Learning Agent-Aware Affordances for Closed-Loop Interaction with Articulated Objects.

Giulio Schiavi*, Paula Wulkop*, Giuseppe Rizzi, Lionel Ott, Roland Siegwart, Jen Jen Chung†

***Abstract*— Interactions with articulated objects are a challenging but important task for mobile robots. To tackle this challenge, we propose a novel closed-loop control pipeline, which integrates manipulation priors from affordance estimation with sampling-based whole-body control. We introduce the concept of agent-aware affordances which fully reflect the agent's capabilities and embodiment and we show that they outperform their state-of-the-art counterparts which are only conditioned on the end-effector geometry. Additionally, closed-loop affordance inference is found to allow the agent to divide a task into multiple non-continuous motions and recover from failure and unexpected states. Finally, the pipeline is able to perform long-horizon mobile manipulation tasks, i.e. opening and closing an oven, in the real world with high success rates (opening: 71%, closing: 72%).**

I. INTRODUCTION

In the future, autonomous mobile robots could relieve humans of tedious, repetitive manual tasks in a wide variety of environments like hospitals, homes or laboratories. Many daily tasks require interaction with articulated objects, for example to open the door of a dishwasher. This is especially challenging for a robotic agent, because of the complex system dynamics caused by the object's degrees of freedom and kinematic constraints.

A common approach is to estimate the kinematic and semantic properties of articulated objects from visual data [1]–[4] and leverage this information for planning and control [5], [6]. This two-staged approach often requires heuristics, for example defining the grasping point on the handle, limiting the flexibility to deal with unseen articulation types and object geometries. A more generic approach using affordances, i.e. where and how an agent can interact with an object, was recently explored [7], [8]. Given a point cloud of an articulated object, they use neural networks to predict point-wise interaction scores (*actionability*), which are used as priors for a downstream robotic controller. While these approaches show promising initial results, they neglect that affordances are always dependent on the agent's capabilities, which are defined by the hardware and the controller. Instead, the state-of-the-art models are trained on a disembodied gripper, disregarding the robot kinematics and joint limits. This can lead to the predictions of motions that are infeasible

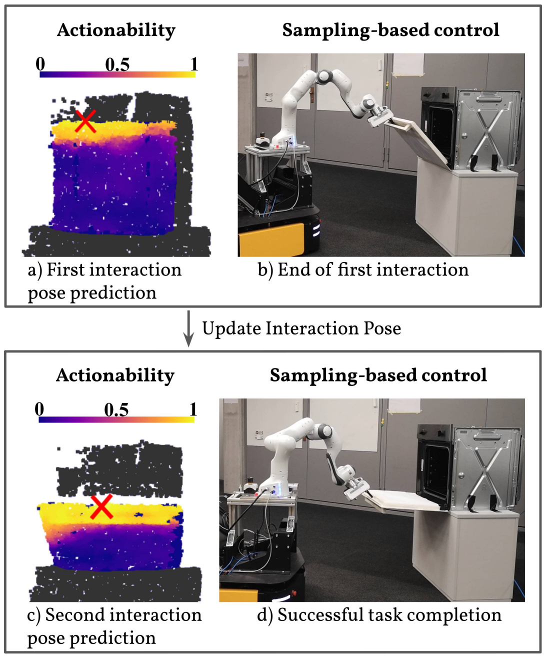
Actionability
Sampling-based control
0
0.5
1
a) First interaction pose prediction
b) End of first interaction
Update Interaction Pose
Actionability
Sampling-based control
0
0.5
1
c) Second interaction pose prediction
d) Successful task completion

Fig. 1: Real-world experiment of opening an oven in two motions. a) & c): Estimated actionability map where the red cross represents the selected interaction point. b) The first interaction pose becomes unfavorable, therefore an update is triggered. d): Successful task completion after the second interaction.

for the real robot. Furthermore, these pipelines query the affordance module only once and then keep the interaction pose and planned trajectory fixed. Due to this open-loop perception and planning setup, they cannot perform long-term tasks requiring a change of interaction pose when the robot reaches kinematic or joint limits.

To overcome these limitations, we propose a novel closed-loop pipeline combining agent-aware affordance perception with sampling-based whole-body control. Concretely, this work deals with the non-prehensile manipulation of articulated objects with one degree of freedom using a mobile manipulator equipped with a single, fixed cylindrical finger. Taking inspiration from the *Where2Act* framework [7], we train an artificial neural network to estimate point-level affordances from visual data, indicating the success likelihood for interactions at each point. The pose proposal with the highest affordance score is passed on to a sampling-based controller (based on [9], [10]), which then iteratively determines the best interaction trajectory. This setup is well suited for non-prehensile manipulation since the interaction location and trajectory are continuously adapted based on real-time feedback, enabling adaptive and robust task execution. In

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony).

* Equal contribution

Authors are with the Autonomous Systems Lab, ETH Zurich, Switzerland. Email: pwulkop@ethz.ch, giulio.schiavi@outlook.com.

†Also with the School of ITEE, The University of Queensland, Australia.

Project page: `https://paulawulkop.github.io/agent_aware_affordances` contrast to previous end-effector-aware approaches, we train our affordance inference network with data collected using the full model of our target robot platform, making our predictions fully agent-aware. Additionally, unlike previous approaches with a fixed interaction pose, our pipeline is able to re-evaluate the affordance model at any point during task execution, which allows the robot to execute long-term tasks where a change of interaction pose is required. In this work, we show in ablation experiments that agent-aware training significantly increases the quality of pose proposals, and that allowing the agent to change the interaction pose during the task increases the robustness of the integrated affordance-control pipeline. Additionally, we benchmark our method against VAT-Mart [8] as a state-of-the-art work and we perform experiments in the real world (Fig. 1), reaching a success rate of more than 70% for both fully opening and closing an oven door using a mobile manipulator. In summary, our contributions are:

- We formulate the concept of agent-aware object affordances which are conditioned on the full shape and kinematics of the robot.
- We propose a novel closed-loop manipulation framework that combines the concept of visual affordances with a sampling-based controller.

II. RELATED WORK

Control strategies for manipulating articulated objects: A common approach for the manipulation of articulated objects is to extract object properties like part segmentation and joint kinematics from visual data [1]–[4], [11] and use this to plan an interaction trajectory [12]–[14]. Mittal et al. [5] recently successfully used this approach by implementing a Model Predictive Controller (MPC) to track the feed-forward trajectory plan. However, MPC struggles with discontinuities caused by contact dynamics, such that a simplification of the problem is required, e.g. splitting the task into first achieving a stable grasp at a fixed interaction position and then executing a predefined motion based on known articulation kinematics. To alleviate these limitations, sampling-based control has recently emerged, allowing a more task-specific and complex interaction representation [9], [15], [16]. In previous work by Rizzi et al. [10], a sampling-based controller was successfully applied as a whole-body closed-loop controller for the non-prehensile manipulation of articulated objects. However, to decrease the size of the sampling space, a fixed end-effector interaction pose still has to be provided a priori for each object.

Affordances for manipulating articulated objects: An affordance is defined as the ability of an agent to perform an action with a target object in a given environment [17], [18]. In robotics, the concept of affordances is commonly applied either on an object-level, e.g. a dishwasher is *openable* [19], [20], or on a point-level to encode information about the object geometry and grasp possibilities [21]–[24]. In the current literature, affordances are usually learned from labeled visual data [25]–[27] or from self-supervised interactions [19], [20], [24], [28]. Recent works have applied this concept to the manipulation of articulated objects. VAT-Mart [8] and Where2Act [7] propose frameworks which learn affordances for robotic manipulation from interactions generated in a photo-realistic simulator [29]. A network is then trained to infer interaction points and trajectory proposals for downstream manipulation tasks from visual data. However, these approaches generate training data by simulating the interaction between a disembodied end-effector and a unit-sphere scaled object. This results in affordances that are not only unaware of the kinematics and control of the actual robot platform, but also of the realistic object size. Consequently, the trajectories derived from these approaches might be infeasible in real life, e.g. due to joint limits or collisions of the arm with the object. Additionally, they only query the visual model at the beginning of the interaction in an open-loop fashion, keeping the planned grasp location and trajectory constant during the interaction. UMPNet [30] on the other hand proposed a closed-loop affordance-based manipulation framework that can update the interaction direction during task execution. However, it still uses a fixed interaction point, determined at the start of the interaction using only end-effector-aware (agent-agnostic) affordances.

III. PROBLEM FORMULATION

We consider a physical system consisting of a mobile manipulator and an articulated object which is composed of two rigid bodies connected by a rotational or translational joint. We define the state $\mathbf{x} = [\mathbf{q}, \mathbf{o}, \dot{\mathbf{q}}, \dot{\mathbf{o}}]$ as the configurations of the robot and object and their time derivatives, where $\mathbf{q}$ consists of the position of the mobile base relative to the object as well as the joint angles of the manipulator. The robot kinematics and the object geometry and articulation are assumed to be provided a priori. Given a desired object configuration $\mathbf{o}^*$, the point cloud from the robot's point of view $\mathcal{C}$ and the current state observation $\mathbf{x}$, the overall objective is to find a robot control policy that generates joint velocity commands $\mathbf{u}$ such that $\mathbf{o} = \mathbf{o}^*$.

IV. AGENT-AWARE AFFORDANCE LEARNING

We propose a novel control pipeline combining an agent-aware affordance network with a sampling-based whole-body controller (Fig. 2). Given an object point cloud $\mathcal{C}$, from which the movable object part is extracted by applying a segmentation mask, and target configuration $\mathbf{o}^*$, we use an affordance neural network to select a favorable end-effector interaction pose (point $\mathbf{p} \in \mathbb{R}^3$ and orientation $\mathbf{R} \in SO(3)$). The interaction pose is then used by the sampling-based controller κ to generate joint velocity commands $\mathbf{u} = \kappa(\mathbf{x} \mid \mathbf{o}^*, \mathbf{p}, \mathbf{R})$. When the interaction fails or stops leading to any improvement of $\mathbf{o}$, a task scheduler triggers a pose update as described in Section IV-B.

A. Affordance-based Pose Module

Pose inference: Let $J(\mathbf{x}_{0:N}, \kappa(\mathbf{x}_{0:N-1}) \mid \mathbf{o}^*)$ be a reward function that, given the target object configuration $\mathbf{o}^*$, maps a trajectory of system states $\{\mathbf{x}_{0:N}\}$ and controller inputs $\{\mathbf{u}_{0:N-1}\}$ to a binary reward value. Given an object

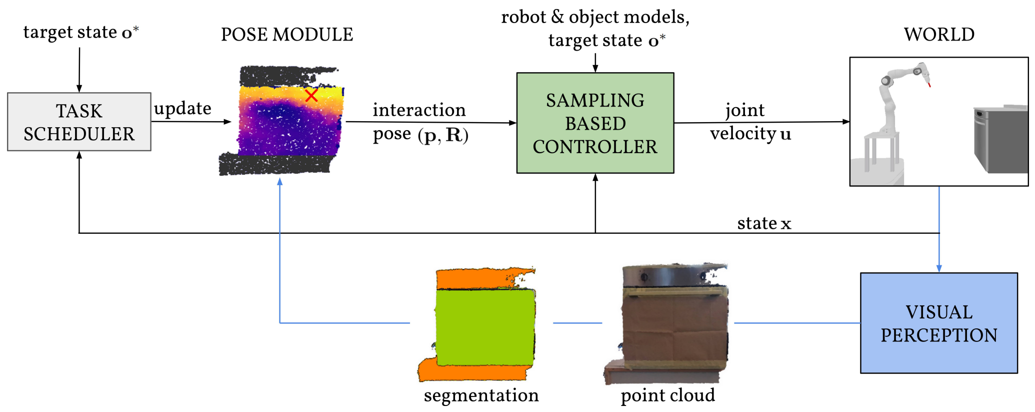
target state $\mathbf{o}^*$
POSE MODULE
robot & object models, target state $\mathbf{o}^*$
WORLD
TASK SCHEDULER
update
interaction pose $(\mathbf{p}, \mathbf{R})$
SAMPLING BASED CONTROLLER
joint velocity $\mathbf{u}$
state $\mathbf{x}$
VISUAL PERCEPTION
segmentation
point cloud

Fig. 2: Block diagram of the control pipeline. The pose module uses affordance estimation to choose the optimal end-effector reference pose. The sampling-based controller generates velocity commands to interact with the object. The task scheduler can trigger an update of the interaction pose if required.

and its target configuration $\mathbf{o}^*$, we define the affordance A of an interaction pose $(\mathbf{p}, \mathbf{R})$ as the expected value of J over all the state-input trajectories induced by the controller $\kappa(\mathbf{x} \mid \mathbf{o}^*, \mathbf{p}, \mathbf{R})$ from initial state $\mathbf{x}_0$. Inferring the optimal interaction pose by maximizing the affordance function A is impractical, as the set of all possible interaction positions and orientations is extremely large. We therefore follow [7], [8] and define two auxiliary functions. The orientation proposal function Q generates end-effector orientation proposals given an interaction point. The actionability function α models the point-wise expected affordance A over a distribution of interaction orientations. At test time, we first select the interaction point that maximizes α. We then generate multiple orientation proposals and score their affordance. Finally, the highest-scoring pose proposal $(\mathbf{p}, \mathbf{R})$ is selected and passed on to the sampling-based controller.

Network architecture: We use a neural network based on [7] to learn the functions required by the optimization problem, namely affordance A, orientation proposal Q and actionability α. The network architecture, shown in Fig. 3, combines a PointNet++ [31] feature encoder with three Multilayer Perceptron (MLP) decoding heads, one for each function. All three networks take as input the point position $\mathbf{p}$, its feature vector $\mathbf{f_p}$ and the target object configuration $\mathbf{o}^*$. The orientation proposal module additionally takes a sampled Gaussian noise vector $\mathbf{z} \in \mathbb{R}^{10} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ as input to generate proposals $\mathbf{R}$ which are then passed on as an input to the affordance prediction module.

Data collection: The training data is collected in a simulation setup where we simulate the complete control pipeline as well as the kinematic chain, collision bodies and dynamics of the target robot platform in order to create agent-aware affordances. A baseline agent samples a random interaction pose $(\mathbf{p}, \mathbf{R})$ on the movable part of the object, where the orientations are sampled only within a 45° cone of the surface normal at point $\mathbf{p}$ (based on [8]). Given the sampled pose and the desired configuration $\mathbf{o}^*$, the sampling-based controller attempts to interact with the object for 10s. A binary reward J rates the interaction as successful if the object configuration was changed by more than 5° towards

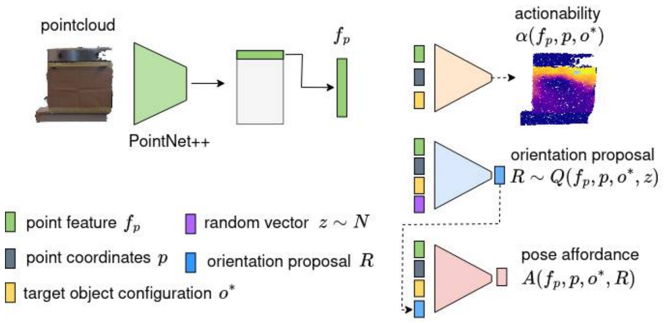
pointcloud
PointNet++
f_p
actionability $\alpha(f_p, p, o^*)$
orientation proposal $R \sim Q(f_p, p, o^*, z)$
pose affordance $A(f_p, p, o^*, R)$
point feature f_p
random vector $z \sim N$
point coordinates p
orientation proposal R
target object configuration o^*

Fig. 3: The pose module, based on [7], combines a PointNet++ [31] feature encoder with three MLP decoding heads, outputting actionability α, orientation proposal Q and pose affordance A.

the target $\mathbf{o}^*$. Finally, we balance the dataset to contain the same number of successful and unsuccessful samples. One advantage of using the sampling-based controller to generate training data is the relatively high rate of successful interactions (13.6%), which makes it much more data efficient than first having to train a control policy (as done in [8]).

Training and losses: Given the point cloud and the interaction sample $(\mathbf{p}, \mathbf{R}, \mathbf{o}^*, J)$, we train the components of the pose module jointly where the affordance module learns to predict the reward realization J using a binary cross-entropy loss, while the orientation proposal module uses the mean cosine similarity loss on quaternion predictions. The actionability module is trained on an L1 loss, where the ground truth actionability for a given pose proposal can be calculated using predictions from the affordance and orientation modules. We use an Adam optimizer and employ learning rate scheduling and early stopping to ensure convergence. Training a network takes around 3 hours on a low-grade GPU (NVIDIA GTX 1050 Ti with 4 GB RAM).

B. Control Pipeline

Sampling-based controller: The sampling-based controller uses a physics engine to forward simulate the system dynamics of the robot and object to iteratively refine the motion trajectory [10]. At every time step, it samples multiple robot joint velocity references $\mathbf{u}$ around the current best guess and simulates the resulting system dynamics for a time horizon of 1s. The best trajectory is then selected based on a cost function, which is made up of multiple components. The object cost encodes the offset from the target $\mathbf{o}^*$, the collision cost punishes robot-object collision, the joint limit cost and the arm reach cost prevent mechanically unreachable configurations. Finally, the pose reach cost is defined as the distance between the end-effector and the reference pose $(\mathbf{p}, \mathbf{R})$, which is required to make the optimization problem feasible. A low-level proportional-integral controller then converts the velocity reference $\mathbf{u}$ to the executed joint torques commands.

Task scheduler: To increase the robustness of the system, we implement a closed-loop affordance inference setup, allowing the robot to change its interaction pose when the current one is no longer favorable. In practice, the task scheduler triggers a pose update whenever the object state cost stagnates. To update the pose, the robot moves to a configuration from where it can observe the full object, records a point cloud, segments it and passes it through the pose module to obtain the new interaction pose.

V. EXPERIMENTS

To evaluate the performance of our pipeline, we conducted three different experiments in simulation and additionally validated our approach in real life. For training, we collect for each task 12000 simulations with eleven different object models from the PartNet-Mobility dataset categories *oven* and *dishwasher*, which both have a revolute joint [32]. First, we compared affordances conditioned only on the end-effector (as in [7], [8]) against affordances conditioned on the full robot. Second, we evaluated the advantage of allowing the agent to update the interaction pose when the current one is no longer advantageous, i.e. split one task into multiple non-continuous motions. These two experiments were performed with seven unseen object models from the training categories since they are especially challenging in terms of reach and collision and are therefore well suited to study the advantages of our method. Finally, we show that our approach is also capable of generalizing to unseen object categories and articulation types. To this end, we tested our pipeline on 140 object models from the categories *safe*, *table* and *washing machine* containing both revolute and prismatic joints and compare it to VAT-Mart [8] as a state-of-the-art benchmark.

A. Agent and Environment

The robot platform consists of a seven degree-of-freedom manipulator mounted on a mobile base. The manipulator hand is equipped with a single, fixed cylindrical finger. One of the advantages of this simple finger design is that it allows the physics engine RAISIM to simulate the contact dynamics, which is a computationally demanding task [33], smoothly in real-time with a simulation timestep of $0.0015\,\mathrm{s}$. To generate the pointclouds, the robot and the object are rendered in the SAPIEN environment [29], where the object's initial position and scale are sampled uniformly at random within predefined, category-specific bounds.

B. Baselines

As a minimum baseline, we employ the same random agent used during training for data collection. To evaluate the importance of agent-aware affordances, we compare against a purely end-effector-aware baseline where we collected the training data with a disembodied, freely moving gripper initialized directly at the interaction pose and thus ignoring the issue of reach. This is similar to the affordance-learning paradigm in Where2Act [7]. As an additional baseline, we enhance the end-effector-aware network with a reachability filter using a Jacobian-based inverse kinematics (IK) check as well as with a robot-object collision filter. These filters are implemented such that each interaction pose proposed by the end-effector-aware network has to pass the feasibility check before being executed. Next, to evaluate the effect of enabling a change of interaction pose, we created for both the end-effector-aware and the agent-aware network one version where the pose stays fixed during the interaction and a closed-loop setup that allows pose updates.

C. Results

Agent-aware vs end-effector-aware: We first evaluate the quality of the proposed agent-aware interaction poses with a simplified version of the pipeline which keeps the interaction pose fixed. As in [7], we report the sample success rate as the percentage of interaction poses creating movement towards the target $\mathbf{o}^*$, i.e. $|\Delta\mathbf{o}| \geq 5^\circ$, as well as the sample reach rate defined as the percentage of pose references that are reachable by the agent. The considered tasks are to *open* or *close* articulated objects from the testing dataset using the full robot model. Over the 500 trials of each task, the initial object configuration $\mathbf{o}_0$ is such that half of the trials start with the object fully closed or fully open (for the *open* and *close* tasks, respectively), the other half start with the object open to an angle sampled randomly over the full joint range.

In the *open* task, both networks learned to interact with a similar set of points (Fig. 4a), leading to similar average performance (Table I). In the *close* task on the other hand, while the end-effector-aware network considers the entire outer surface of the door to be actionable, the agent-aware network strongly prefers interaction points close to the top edge (Fig. 4b). As the end-effector-aware network was trained on a disembodied gripper, it often predicts interaction poses that are unreachable for the full agent due to kinematic or collision constraints. This results in a significantly lower reach and success rate compared to the agent-aware version.

Table I shows that even though the feasibility checks progressively improve the reachability and success rate of poses, the performance is still lower than for our proposed agent-aware network. From this experiment we conclude that there is no simple heuristic filter that can restore the agent-awareness a posteriori. The failure cases of the agent-aware model on the other hand are mostly due to suboptimal actionability predictions of the network, i.e. it proposes poses that can be reached but do not result in successful interactions. Additionally, we compared multiple network instances trained with different random seeds and found that

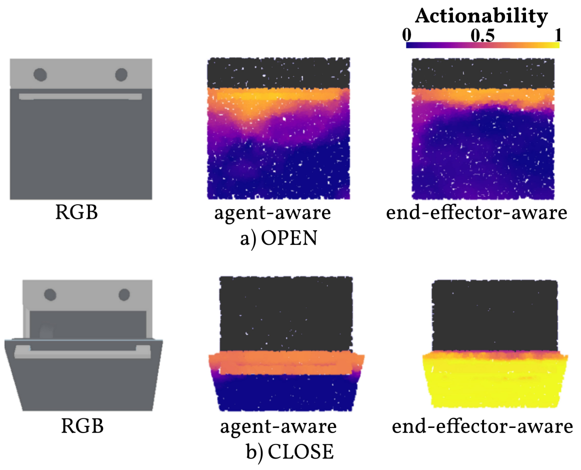

Fig. 4: a) *Open* task: both networks learn to interact with a similar set of points. b) *Close* task: the agent-aware network strongly prefers reachable points while the end-effector-aware network, which was trained on a disembodied gripper, also predicts points on the door surface which are unreachable for the full agent.

OPEN	s. success r.	s. reach r.
Random	15.5%	68.5%
EE-aware	91.9%	99.8%
Agent-aware	**92.1%**	**99.9%**
CLOSE	s. success r.	s. reach r.
Random	21.0%	61.5%
EE-aware	38.5%	54.2%
EE-aware + IK check	51.7%	65.0%
EE-aware + IK + collision check	59.5%	76.6%
Agent-aware	**76.1%**	**92.0%**

TABLE I: Sample success rate and sample reach rate using the full robot model for interaction trials of the *open* and *close* task. We compare our agent-aware network to a random baseline, the end-effector-aware network (EE-aware) and enhanced versions of the EE-aware network combined with an inverse kinematic (IK) and a collision check.

while stochasticity has a visible effect on the affordance map, the resulting performance is consistent across networks. In general, we observe that the sampling-based controller provides a certain robustness against small errors of the predicted interaction pose, as it locally adapts the pose while minimizing the controller cost.

Closed-loop pose update: In this experiment we evaluate the performance on the full opening ($\mathbf{o}_0 = 0°, \mathbf{o}^* = 90°$) and closing ($\mathbf{o}_0 = 90°, \mathbf{o}^* = 0°$) tasks and report the task success rate. Over 1000 trials we compare our closed-loop pipeline, which allows the robot to change its interaction pose during the task, with an ablated fixed pose version. The agent-aware closed-loop pipeline achieves an extremely good task success rate of 88.6% on both *open* and *close* tasks (Table II). We observe two advantages of the closed-loop setup: Firstly, when an initially successful interaction pose becomes infeasible at an intermediate state, the task scheduler triggers an interaction pose update. Secondly, when the sampling-based controller is unable to perform the task at all by interacting with the current reference pose, the agent is able to recover by trying again at a different reference pose. In this simulated setup, we predominantly observed

	OPEN		CLOSE	
	Agent-aware	EE-aware	Agent-aware	EE-aware
Closed-loop	**88.6%**	76.6%	**88.6%**	64.3%
Fixed pose	83.6%	64.7%	69.2%	44.1%

TABLE II: Task success rate of the agent-aware and the end-effector-aware pipeline for the long-horizon open and close tasks. While the closed-loop setup improves the performance for both pipelines, the agent-aware version clearly outperforms its ablation.

the second effect, because the sampling-based controller is able to locally adapt the interaction pose such that the task can be fulfilled in one motion, e.g. by sliding or rotating. Finally, the closed-loop setup also improves the results of the end-effector-aware network, but it is still outperformed by the agent-aware network.

Generalization capabilities: To benchmark against VAT-Mart [8], we evaluated our pipeline on their test objects (*safe*, *table* and *washing machine*) and use their task definition which is to sample the target $\mathbf{o}^*$ and the initial object configuration $\mathbf{o}_0$ randomly over the full joint space. Qualitative results in Fig. 5 show that our pipeline learned interaction strategies that generalize well to unseen object categories despite the limited object variability seen during training. Table III shows that we outperform the VAT-Mart benchmark in all tasks and categories, even when only using a fixed interaction pose. Our pipeline is explicitly designed to profit from available information about the object and the robot which VAT-Mart does not use, namely the collision shape and articulation model as well as the real-time feedback of robot joints and articulation angle which are required by the controller. For our targeted service robotics applications, object CAD models are often readily available [34], while generating novel models would also only require a one-time effort.

	close door	open door	close drawer	open drawer
VAT-Mart [8]	36.5%	14.3%	38.3%	31.1%
Ours (Fixed Pose)	44.0%	49.0%	68.0%	46.5%
Ours (Closed-loop)	**65.8%**	**66.7%**	**80.5%**	**59.1%**

TABLE III: Task success rate, defined according to [8] as within a tolerance of 15% of the task, of our framework on unseen object categories compared to an ablated fixed pose version of our method and the VAT-Mart benchmark.

D. Real-world experiments

We deployed our pipeline on a mobile manipulator interacting with an oven (Fig. 1). Collision bodies and articulation type of the oven are provided to the controller a priori, while the articulation joint angle is obtained by integrating the angular velocity measurement of an IMU mounted on the back of the oven door. The pose of the robotic platform is tracked by an OptiTrack motion capture system and the point cloud data is collected from an onboard Azure Kinect sensor. The reflective glass surface of the oven door was covered to make it visible to the RGB-D camera. For part segmentation, the Iterative Closest Point (ICP) algorithm is used to match the real-world point cloud to the point cloud rendered in simulation, allowing a projection of the segmentation mask.

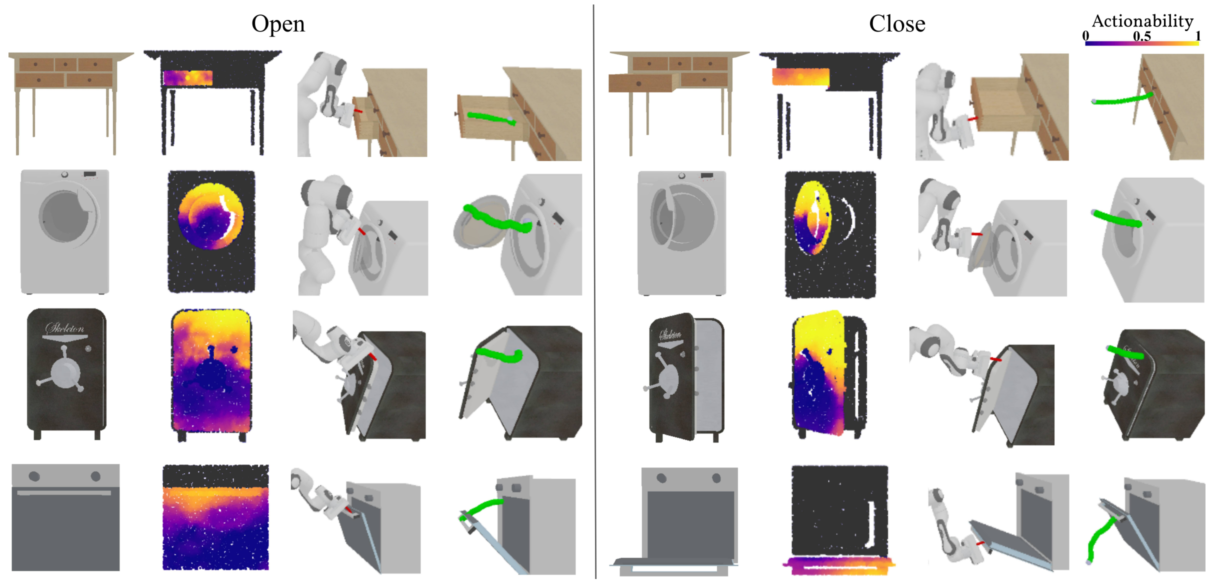

Fig. 5: Qualitative analysis of the interactions with different articulated objects. Each interaction block shows from left to right: The RGB image, the predicted actionability map, the robot interacting with the object and the full trajectory of the end-effector (shown in green).

The closed-loop pipeline was tested over 25 trials of both the *open* ($\mathbf{o}_0 = 0°$, success if $\mathbf{o}_N \geq 75°$) and *close* ($\mathbf{o}_0 \geq 75°$, success if $\mathbf{o}_N \leq 2°$) tasks, where the joint limit of the real oven was at 85°. In Table IV, real-world results are compared to simulation trials with the same setup and testing object. We report the task success rate (*task s. r.*), the average number of interaction attempts (*n. int.*) and the average number of successful motions ($|\Delta\mathbf{o}| \geq 5°$) needed to complete a task (*n. mot.*).

Results: The network is able to produce reasonable actionability maps and interaction poses from real-world point cloud data (Fig. 1 and supplementary video). For the *close* task, the behavior transfers very well from simulation to the real world: The robot is able to complete the task 72% of the time and often in one motion (Table IV). In the *open* task, unlike in simulation, the real robot typically needs multiple attempts to slide the finger into the handle due to errors introduced by the state estimation and model mismatches. Once the end-effector is placed such that it can exert enough force, it opens the door to an intermediate state. This end-effector pose usually becomes unfavorable at a certain articulation angle (as seen in Fig. 1b), such that a pose update is triggered and the robot uses a second and sometimes third motion to fully open the door. The recovery capabilities of the closed-loop pipeline are therefore essential to overcome the described sim-to-real gap, such that we achieve an overall task success rate of 71% for opening.

OPEN	task s. r.	time (s)	n. int.	n. mot.
real	71%	112±41	4.3±1.6	3.0±0.9
sim	100%	50±52	2.0±1.6	1.0±0.2
CLOSE	task s. r.	time (s)	n. int.	n. mot.
real	72%	70±53	2.1±1.6	1.4±0.7
sim	93%	39±29	1.9±1.2	1.2±0.5

TABLE IV: Results from real-world and simulation testing for fully *opening* and *closing* an oven. The average number of interaction attempts (*n. int.*) shows how often the task scheduler triggers a pose update and the number of motions (*n. mot.*) is the subset of interactions that leads to an improvement of the articulation state.

VI. CONCLUSIONS

We introduced a novel closed-loop manipulation pipeline combining point-level affordance inference from visual data with whole-body control. The pipeline is based on sampling-based control, using affordances to estimate the end-effector interaction pose. Current state-of-the-art methods only condition affordance predictions on end-effector geometry. In contrast, we show that our agent-aware network is able to exploit the full robot model and low-level controller to improve the quality of the inferred interaction priors. Our pipeline also enables the agent to re-evaluate the pose model and update the interaction pose at any time during task execution. This allows the agent to split a task into two or more non-continuous motions and recover from failure and unexpected states. We find this to be crucial especially in the real-world experiments, partially compensating for the effects of the sim-to-real gap and allowing our pipeline to perform long-horizon mobile manipulation tasks with high success rates.

Limitations and future work: The sampling-based controller used in our pipeline requires precise measurements of the object joint state, currently provided by an external sensor (IMU). While this is not an issue in simulation and in controlled real-world environments, for general deployment the object pose estimation should be performed using onboard sensors, e.g. from RGB-D data [2], [35]–[38]. The controller also requires a precise object model, which could be estimated from visual data and optionally refined using interaction data [3], [39], [40]. Additionally, in future work the pipeline could be extended to allow for grasping, since not all tasks can be solved through non-prehensile manipulation.

References

[1] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to Generalize Kinematic Models to Novel Objects." *Proc. of the Conference on Robot Learning (CoRL)*, 2019.

[2] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] A. Jain, R. Lioutikov, and S. Niekum, "ScrewNet: Category-Independent Articulation Model Estimation From Depth Images Using Screw Theory," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2021.

[4] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2Motion: Joint Analysis of Motion Parts and Attributes From 3D Shapes," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation," *arXiv*, 2021.

[6] M. Arduengo, C. Torras, and L. Sentis, "A Versatile Framework for Robust and Adaptive Door Operation with a Mobile Manipulator Robot," *arXiv*, 2019.

[7] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2Act: From Pixels to Actions for Articulated 3D Objects," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2021.

[8] R. Wu, Z. Guo, Q. Fan, Y. Zhao, X. Chen, L. Guibas, H. Dong, Y. Wang, T. Wu, and K. Mo, "VAT-Mart: Learning Visual Action Trajectory Proposals for Manipulating 3D ARTiculated Objects," *Proc. of the International Conference on Learning Representations (ICLR)*, 2021.

[9] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2017.

[10] G. Rizzi, J. J. Chung, A. Gawel, M. Tognon, L. Ott, and R. Siegwart, "Stochastic control for reactive whole-body manipulation," 2022, under review, open-source implementation: http://www.github.com/ethz-asl/sampling_based_control/.

[11] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard, "Operating articulated objects based on experience," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[13] R. Martín-Martín and O. Brock, "Coupled recursive estimation for online interactive perception of articulated objects," *The International Journal of Robotics Research*, 2019.

[14] A. Jain and S. Niekum, "Learning Hybrid Object Kinematics for Efficient Hierarchical Planning Under Uncertainty," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[15] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[16] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, "Stein Variational Model Predictive Control," *Proc. of the Conference on Robot Learning (CoRL)*, 2020.

[17] J. J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston MA, U.S.A, 1979.

[18] P. Ardon, E. Pairet, K. S. Lohan, S. Ramamoorthy, and R. P. A. Petrick, "Affordances in Robotic Tasks – A Survey," *arXiv*, 2020.

[19] T. Nagarajan and K. Grauman, "Learning affordance landscapes for interaction exploration in 3d environments," *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Q. Li, K. Mo, Y. Yang, H. Zhao, and L. Guibas, "IFR-Explore: Learning Inter-object Functional Relationships in 3D Indoor Scenes," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2022.

[21] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[22] C. Pohl, K. Hitzler, R. Grimm, A. Zea, U. D. Hanebeck, and T. Asfour, "Affordance-based grasping and manipulation in real world applications," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[23] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, "Learning to See before Learning to Act: Visual Pre-training for Manipulation," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2020.

[24] K. Mo, Y. Qin, F. Xiang, H. Su, and L. J. Guibas, "O2O-Afford: Annotation-Free Large-Scale Object-Object Affordance Learning," *Proc. of the Conference on Robot Learning (CoRL)*, 2021.

[25] E. Ruiz and W. W. Mayol-Cuevas, "Geometric Affordance Perception: Leveraging Deep 3D Saliency With the Interaction Tensor." *Frontiers in Neurorobotics*, 2020.

[26] S. Deng, X. Xu, C. Wu, K. Chen, and K. Jia, "3D AffordanceNet: A Benchmark for Visual Object Affordance Understanding," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[28] Y. Wang, R. Wu, K. Mo, J. Ke, Q. Fan, L. Guibas, and H. Dong, "AdaAfford: Learning to Adapt Manipulation Affordance for 3D Articulated Objects via Few-shot Interactions," *arXiv*, 2021.

[29] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A SimulAted Part-based Interactive ENvironment," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] Z. Xu, Z. He, and S. Song, "UMPNet: Universal manipulation policy network for articulated objects," *IEEE Robotics and Automation Letters*, 2022.

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[32] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] A. M. Castro, A. Qu, N. Kuppuswamy, A. Alspach, and M. Sherman, "A Transition-Aware Method for the Simulation of Compliant Contact with Regularized Friction," *IEEE Robotics and Automation Letters*, 2020.

[34] "IKEA Furniture Models," https://www.ikea.com/ch/en/planners, accessed: 2022-06-15.

[35] J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang, "A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation," *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2021.

[36] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas, "CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds," *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2021.

[37] J. Pavlasek, S. Lewis, K. Desingh, and O. C. Jenkins, "Parts-Based Articulated Object Localization in Clutter Using Belief Propagation," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[38] K. Desingh, S. Lu, A. Opipari, and O. C. Jenkins, "Factored Pose Estimation of Articulated Objects using Efficient Nonparametric Belief Propagation," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2019.

[39] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer, "Visual Identification of Articulated Object Parts," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[40] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building Digital Twins of Articulated Objects from Interaction," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[41] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[42] "Open3D ICP Registration," http://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html, accessed: 2022-06-22.

APPENDIX

A. Controller

a) Sampling-based controller: In this section, we state the equations of our sampling-based controller based on [10]. Let $\mathbf{q} \in \mathbb{R}^{n_q}$ and $\dot{\mathbf{q}} \in \mathbb{R}^{n_q}$ be the configuration of the agent robot and its time derivative, where $n_q \in \mathbb{N}_{>0}$ is the agent's degrees of freedom (DOF) — in our implementation $n_q = 10$. Similarly, let $\mathbf{o} \in \mathbb{R}^{n_o}$ and $\dot{\mathbf{o}} \in \mathbb{R}^{n_o}$ be the configuration of the object and its time derivative, where $n_o \in \mathbb{N}_{>0}$ is the object's DOF — in our implementation $n_o = 1$. The controller state vector is defined as:

$$\mathbf{x} = [\mathbf{q}^T, \dot{\mathbf{q}}^T, \mathbf{o}^T, \dot{\mathbf{o}}^T]^T \in \mathbb{R}^{2(n_q+n_o)}. \tag{1}$$

The controller defines several stage cost function components. The first component is an object cost, which encodes the object manipulation objective:

$$c_{\mathbf{o}}(\mathbf{o}, \mathbf{o}^*; \mathbf{W_o}) = ||\mathbf{o} - \mathbf{o}^*||^2_{\mathbf{W_o}}, \tag{2}$$

where $\mathbf{o}^*$ is the target object configuration and $\mathbf{W_o} \in \mathbb{R}^{n_o \times n_o}$ is a diagonal weight matrix. The second component is the pose reach cost, which encodes the distance of the end-effector to the reference interaction pose $\mathbf{T}^* =$ (position $\mathbf{p}^*$, orientation $\mathbf{R}^*$):

$$c_t(\mathbf{q}, \mathbf{T}^*; \mathbf{W}_t) = || \log(\mathbf{T}(\mathbf{q}) - \mathbf{T}^*)||^2_{\mathbf{W}_t}, \tag{3}$$

where the actual end-effector pose $\mathbf{T}(\mathbf{q})$ is computed via forward kinematics from the joint configuration vector $\mathbf{q}$ and $\mathbf{W}_t \in \mathbb{R}^{6\times 6}$ is a diagonal weight matrix.

A number of surrogate objectives are defined to encode agent limits and constraints:

- Collision cost:

  $$c_c(\mathbf{o}, \mathbf{q}; w_c) = \begin{cases} w_c & \text{if agent-object collision} \\ 0 & \text{otherwise} \end{cases}$$

  where $w_c \in \mathbb{R}$.
- Joint limit cost

  $$\begin{aligned} c_j(\mathbf{q}; w_j, \mathbf{W}_j) =& \mathbb{1}[\mathbf{q} > \mathbf{q}_{\text{upper}}](w_j + ||\mathbf{q} - \mathbf{q}_{\text{upper}}||^2_{\mathbf{W}_j}) + \\ & \mathbb{1}[\mathbf{q} < \mathbf{q}_{\text{lower}}](w_j + ||\mathbf{q} - \mathbf{q}_{\text{lower}}||^2_{\mathbf{W}_j}) \end{aligned}$$

  where $\mathbf{q}_{\text{upper}}$ and $\mathbf{q}_{\text{lower}}$ are the upper and lower joint limits, $w_j \in \mathbb{R}$ and $\mathbf{W}_j \in \mathbb{R}^{n_q \times n_q}$.
- Arm reach cost

  $$c_a(\mathbf{q}; w_a, w_{as}) = \mathbb{1}[r(\mathbf{q}) > r_{\text{max}}](w_a + w_{as}(r(\mathbf{q}) - r_{\text{max}}))$$

  where the current reach r is calculated from forward kinematics from the joint state $\mathbf{q}$ and $w_a, w_{as} \in \mathbb{R}$ are weights.

The controller defines two modes of operation $m \in \{1, 2\}$, corresponding to two stages in object manipulation *reaching* and *interacting*. In the first phase, the end-effector reaches the target interaction pose while avoiding contact with the object. This is achieved by setting the stage cost as:

$$l(\mathbf{x}_i, \mathbf{u}_i; m = 1) = c_c + c_j + c_a + c_t. \tag{4}$$

In the second phase, the agent interacts with the object and moves it towards the target state:

$$l(\mathbf{x}_i, \mathbf{u}_i; m = 2) = c_{\mathbf{o}} + c_j + c_a + c_t. \tag{5}$$

b) Low-level controller: We use a dynamically compensated low-level proportional-integral controller to convert the velocity references $\mathbf{u}$ from the sampling-based controller to joint torque commands τ. Let $\mathbf{q}^*$ be the desired agent configuration, calculated by integrating the velocity references over the time interval dt:

$$\mathbf{q}^* = \mathbf{q} + \mathbf{u}dt.$$

Let $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_q \times n_q}$ be the matrix of inertia of the agent. The Coriolis and gravity terms of the system dynamics are denoted as $\mathbf{Co}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{g}(\mathbf{q})$, respectively. The joint torque vector is calculated as:

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}^* + \mathbf{Co}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}^* + \mathbf{g}(\mathbf{q}) - \mathbf{K}_D\dot{\tilde{\mathbf{q}}} - \mathbf{K}_I \int_0^t \dot{\tilde{\mathbf{q}}} d\tau, \tag{6}$$

with the auxiliary error variable $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}^*$ and $\mathbf{K}_D$ and $\mathbf{K}_I$ being positive diagonal gain matrices.

c) Task scheduler: The task scheduler selects the appropriate mode of the sampling-based controller, which can either be *reaching* ($m = 1$) or *interacting* ($m = 2$). It also regulates updates in the interaction pose $\mathbf{T}$. Given the current state observation $\hat{\mathbf{x}}$, the target object configuration $\mathbf{o}^*$ and the current pose reference $\mathbf{T}$, the task scheduler implements the following simple heuristic:

- At the beginning of the interaction, update the reference pose and go into *reaching* mode ($m = 1$).
- During the interaction:
  - If in *reaching* mode and the current reference pose was reached by the end-effector, change into *interacting* mode ($m = 2$).
  - If in *interacting* mode and the object configuration did not improve for a set number of seconds (5s in simulation, 1s in real-life testing), lift the robot arm to prevent camera occlusion while recording a new point cloud, trigger an update of the reference pose and return to *reaching* mode.
  - If $|\mathbf{o} - \mathbf{o}^*| \leq 5°$, stop the interaction as the task is fulfilled.

The task scheduling described above allows the agent to change the interaction pose when the current one is not advantageous anymore as well as to try again if the current interaction pose failed to lead to any improvement in the object configuration. As both the sampling-based controller and the pose module are non-deterministic and the agent moves in between the pose updates, the same initial conditions of task and object might result in different behavior, i.e. success where before the agent failed.

B. Affordance Definitions

This section gives an overview of the equations used for the pose module. Let $J(\mathbf{x}_{0:N}, \mathbf{u}_{0:N-1}; \mathbf{o}^*)$ be a reward function that, given the the target object configuration $\mathbf{o}^*$, maps a trajectory of system states $\{\mathbf{x}_{0:N}\}$ and controller inputs $\{\mathbf{u}_{0:N-1}\}$ to a binary reward value:

$$J(\mathbf{x}_{0:N}, \mathbf{u}_{0:N-1}; \mathbf{o}^*) = \begin{cases} 1 & \text{if} \quad |\mathbf{o}_0 - \mathbf{o}^*| - |\mathbf{o}_N - \mathbf{o}^*| > \theta \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $\theta \in \mathbb{R}_{>0}$ is a success threshold (in our implementation $\theta = 5°$).

Given an object, a target configuration $\mathbf{o}^*$ and an initial state $\mathbf{x}_0$, we define the affordance A_J of an interaction pose $(\mathbf{p}, \mathbf{R})$ as the expected value of J over all the state-input trajectories induced by the controller $\kappa(\mathbf{x}; \mathbf{o}^*, \mathbf{p}, \mathbf{R})$ on the agent-object system from initial state $\mathbf{x}_0$. In our method, the object and the initial state are implicitly represented by the point cloud which is passed through the PointNet++ encoder [31] to generate point-wise feature vectors $\mathbf{f}_\mathbf{p}(\mathbf{p}, \mathbf{x}_0, \text{object})$. Therefore we use the following definition of affordance A_J:

$$A_J(\mathbf{p}, \mathbf{R}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*) := \mathop{\mathbb{E}}_{\mathbf{x}_i, \kappa(\mathbf{x}_i)} \left(J\left(\mathbf{x}_{0:N}, \kappa(\mathbf{x}_{0:N-1}); \mathbf{o}^*\right)\right). \tag{8}$$

Following this affordance definition, the optimal interaction pose $(\mathbf{p}^*, \mathbf{R}^*)$ is given by maximizing the affordance function A_J:

$$\mathbf{p}^*, \mathbf{R}^* = \arg\max_{\mathbf{p}, \mathbf{R}} A_J(\mathbf{p}, \mathbf{R}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*). \tag{9}$$

Solving this equation for all possible interaction poses is computationally intractable, therefore we define auxiliary functions to optimize the position and orientation in a hierarchical manner. The orientation proposal distribution Q_J generates orientation samples $\mathbf{R} \approx \mathbf{R}^*$ that approximate the optimum for a given interaction point $\mathbf{p}$:

$$\mathbf{R} \sim Q_J(\mathbf{p}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*).$$

The actionability function α_J is defined as the point-wise expected value of the affordance over interaction orientations sampled from an orientation proposal distribution Q_J:

$$\alpha_J(\mathbf{p}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*) = \mathop{\mathbb{E}}_{\mathbf{R} \sim Q} A_J(\mathbf{p}, \mathbf{R}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*). \tag{10}$$

This allows us to first obtain the interaction point $\hat{\mathbf{p}}$ by sampling the actionability:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} \alpha_J(\mathbf{p}, \mathbf{f}_\mathbf{p}, \mathbf{o}^*). \tag{11}$$

Next, a set of orientation proposals are sampled from Q_J. Finally the affordance function A_J is used to score the orientation proposals and sample the highest-scoring one:

$$\hat{\mathbf{R}} = \arg\max_{\mathbf{R} \sim Q} A_J(\hat{\mathbf{p}}, \mathbf{R}, \mathbf{f}_{\hat{\mathbf{p}}}, \mathbf{o}^*). \tag{12}$$

This hierarchical procedure does not necessarily output the globally optimal interaction pose. Instead, the quality of the solution heavily depends on the orientation sampling function Q_J. If $\mathbf{R} \sim Q$ only and always samples the optimal interaction orientation, the procedure will yield the global pose optimum.

C. *Simulation Settings and Training*

a) Simulation settings: Dynamics are simulated using a physics simulation based on the commercial software RAISIM. We set the simulation timestep to 0.0015 s. The range of motion of the object articulation joint is set to $90°$. To increase simulation speed, we do not check for collisions between the two object links. We set a damping coefficient of 20 and friction of 40 at the object articulation joint. Between surfaces of contact bodies, we set a low friction coefficient (0.01).

b) Training data collection: Each object in the PartNet Mobility dataset is scale-normalized within a unit sphere. For each simulation instance, we sample an object from the category *oven* or *dishwasher* and apply a scale factor $\in [45\%, 55\%]$ of the normalized scale. This ensures a realistic relative scale between our object and agent models, and serves as augmentation. The position of the object, measured from the world origin to the center of the object, is sampled uniformly between the following bounds ($x = 0$ m$, y \in [-0.3$ m$, 0.3$ m$], z \in [0.75$ m$, 0.95$ m$]$). We initialize the object articulation joint at rest 50% of the time (i.e. $0°$ for the *open* task, $90°$ for the *close* task) and in an intermediate state for the other 50% of initializations. We sample the initial and target configuration within the articulation bounds and according to the task, with a minimum task distance $\Delta\mathbf{o} = |\mathbf{o}^* - \mathbf{o}_0|$ of $20°$.

The agent is initialized facing the world origin, with a uniformly sampled distance $\in [1.75$ m$, 3.0$ m$]$ and base rotation angle $\arctan(y, x) \in [-20°, 20°]$. During the simulated interaction, the agent is given 30s to move to the pose reference (i.e. mode $m = 1$ of the sampling-based controller) and 5s to interact with the object (i.e. $m = 2$). A simulation is considered successful if the object configuration was moved by at least $5°$ towards the task. Figure 6 shows our simulation environment, complete with axes for reference. We collect 12000 simulations for each task, which takes around 15 hours per task on a commercial CPU (Intel Core i7-7700HQ quad-core processor).

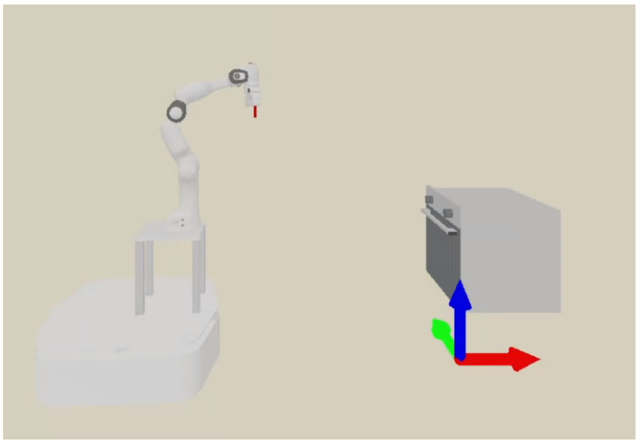

Fig. 6: Rendered simulation environment with world axes (x: red, y: green, z: blue) for reference.

c) Network training: We use an Adam optimizer with an initial learning rate of $1\text{e}{-4}$ and batch size of 10. To ensure convergence, we employ both learning rate scheduling (patience of 5 epochs, discount factor 0.2, minimum improvement of 2%, minimum learning rate of $1\mathrm{e}{-6}$) and early stopping (patience of 12 epochs, minimum improvement of 5%). Fully training a network usually requires between 30 and 50 epochs, and takes around 3 hours on a low-grade GPU (NVIDIA GTX 1050 Ti, with 4 GB of dedicated RAM).

d) Evaluation setup: For evaluation, the simulation is initialized in the same manner as during training data collection. In the first experiment of the paper, the agent is given 40s to move to the pose reference and 30s to interact with the object. The sample success rate is evaluated, which is the percentage of successful pose references and success is defined as a movement of the object configuration $\mathbf{o}$ by at least 5° towards the target $\mathbf{o}^*$:

$$\text{sample success rate} = \frac{\text{n successful proposals}}{\text{n proposals}}. \tag{13}$$

We also report the sample reach rate, which is the percentage of pose references that are reachable by the agent if given as a target pose to the sampling-based controller:

$$\text{sample reach rate} = \frac{\text{n reached proposals}}{\text{n proposals}}. \tag{14}$$

In the second experiment, which evaluates the effect of the closed-loop dynamic pose update, the agent is given 40s to move to the pose reference and 300s to interact with the object. We report the task success rate, where a task is considered successful if the final object configuration $\mathbf{o}$ is within 5° of the target $\mathbf{o}^*$:

$$\text{task success rate} = \frac{\text{n successful tasks}}{\text{n tasks}}. \tag{15}$$

In the third experiment, the pipeline is tested on objects of the category *safe*, *washing machine* and *table*, applying a scale factor of 50%, 50% and 100% respectively to ensure a realistic proportion between agent and object. Following VAT-Mart [8], we first uniformly sample an object category, and then sample a shape from this category. We sample a task $\Delta\mathbf{o} = |\mathbf{o}^* - \mathbf{o}_0|$ in a range of $[10^\circ, 70^\circ]$ for revolute joints and $[0.1, 0.7]$ for prismatic joints. The initial object configuration $\mathbf{o}_0$ is randomly sampled such that it lies within the articulation joint bounds given the task. We report the task success rate, where we define a successful articulation joint value equivalently to VAT-Mart within a tolerance of 15% of the task $\Delta\mathbf{o} = |\mathbf{o}^* - \mathbf{o}_0|$.

D. Network Sensitivity Study

We identify three sources of randomness in the results of our interaction trials. Firstly, the random seed used for initialization of the artificial neural networks causes variability in the output of the network. Secondly, in our testing protocol the initial state and target object configuration are sampled at random. The third source of randomness lies in the stochastic nature of the sampling-based controller. To analyse these effects, we use the training data from the first experiment to train multiple networks initialized with different random seeds. The results are evaluated on 500 interaction trials for each task and model (end-effector-aware and agent-aware). Figure 7 reports the sample success rate of the two models for the *open* task across 5 training runs and for the *close* across 8 training runs. We observe that for the *open* task, the performance of the networks is repeatable. In the *close* task on the other hand, the end-effector-aware networks are far more inconsistent. This could be caused by the fact that behavior learned during training with a disembodied gripper doesn't transfer consistently to the testing task with the full agent. Overall, this analysis shows that the performance of the agent-aware pipeline is consistent over multiple training runs, and that therefore the influence of the stochasticity in controller and neural network is minimal.

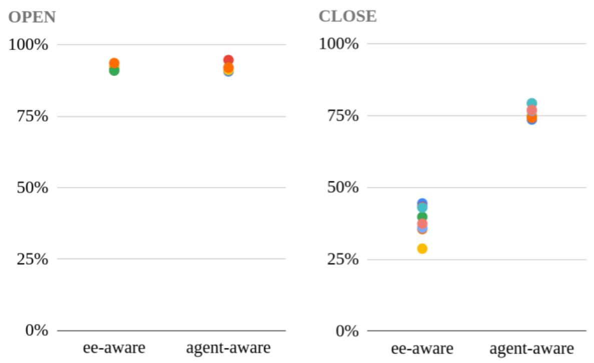

Fig. 7: Sample success rate of the end-effector-aware and the agent-aware networks for the *open* task across 5 training runs (left) and for the *close* task (right) across 8 training runs with different random seeds (indicated by the colors).

We analyzed the predictions of networks trained with different random seeds and observed that while the test-time performance is consistent across different networks, the spread of the predicted actionabilities varies (c.f. Fig. 8). This suggests that the networks converge to different local minima, which could be caused by the fact that our training data is relatively sparse and has a stochastic component due to the controller. Additionally, it should be noted that at test time only the points with the highest actionability are sampled, therefore the distribution of the intermediate-level actionability scores does not have a direct influence on the executed interaction.

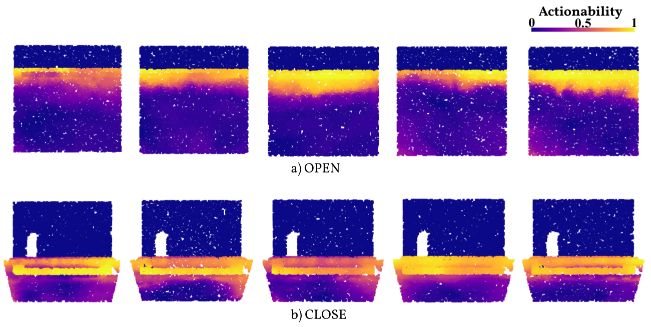

Fig. 8: The predicted actionability scores for networks trained with different random seeds on (a) the *open* and (b) the *close* task.

E. Hardware experiments

The hardware experiments were conducted with an omni-directional platform (Clearpath Ridgeback) with a 7 degree-of-freedom robotic arm (Franka Emika Panda) mounted on top. The selected target object is an oven which was placed on a workbench, as seen in Fig. 9. As it is required by our control pipeline, we build an articulation and collision model of the oven from real-world measurements.

The sampling-based controller is run on an onboard computer (Intel Core i7-8550U quad-core processor) at a frequency of 66Hz, and a proportional-integral velocity controller computes joint torque commands for the arm at 1000Hz. The Ridgeback base is controlled directly in velocity-space at a frequency of 50Hz. The task scheduler is run at 30Hz. The pose module is only run when a pose update is triggered, where each pose update requires roughly 1s of computational time. The pose module and task scheduler are run on a separate laptop (Intel Core i7-7700HQ quad-core processor). The two devices communicate wirelessly over the network using the ROS framework.

The sampling-based controller requires an accurate state estimation. The pose of the Ridgeback mobile base is tracked using an OptiTrack motion capture system, while for base velocity we use odometry data. At startup, we calibrate the oven position in the global reference frame. To obtain the articulation joint angle, filtered linear acceleration and angular velocity are continuously extracted from an inertial measurement unit (IMU) mounted on the back of the oven door. The joint velocity is estimated by projecting the angular velocity measurements onto the articulation rotation axis, while joint position is estimated by integrating velocity.

Point cloud data is collected by an Azure Kinect sensor mounted on the robot base (Figure 9). While state-of-the-art object segmentation, e.g. a neural network such as Mask R-CNN [41], could be implemented for segmentation, we used a simple heuristic which exploits the knowledge about the position and size of the object. The segmentation of the object from the scene is implemented using bounding boxes. For part segmentation, the Iterative Closest Point (ICP) algorithm is used to closely match the real-world point cloud to an expected point cloud obtained from our simulator/rendering setup [42]. Once the two point clouds are matched, the simulated point cloud is converted to a rough voxel map with a voxel size of 50mm and the real-world point cloud is segmented by checking if its points are contained in the voxel map. Example images from this process can be found in Figure 10.

While the real-world experiment shows that our proposed system performs well on the oven, we would also like to showcase its generalization capabilities. Fig. 11 depicts the point cloud of multiple articulated objects with the corresponding affordance prediction. It shows that despite the partial and noisy point cloud of unseen objects, the affordance prediction module creates reasonable interaction scores.

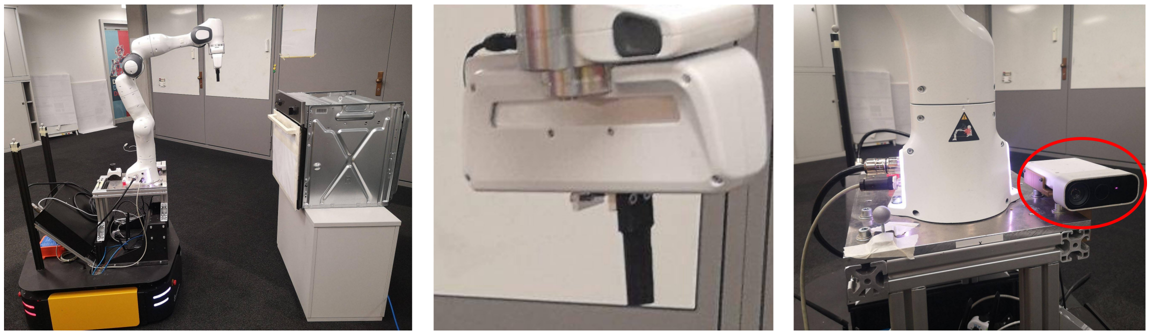
Fig. 9: Real-world setup showing the robotic platform and target object (left), the end-effector consisting of a single fixed finger (center) and the placement of the RGB-D Camera (right).

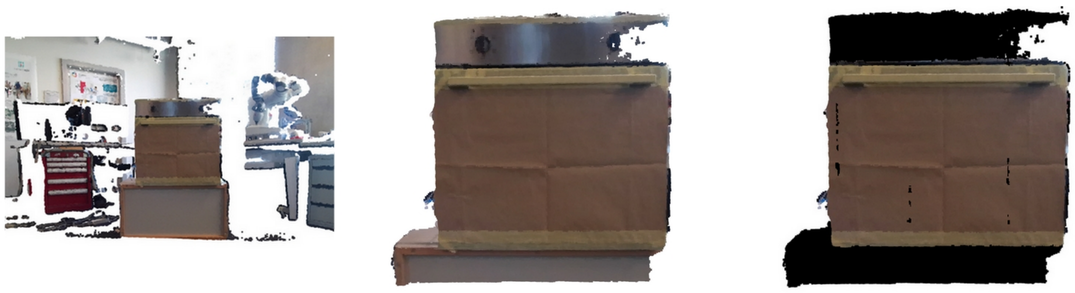
Fig. 10: Segmentation process: scene point cloud (left), object point cloud (center) and part segmentation mask (right).

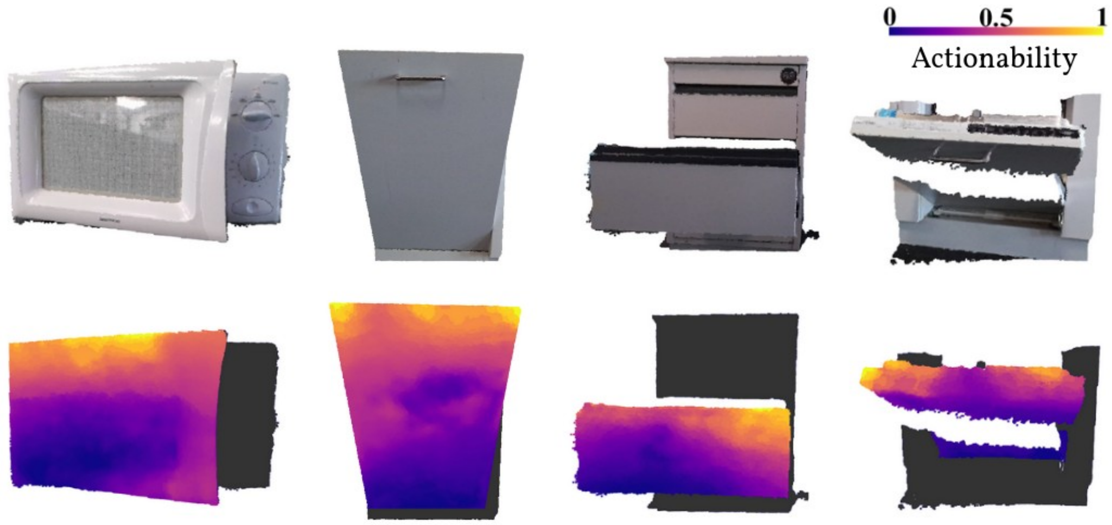

Fig. 11: Real-world point-clouds of unseen articulated objects (top row) and the respective predicted actionability scores (bottom row).