-critical *functionality* features on objects, but also optimize its potential *augmentation* in the presence of grasp and movement constraints (due to obstacles), with the ultimate goal of recognizing and using a vast range of daily-life objects as tools to accomplish various tasks

Our robot tool cognition framework is described in detail in the main text. In this section we give a brief description of the chosen procedures in this manuscript, and related previous works.

S1.1 Terminologies: Functionality and Affordance

Tool use studies have often talked about affordances offered by tools [15, 53, 54]. According to [14], tool affordance refers to the *"awareness within robot about the different kind of effects it can create in the environment using a tool. It incorporates the association of effects and abstract geometrical properties of tools with the perception of the initial environment and the executed action."* Our definition of functionality is a subset of the above, and specifically refers to interaction with the environment enabled by certain features of a given tool. Therefore, our proposed tool representation allows for the prediction of environmental effects across different tools.

With the advent of deep convolutional neural networks (CNNs), many studies on end-to-end object affordance learning and detection based on CNNs have been reported in the literature (see e.g. [16, 17]). These approaches involve learning directly on instances of the object classes and rely on copious amounts of labelled image/CAD data to achieve high accuracy rates of affordance detection. Tests of unseen objects are drawn from the same set of object classes used for training. For example, to detect a rake affordance in an umbrella, the robot needs to train on instances of a class of umbrella-like objects. Our approach is different in that learning of functionality features is performed on only on the robot's limb and while performing actions with its prior skills, and yet it is able to extrapolate the detection of similar features on unseen objects that do not belong to the same object class as a 'limb'.

The functionality recognition procedure we propose may be viewed similar to zero shot learning [55, 56, 57] proposed for visual and haptic interactions and during transfer learning of tools[18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. These procedures suggest to explore and learn task relevant features, in this case from one set of tools, to recognize other tools for the same task. However, by definition, zero shot

learning or transfer learning requires the robot to have atleast some prior tool experience, and hence also posses skill to pick up and explore these tools. This is fundamentally different from our proposed framework, that requires zero experience or observation of tools.

S1.2 Tool-Use Planning

Many existing approaches on planning for tool use focus on planning where to grasp the tool while the part of the tool that interacts with objects or the environment is known or fixed [58, 59]. In this work, our goal is the optimization of tool augmentation, which is the joint optimization of both grasp and interaction (e.g. push) locations, subject to external task/environment constraints and the robot's internal constraints. To solve this combinatorial optimization problem, we employ the Monte Carlo Tree Search (MCTS) method, which is advantageous because it efficiently handles the exploration-exploitation trade-off through iterative learning of the value function with only a reward signal [60]. The MCTS method, popularized by AlphaGo for game-playing [61], has been studied in the robotics domain for autonomous vehicle behavior planning [62], multi-agent collaborative manipulation [63], rearrangement planning [60], and tool-use planning [64]. In [64], multi-bound tree search was used, under a general geometric-logic programming framework, to jointly optimize the manipulator motion path and decision parameters (e.g. grasp pose, hit angle) when full information is given from a simulated world. Our work applies MCTS to optimize the decision parameters in tool augmentation (including grasp and interaction points, and via points for obstacle avoidance), which then feed into state-of-the-art manipulator motion planners. This reduces the complexity of the solution and allows us to validate on a physical robotic system recognizing and manipulating real world objects as tools.

S2 Grasp Candidates

While grasp candidates are just as important as functionality candidates as inputs to the algorithm, developing algorithms for generation of grasp candidates is out of the scope of this paper. Interested readers may refer to grasp planning/generation techniques in the literature, e.g. [28, 29]. For simplicity and without loss of generality in our algorithm for tool augmentation optimization, we generated grasp candidates by placing them at regular intervals along the medial axis of the potential tool (see (22) in Sec. 4.5).

S3 Functionality Feature Detection

Additional results for detection of functionality features on potential tools for pushing and lifting tasks are shown in Figs. S2 and S3.

S4 Handling Obstacles in Tool Augmentation Optimization

To circumvent obstacles in the environment impeding the tool and/or object motion, via points are generated in the vicinity of the obstacles. By visiting feasible via points, the obstacle can be avoided, and the task completed using the tool. In some cases, there can be a switch of tool pose between reaching a via point and moving to the target or next via point, resulting in a change of end-effector pose and arm

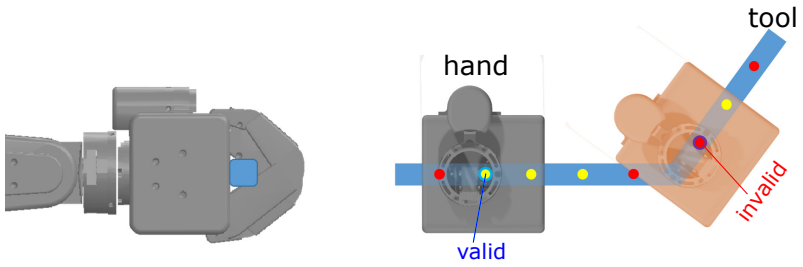


Figure S1: Illustration of grasp candidate placement. Grasp candidates are placed at regular intervals along the medial axis of the 2D tool. Locations near the tool ends, and those that cannot fit the end effector, are invalid and indicated in red. Valid candidates are indicated in yellow.

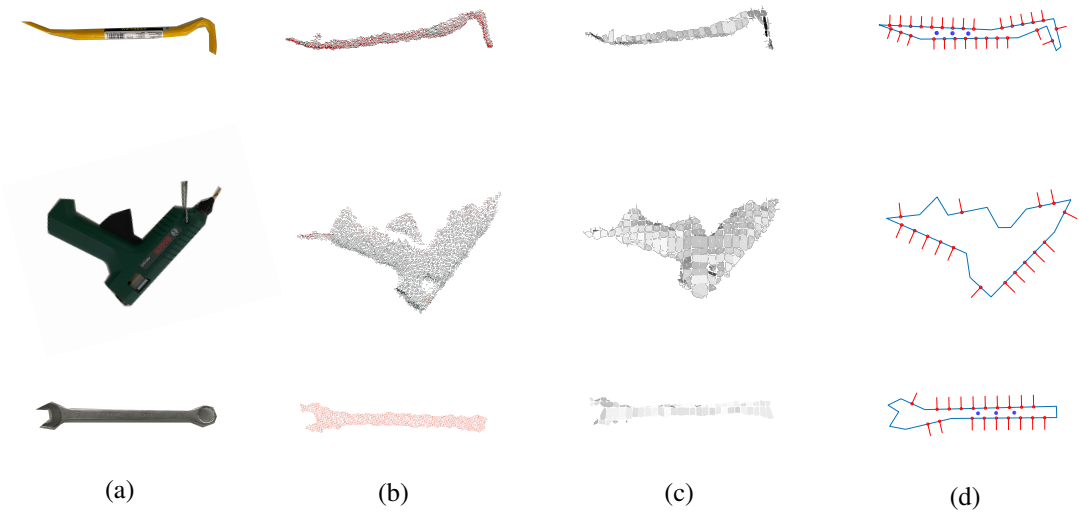


Figure S2: Perception of push functionality on tools. (a) Visual appearance of tools (b) Segmented point clouds of tools. (c) Voxel surfels on tools. (d) Detected functionality candidates (red circles and 'spikes') and prescribed grasp candidates (blue dots) on tools.

(a) (b) (c) (d)

Figure S3: Perception of lift functionality on tools. (a) Visual appearance of tools (b) Segmented point clouds of tools. (c) Voxel surfels on tools. (d) A representative functionality candidate is shown as a circular patch with a direction vector at the center of the patch pointing out of the page. To avoid clutter, the other functionality candidate is shown simply as red dots representing the center of the circular patch. The prescribed grasp candidate is shown as a blue dot on the tool handle.

joint configuration. For example, consider the task of pushing an object to a target location using a stick. If an obstacle lies in the path between the object and the target and a via point is generated, the stick first pushes the object to the via point, and then to the target. After the via point is reached, there can be a change of the functionality pose on the stick such that the object is pushed to the target more efficiently (see task 2 of Fig. 4). In this paper, for simplicity and without loss of generality, we generated via point candidates near each obstacle with a simple rule, with details in Sec. 4.5. It is possible to use other more sophisticated methods of via point generation e.g. [65, 66], but that is out of the scope of this study.

To this end, for optimization of tool augmentation, besides maximizing manipulability, we also minimize end effector pose changes and arm joint configuration changes. This results in the maximization of the following utility function subject to the constraints C1-C4:

$$\mathcal{L}_{obs} = \alpha \det(JJ^T) - \beta \sum_{i}^{N_v} \left(\|\Delta X_i\|^2 + \|\Delta q_i\|^2 \right) \tag{23}$$

where N_v the number of via points, ΔX_i the change of end effector position at the ith via point, Δq_i the change of joint configuration at the ith via point, and α, β positive weighting constants.

Since the via points affects the tool path and hence the grasp and functionality poses, they form part of the solution $\theta_{obs} = (\theta_g, \theta_f, \theta_v)$ to be found by maximizing the utility function $\mathcal{L}_{obs}$:

$$\theta_{obs} \quad = \quad \underset{\theta_g \in \Omega_g, \theta_f \in \Omega_f, \theta_v \in \Omega_v}{\operatorname{argmax}} \quad \mathcal{L}_{obs} \tag{24}$$

where $\Omega_v = \{\theta_{v_1}, ..., \theta_{v_n}\}$ is the set of via point candidates.

The action space for the original MCTS formulation (without obstacles) is shown in Fig. S4. For obstacle avoidance, additional decision layers are appended to form a new action space $\mathcal{A}_{obs}$:

$$\mathcal{A}_{obs} = G \times \Phi \times \prod_{i=1}^{N_v} (S_i \times P_i \times V_i) \times S_{N_v+1} \times P_{N_v+1} \tag{25}$$

where V_i is a set of location candidates for the ith via point, S_i and P_i the tool surface segment and functionality location before the ith via point (or equivalently after the $(i-1)$th via point). Here, we

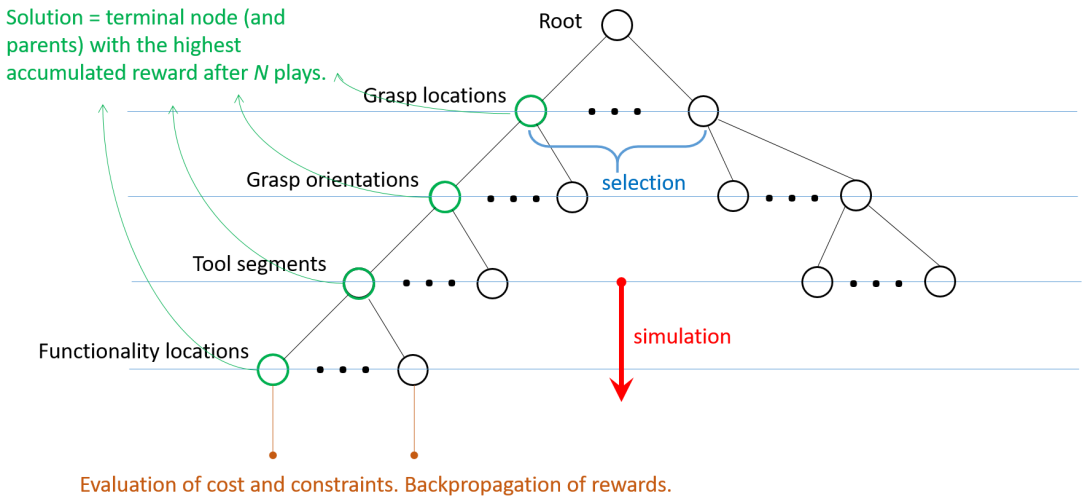


Figure S4: Search tree for finding the best combination of robot-tool and tool-environment interface candidates using MCTS.

precluded any change in the grasp pose at the via points to enable smoother and more efficient task performance. Algorithms 5-6 illustrates the MCTS algorithm for finding the best combination of robot-tool and tool-environment interface candidates in the presence of an obstacle.

Algorithm 5 MCTS for Augmentation Optimization (with Obstacle)

Input: Action space $\mathcal{A}_{obs}$, maximum number of iterations N_{sim}, desired task functionality poses $\{X^{i,j}_{task}\}_{i=1:n_v,\, j=1:n_f}$.

Output: Best action $a^* = \{g^*, \phi^*, v^*, s_1^*, p_1^*, \ldots, s_{n_v+1}^*, p_{n_v+1}^*\} \in \mathcal{A}_{obs}$.

$r_{best} \leftarrow 0$
$r_{terminal} \leftarrow 0$
for $i = 1$ to N_{sim} **do**
    $leaf \leftarrow$ Selection&Expansion$(root)$
    $a \leftarrow$ Simulation$(leaf)$
    $r_{terminal} \leftarrow r_{terminal} +$ RewardEvaluation $\left(a, \{X^{i,j}_{task}\}_{i=1:n_v,\, j=1:n_f}\right)$
    $r_{best} \leftarrow \max(r_{best}, r_{terminal})$
    Backpropagation$(root, r_{terminal})$
$a^* \leftarrow$ BestAction(r_{best})
return a^*

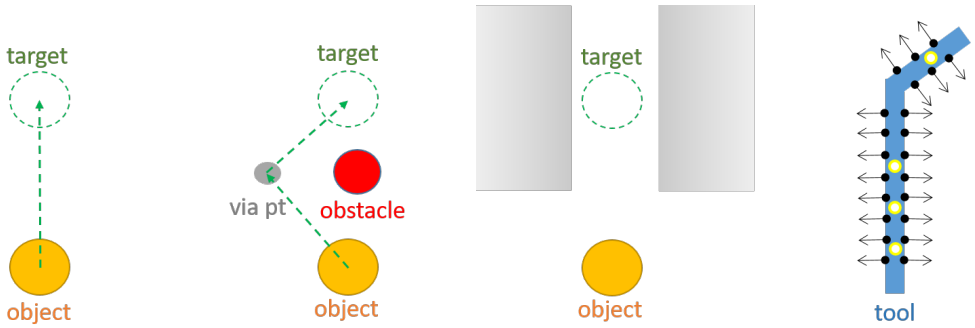
Algorithm 6 RewardEvaluation (with Obstacle)

Input: Action $a = \{g, \phi, v, s_1, p_1, \ldots, s_{n_v+1}, p_{n_v+1}\}$, desired task functionality poses $\{X^{i,j}_{task}\}_{i=1:n_v,\ j=1:n_f}$.
Output: Reward Δr

$P_r \leftarrow$ true
$\theta_g \leftarrow$ ComputeLocGraspPose(g, ϕ)
$\mathcal{L}_{min} \leftarrow$ large_number
for $i = 1$ to $n_v + 1$ **do**
    $\theta^i_f \leftarrow$ ComputeLocFuncPose(s_i, p_i)
    for $j = 1$ to n_f **do**
        $T^{i,j}_{w\to t} \leftarrow X^{i,j}_{task}(\theta^i_f)^{-1}$
        $q^{i,j} \leftarrow$ InvKin$(T^{i,j}_{w\to t}\theta_g)$
        $\mathcal{L}_{min} \leftarrow \min\left(\mathcal{L}_{min}, \det\left(J(q^{i,j})J^T(q^{i,j})\right)\right)$
        $P_r \leftarrow P_r \wedge$ CheckCollision$(T^{i,j}_{w\to t}) \wedge$ CheckJointLim$(q^{i,j})$
if P_r is true **then**
    $\Delta r \leftarrow c_1\mathcal{L}_{min} + c_2$
else $\Delta r \leftarrow 0$
return Δr

S5 Simulated Experiment

Illustrations of the 3 tasks in the simulated experiment of Sec. 4.5 are provided in Fig. S5. For tasks 1 and 3, the number of push candidates on Tools 1-3 are 40, 68, and 55 respectively, and the number of grasp candidates on Tools 1-3 are 6, 12, and 7 respectively. For task 2 dealing with obstacle avoidance, to factor in the branching of possibilities as a result of the 2 via point candidates, the numbers of first-push candidates are doubled for all tools, i.e. 80, 136, and 110 respectively for Tools 1-3. The numbers of second-push candidates remain the same.



(a) (b) (c) (d)

Figure S5: 2D representation of the simulation tasks, including (a) Task 1: pushing an object directly to the target, (b) Task 2: pushing object to target while avoiding an obstacle, and (c) Task 3: pushing object to a target in a channel. (d) An example tool with the grasp (yellow dots) and functionality (black arrows) candidates on the tool surface segments.

Table S1 compares the number of solutions found using the proposed tool cognition framework with that in [30]. The 3 tasks and 3 tools are the same ones shown in Fig. 4. From Table S1b, solutions based on [30] were found by finding regions on the tool that match the shape template for the push functionality, and then testing inverse kinematics feasibility for wielding the tool at each combination of grasp and push locations. For fairness of comparison, the same grasp and via point candidates were given for the 2 methods. Unlike [30], which uses hand shape descriptors to represent task functionality, our proposed framework employs less restrictive features that are learnt through hand-object interactions in an offline development phase. As a result of these learnt functionality features, tool augmentation can be optimized by finding the best combination of grasp and push locations on the tool from a larger pool of feasible combinations. For [30], less options, in general, are available due to the restrictive shape feature. As a result, it yields less solutions than our proposed framework for Tasks 1 and 2. Task 3 yields the same number of solutions because the push location options are inherently limited, due to the task itself, to the extent that the solutions from our framework are identical to those offered by the shape features of [30].

Table S1: Comparison of the number of solutions found when using a) the proposed framework and b) method in [30].

(a)

	Tool 1	Tool 2	Tool 3
Task 1	8	42	16
Task 2	17	21	32
Task 3	12	11	7

(b)

	Tool 1	Tool 2	Tool 3
Task 1	4	8	7
Task 2	5	6	12
Task 3	12	11	7

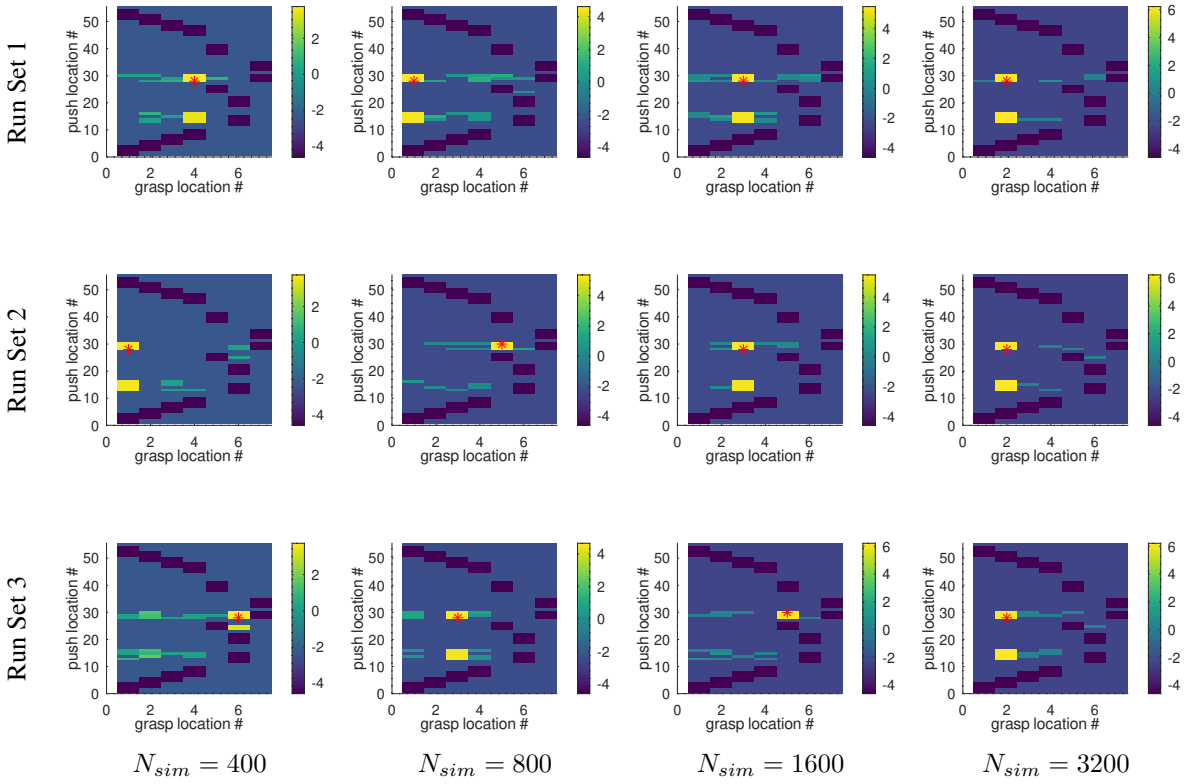


Figure S6: Evolution of the heat maps with the number of simulations, taking task 1 and tool 3 as an exemplar. Each heat map is a an individual run instance with number of MCTS simulations, N_{sim} at 400, 800, 1600, or 3200. Three representative run sets under the same conditions are presented. For the runs with $N_{sim} = 400$ and $N_{sim} = 800$, the solution regions, indicated by warmer colors, as well as the optimal solutions, are very different across the runs. When $N_{sim} = 1600$, the solution regions and optimal solutions are similar between Runs 1 and 2 but different for Run 3. When $N_{sim} = 3200$, the solution regions and optimal solutions are similar for all 3 runs. This shows that with sufficient number of MCTS simulations, the optimal solution tends to converge. However, if the number of simulations is small, the optimal solution can have high variability from one run to another. Similar results can be obtained for other combinations of tasks and tools.

S6 Real Robot Experiment

Table S2: Positions of object and target during trials without obstacle and using L-shaped tool. The world coordinate frame was centered at the robot's base joint, with the $x-$axis pointing towards the table in front of the robot, the $y-$axis towards the left arm, and $z-$axis upwards. Units in *m*.

Position	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**
p_{obj}	[0.89,0.16]	[0.75,-0.47]	[0.90,-0.16]	[0.73,-0.17]	[0.94,-0.21]
p_{tar}	[0.94,-0.09]	[0.90,-0.33]	[0.76,-0.15]	[0.97,-0.28]	[1.02,0.02]

Table S3: Positions of object, target, and obstacle during trials with obstacle and using L-shaped tool. The world coordinate frame was centered at the robot's base joint, with the $x-$axis pointing towards the table in front of the robot, the $y-$axis towards the left arm, and $z-$axis upwards. Units in *m*.

Position	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**
p_{obj}	[0.78,0.28]	[0.99,0.19]	[0.63,-0.07]	[0.68,-0.07]	[0.92,-0.07]
p_{tar}	[1.07,-0.03]	[0.68,-0.08]	[1.03,0.20]	[1.01,0.19]	[0.95,0.26]
p_{obs}	[0.94,0.16]	[0.79,-0.01]	[0.82,0.03]	[0.83,0.09]	[0.92,0.09]

Table S4: Positions of object, target, and obstacle during trials with a real umbrella. The world coordinate frame was centered at the robot's base joint, with the $x-$axis pointing towards the table in front of the robot, the $y-$axis towards the left arm, and $z-$axis upwards. Units in *m*.

Position	**without obstacle**	**with obstacle**
p_{obj}	[0.85,0.4]	[0.55,0.4]
p_{tar}	[0.86,0.5]	[0.6,0.25]
p_{obs}	-	[0.79,0.35]

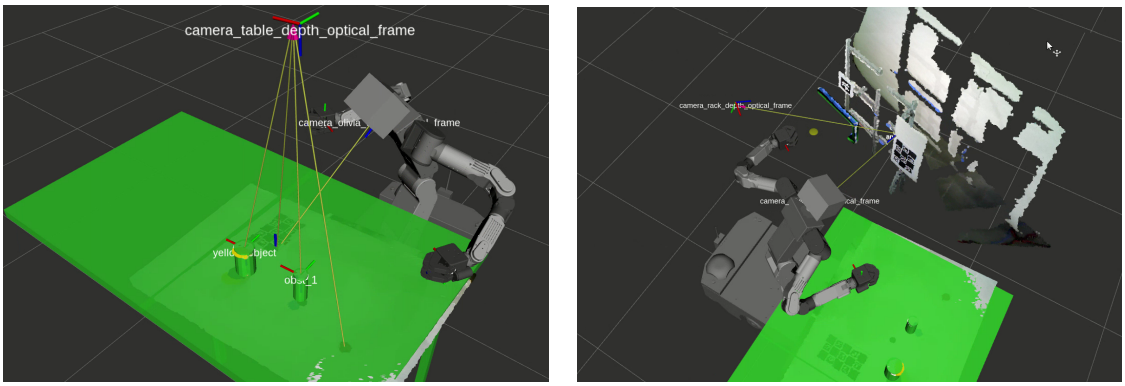


Figure S7: Robot perception of the object, target and obstacle on the table (left), and the tool on the rack (right), with the help of Charuco markers for multi-camera view registration.

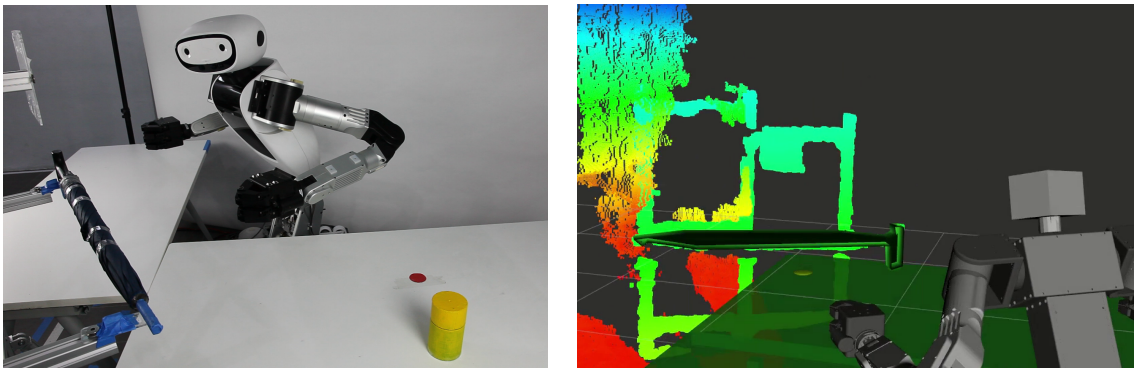


Figure S8: Setup for experiment with umbrella as tool (left) and perception of umbrella by robot (right).