FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects

Ben Eisner*, Harry Zhang*, David Held
Carnegie Mellon University
Pittsburgh, PA, USA
{baeisner, haolunz, dheld}@andrew.cmu.edu

***Abstract*—We explore a novel method to perceive and manipulate 3D articulated objects that generalizes to enable a robot to articulate unseen classes of objects. We propose a vision-based system that learns to predict the potential motions of the parts of a variety of articulated objects to guide downstream motion planning of the system to articulate the objects. To predict the object motions, we train a neural network to output a dense vector field representing the point-wise motion direction of the points in the point cloud under articulation. We then deploy an analytical motion planner based on this vector field to achieve a policy that yields maximum articulation. We train a single vision model entirely in simulation across all categories of objects, and we demonstrate the capability of our system to generalize to unseen object instances and novel categories in both simulation and the real world using the trained model for all categories, deploying our policy on a Sawyer robot with no finetuning. Results show that our system achieves state-of-the-art performance in both simulated and real-world experiments. Code, data, and supplementary materials are available here.**

I. Introduction

Understanding and being able to manipulate articulated objects such as doors and drawers is a key skill for robots operating in human environments. While humans can rapidly adapt to novel articulated objects, constructing robotic manipulation agents that can generalize in the same way poses significant challenges, since the complex structure of such objects requires three-dimensional reasoning of their parts and functionality. Due to the large number of categories of such objects and intra-class variations of the objects' structure and kinematics, it is difficult to train efficient perception and manipulation systems that can generalize to those variations.

To address these challenges, we propose to separate this problem into one of "affordance learning" and "motion planning." If a robot can predict the potential movements of an objects' parts (a.k.a. "affordances"), it would be relatively easy for the agent to derive a downstream manipulation policy by following the predicted motion direction. Thus, we tackle the problem of manipulating articulated objects by learning to predict the motion of individual parts on articulated objects.

Previous work has proposed to learn the articulation parameters (i.e. rotation axis of revolute joints and translation axis of prismatic joints) in order to guide the manipulation policy [5]. However, such methods often rely on knowing class-specific articulation structures. Without such knowledge, the policies can neither operate nor be applied to novel categories.

*Equal contribution.

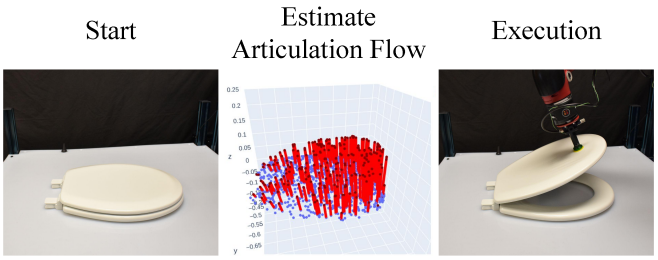


Fig. 1: FlowBot3D in action. The system first observes the initial configuration of the object of interest, estimates the per-point articulation flow of the point cloud (3DAF), then executes the action based on the selected flow vector. Here, the red vectors represent the direction of flow of each point (object points appear in blue); the magnitude of the vector corresponds to the relative magnitude of the motion that point experiences as the object articulates.

To learn a generalizable perception and manipulation pipeline, we need to be robust to the variations of the articulated objects' geometries and kinematic structures. We seek to construct a vision system that can learn to predict how the parts move under kinematic constraints without explicitly knowing the articulation parameters: specifically, the location of the rotational or translational axes for revolute or prismatic parts, respectively.

In this paper, we present FlowBot3D, a deep 3D vision-based robotic system that predicts dense per-point motion of an articulated object in 3D space, and leverages this prediction to produce actions that articulate the object. We define such per-point motion as the **3D articulation flow (3DAF)** vectors, since this representation describes how each observed point on the articulated part would "flow" in the 3D space under articulation motion. Such a dense vector field prediction can then be used to aid downstream manipulation tasks for both grasp point selection as well as predicting the desired robot motion after grasping. We train a ***single*** 3D perception module to perform this task across many object categories, and show that the trained model generalizes to a wide variety of objects – both in seen categories, and entirely unseen object categories.

The contributions of this paper include:

1) A novel per-point representation of the articulation structure of an object, 3D Articulation Flow (3DAF).
2) A novel 3D vision neural network architecture (which we call ArtFlowNet) that takes as input a static 3D point

cloud and predicts the 3D Articulation Flow of the input point cloud under articulation motion.
3) A novel robot manipulation system (FlowBot3D) for using the predicted 3D Articulation Flow to manipulate articulated objects.
4) Simulated experiments to test the performance of our system in articulating a wide range of PartNet-Mobility dataset objects.
5) Real-world experiments deployed on a Sawyer robot to test the generalizablity and feasibility of our system in real-world scenarios.

II. Related Work

Articulated Object Manipulation: Manipulation of articulated objects and other objects with non-rigid properties remains an open research area due to the objects' complex geometries and kinematics. Previous work proposed manipulating such objects by hand-designed analytical methods, such as the immobilization of a chain of hinged objects by Cheong et al. [5]. Berenson et al. [3] proposed a planning framework for manipulation under kinematic constraints. Katz et al. [14] proposed a method to learn such manipulation policies in the real-world using a grounded relational representation learned through interaction.

With the development of larger-scale datasets of articulated objects such as the PartNet dataset by Mo et al. [18] and Partnet-Mobility by Xiang et al. [28], several works have proposed learning methods based on large-scale simulation and supervised visual learning. Mo et al. [19] proposed to learn articulation manipulation policies through large-scale simulation and visual affordance learning. Xu et al. [29] proposed a system that learns articulation affordances as well as an action scoring module, which can be used to articulate objects. Mu et al. [20] provided a variety of baselines for the manipulation tasks of 4 categories of articulated objects in simulation. Several works have focused specifically on visual recognition and estimation of articulation parameters, learning to predict the pose [31, 30, 26, 11, 15] and identify articulation parameters [13, 32] to obtain action trajectories. Moreover, [21, 4, 6] tackle the problem using statistical motion planning.

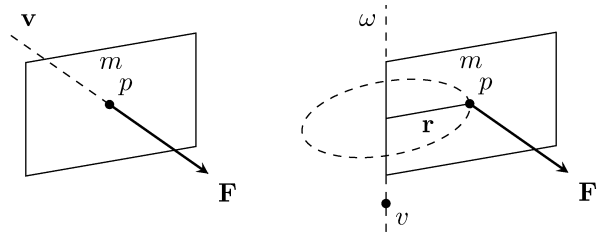
Optical Flow for Policy Learning: Optical flows [10] are used to estimate per-pixel correspondences between two images for object tracking and motion prediction and estimation. Current state-of-the-art methods for optical flow estimation leverage convolutional neural networks [9, 12, 25]. Dong et al. [8], Amiranashvili et al. [1] use optical flow as an input representation to capture object motion for downstream manipulation tasks. Weng et al. [27] uses flow to learn a policy for fabric manipulation. While the aforementioned optical flows are useful for robotic tasks, we would like to generalize the idea of optical flow beyond pixel space into full three-dimensional space. Instead, we introduce "3D Articulation Flow", which describes per-point correspondence between two point clouds of the same object. Another work that is highly related to ours is Pillai et al. [22], which learns to predict the articulated objects' parts motion using a motion manifold learner. First, while we both predict the parts' motion to derive an implicit policy, we do not rely on the intermediate articulation parameters in order to predict the motion manifold. Second, we do not rely on any demonstration to learn from - our method learns in a completely self-supervised fashion.

III. Method - From Theory to Practice

In this section, we examine the physical task of manipulating the articulation of an articulated object. We first present the theoretical grounding behind the intuition of our method, and we slowly relax assumptions and approximations to create a system that articulates objects in the real world based on point cloud observations.

A. An Idealized Policy Based On Dynamics and Kinematics

The articulated objects we consider in this work are generally objects that 1) consist of one or more rigid-bodies – or "links" – which are 2) connected to one another by revolute or prismatic joints with exactly 1 degree of freedom each, and 3) have at least one link rigidly attached to an immovable world frame so that the only motion the object experiences is due to articulation. Each joint connects a *parent link* (often the fixed-world link) and a *child link*, which can move freely subject to the articulation constraints. While these conditions may seem restrictive, under normal "everyday" forces many real-world articulated objects (ovens, boxes, drawers, etc.) meet these conditions to a very good approximation.[1]



(a) Prismatic (b) Revolute

Fig. 2: Illustrations of prismatic and revolute joints.

We now consider an idealized policy to actuate an articulated object. Suppose we are able to attach a gripper to any point $p \in \mathcal{P}$ on the surface $\mathcal{P} \subset \mathbb{R}^3$ of a child link with mass m. At this point, the policy can apply a 3D force $\mathbf{F}$, with constant magnitude $||\mathbf{F}|| = C$ to the object at that point. Our objective is to choose a contact point and force direction $(p^*, \mathbf{F}^*)$ that maximizes the acceleration $\mathbf{a}$ of the articulation's child link. If we limit our analysis to two special classes of articulation, revolute joints and prismatic joints, we can very intuitively arrive at the following optimal settings of $(p^*, \mathbf{F}^*)$:

Prismatic: A prismatic joint (such as a drawer) can be described as a single 3D unit vector $\mathbf{v}$ which is parallel to its direction of motion. Since motion of the joint is constrained to $\mathbf{v}$, the object will provide a responding force $\mathbf{F}_n$ to any component of $\mathbf{F}$ not parallel to $\mathbf{v}$. The net force exerted on

[1]We therefore exclude objects with socket joints, free-body objects, and deformable objects from our analysis.

the joint by the robot is thus $\mathbf{F}_{\text{net}}$, the component of $\mathbf{F}$ in the $\mathbf{v}$ direction:

$$\begin{aligned}\mathbf{F}_{\text{net}} &= \mathbf{F} - \mathbf{F}_n \\ &= \mathbf{F} - (\mathbf{F} - (\mathbf{F}\cdot\mathbf{v})\mathbf{v}) = (\mathbf{F}\cdot\mathbf{v})\mathbf{v} = \mathrm{m}\mathbf{a} \end{aligned} \tag{1}$$

As one might expect, the force vector $\mathbf{F}^*$ which maximizes the acceleration $\mathbf{a}$ occurs when $||\mathbf{F}^* \cdot \mathbf{v}|| = C$, i.e. when $\mathbf{F}^*$ is parallel to $\mathbf{v}$. Because each point $p \in \mathcal{P}$ moves in parallel, applying the force at any point p on the surface will yield the maximum acceleration. Thus, the optimal policy to articulate a prismatic joint is to select any point on the surface and apply a force parallel to $\mathbf{v}$ at every time step.

Revolute: A revolute joint (such as a door hinge) can be parameterized by a pair $(v, \boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a unit vector representing the direction of the axis of rotation about which the child link moves, and $v \in \mathbb{R}^3$ is a point in 3D space that the axis of rotation passes through. Each point p on the child link is constrained to move on the 2D circle perpendicular to the axis of rotation with radius $\mathbf{r}$ (where $||\mathbf{r}||$ is the length of the shortest vector from p to the line given by $f(t) = v + t\boldsymbol{\omega}$). Given any point p, we can maximize the acceleration by a similar argument as before, except any force in the direction of $\mathbf{r}$ or $\boldsymbol{\omega}$ will be resisted:

$$\mathbf{F}_{\text{net}} = \mathbf{F} - \mathbf{F}_n = \mathbf{F} - \left(\frac{\mathbf{F}\cdot\mathbf{r}}{||\mathbf{r}||^2}\right)\mathbf{r} - (\mathbf{F}\cdot\boldsymbol{\omega})\boldsymbol{\omega} \tag{2}$$

Thus, for any point p the net force (and thus acceleration) is maximized when $\mathbf{F}^*$ is tangent to the circle defined by $\mathbf{r}$. Selecting the point p which produces the maximal linear acceleration when $\mathbf{F}^*$ is applied there is simply the point p on the child link that maximizes $||\mathbf{r}||$, or the point on the object farthest from the axis of rotation $\boldsymbol{\omega}$. Thus, the optimal policy to articulate a revolute joint is to pick the point on the surface farthest from the axis of rotation $\boldsymbol{\omega}$ and apply a force parallel to $\mathbf{r} \times \boldsymbol{\omega}$ at every time step.

B. Articulation Parameters to 3D Articulation Flow

These parameterizations[2] are an elegant representation of single articulations in isolation. However, when an object contains more than one articulation, or contains points that do not move at all (e.g. the base of a cabinet), in order to create a minimal parametric representation of the object we must describe a kinematic tree (a tree of rigid links, connected by joints described by a set of parameters) and associate each point on the object with a link. This is a hierarchical representation, which is difficult to construct from raw observation without prior knowledge of the hierarchical structure or link membership. A hierarchy-free representation of the kinematic properties of the object could assign each point on the object its own set of parameters; however, this would require a full 6 parameters (v, ω) for each point on the object, and the position v can occur anywhere in $\mathbb{R}^3$ depending on the object's coordinate frame. A more compact, bounded, hierarchy-free representation is the **3D articulation flow (3DAF)** that each point on the object would experience were its part articulated in the positive direction with respect to its articulation parameters. In other words, for each point on each link on the object, define a vector in the direction of motion of that point caused by an infinitesimal displacement $\delta\theta$ of the joint, and normalize it by the largest such displacement on the link. Thus, the 3D articulation flow f_p for point $p \in \mathcal{P}_i$ in link i is:

$$f_p = \begin{cases} \mathbf{v}, & \text{if } i \text{ is a prismatic joint} \\ \frac{\omega \times \mathbf{r}}{||\omega \times \mathbf{r}_{\max}||} & \text{if } i \text{ is a revolute joint} \end{cases} \tag{3}$$

where $\mathbf{v}, \omega$, and $\mathbf{r}$ are defined above; note that $\mathbf{v}$ is already a unit vector. We denote the full set of flow vectors for an object as $F = \{f_p\}_{p\in\mathcal{P}}$ where $\mathcal{P} = \cup_i \mathcal{P}_i$.

While this representation is mathematically equivalent to both the hierarchical and point-wise parameter-based representations, 3D articulation flow has several key advantages over parameter-based representations:

1) It is hierarchy-free, meaning that it can be easily approximated without an explicit model (i.e. kinematic structure); this property will allow our learned method to generalize to novel object categories.
2) Each element in the representation is a scaled orientation vector constrained to lie inside the unit sphere in $\mathbb{R}^3$. This means that the representation is invariant under translation and scaling in the coordinate frame of the underlying object.

Since this representation is defined for any arbitrary point in or on an object, it could be applied to any discrete or continuous geometric representation of said object. However, for the purposes of this work, we apply this representation to 3D point clouds produced from depth images. Thus for a pointcloud $P = \{p_k \in \mathbb{R}^3\}_{k\in[n]}$, we associate each point p_k in P with a flow vector $f_k \in \mathbb{R}^3, \text{s.t.} ||f_i|| \leq 1$.

This formulation of 3D articulation flow is similar in spirit to the intermediate representation proposed by Zeng et al. [32] in their articulation estimation system, FormNet. However, our representation differs in two key ways. First, our representation describes the instantaneous motion of a link, whereas the FormNet formulation predicts the current absolute displacement of a part from a reference position (i.e. a fully-closed door). Second, we demonstrate that our formulation can be used directly by a manipulation policy, whereas the downstream task of FormNet's representation was predicting the articulation parameters of an object.

C. Predicting 3D Articulation Flow from Vision

We now turn to the question of estimating 3D Articulation Flow from a robot's sensor observations. We consider a single articulated object in isolation; let $s_0 \in \mathcal{S}$ be the starting configuration of the scene with a single articulated object where $\mathcal{S}$ is the configuration space. We assume that the robot has a depth camera and records point cloud observations

[2]An astute reader may recognize that these parameterizations are special cases of twists from Screw Theory. Without loss of accuracy, we choose to omit a rigorous screw-theoretic treatment of articulated objects in favor of an explanation that requires only basic knowledge of physics.

$O_t \in \mathbb{R}^{3\times N}$, where N is the total number of observable points from the sensor. The task is for the robot to articulate a specified part through its entire range of motion.

For each configuration s_t of the object, there exists a unique ground-truth flow F_t, where the ground-truth flow of each point is given by Equation 3. Thus, we would like to find a function $f_\theta(O_t)$ that predicts the 3D articulation flow directly from point cloud observations. We define the objective of minimizing the L2 error of the predicted flow:

$$\mathcal{L}_{\text{MSE}} = \sum_i ||F_{t,i} - f_\theta(O_t)_i||_2 \tag{4}$$

where i indexes over the objects in the training dataset. While f_θ can be any estimator, we choose to use a neural network, which can be trained via a standard supervised learning with this loss function.

D. A General Policy using 3D Articulation Flow

Algorithm 1 The FlowBot3D articulation manipulation policy

Require: $\theta \leftarrow$ parameters of a trained flow prediction network

$(O_0) \leftarrow$ Initial observation
$\hat{F}_0 \leftarrow f_\theta(O_0, [M_0])$, Predict the initial flow
$g_0 = \texttt{SelectContact}(O_0, \hat{F}_0)$, Select a contact pose.
$in\_contact \leftarrow$ False
while not $in\_contact$ **do**
  Drive an end effector towards g_0
  if `DetectContact()` **then**
    $in\_contact \leftarrow$ True
    `Grasp`(g_0)
$done \leftarrow$ False
while not $done$ **do**
  $(O_t) \leftarrow$ Observation
  $\hat{F}_t \leftarrow f_\theta(O_t, [M_t])$, Predict the current flow
  $v_t \leftarrow$ `SelectDirection()`
  Apply a force to the end-effector in the direction of v for small duration t
  $done \leftarrow$ `EpisodeComplete()`

Our method first takes an observation O_0 and estimates the 3D articulation flow $\hat{F}_0 = f_\theta(O_0)$ for all points in the observation. Given the estimate of the 3D articulation flow $\hat{F}_0$, we now describe a general, closed-loop policy which takes flow as input and actuates an articulated object. The policy is executed in two phases:

1) Grasp Selection: Based on the estimated 3D articulation flow $\hat{F}_0$, the policy must decide the best place to grasp the object. In this work, we assume access to a suction-type gripper that (in the ideal case) can grasp any point on the object surface. We know that the ideal attachment point is the location on a part where the flow has the highest magnitude in order to achieve the most efficient actuation of the articulated part by maximizing its acceleration, as we showed by maximizing Equations 1 and 2. We use motion planning to move the end effector to this point, with the end-effector aligned to directly oppose the flow direction. We then grasp the object at this position (using a suction gripper), shown in the left hand side of Fig. 3. We assume a rigid contact between the gripper and this contact point going forward.

2) Articulation Execution: At each time step t, we record a new observation O_t and estimate the current flow $\hat{F}_t$. We then select the predicted flow direction $\mathbf{v}_t$ with the greatest magnitude from the visible points from the observation, as shown in the right hand side of Fig. 3. To handle objects with multiple articulated parts, we only consider flow vectors close to our point of contact (the contact point itself is likely occluded by the gripper and is thus not visible). While continuing to grasp the object, we then move the gripper in the direction $\mathbf{v}_t$. This process repeats in a closed loop fashion until the object has been fully-articulated, a max number of steps has been exceeded, or the episode is otherwise terminated. See Algorithm 1 for a full description of the generalized flow articulation algorithm.

E. FlowBot3D: A Robot Articulation System

With all the pieces of our generalized articulation policy in place, we now describe a real-world robot system – FlowBot3D – which leverages this generalized articulation policy. We define a tabletop workspace that includes a Sawyer BLACK 7-DoF robotic arm mounted to the tabletop with a pneumatic suction gripper as its end-effector, and an Azure Kinect RGB-D camera mounted at a fixed position and pointing at the workspace. See Figure 4 for an image of the workspace. We obtain point cloud observations of the scene from the Azure Kinect in the robot's base frame, filtering out non-object points, we use the method proposed in [33] to denoise the data (see supplementary materials for details). For robot control, we use a sampling-based planner, MoveIt! [7], which can move our robot to any non-colliding pose in the scene; we thus use motion planning to move the gripper to a pre-grasp pose. For the grasp and articulation, we directly control the end-effector velocity.

To select the point of contact for the suction gripper, we need to make some modifications from the idealized system described earlier. Unfortunately, a real suction gripper cannot make a proper seal on locations with high curvature (i.e. edges of the object and uneven surface features such as handles). Since the flow vector with the maximum magnitude is often at one of these extreme points, we must choose an alternative grasp point. While contact selection for suction-based grasping is a well-studied problem [2, 16, 17], we find that a simple heuristic performs acceptably; we choose the point with the highest flow magnitude subject to the following constraints:

1) The point itself is not within a certain distance of an edge, where edges are computed using a standard edge-detection algorithm (see supplement for details).
2) The estimated Gaussian curvature of that point does not exceed a certain threshold (see supplement for details).
3) The point is not within a distance of d of any points violating conditions 1 and 2. In practice, we set $d = 2\,\text{cm}$ (the radius of the suction tip).

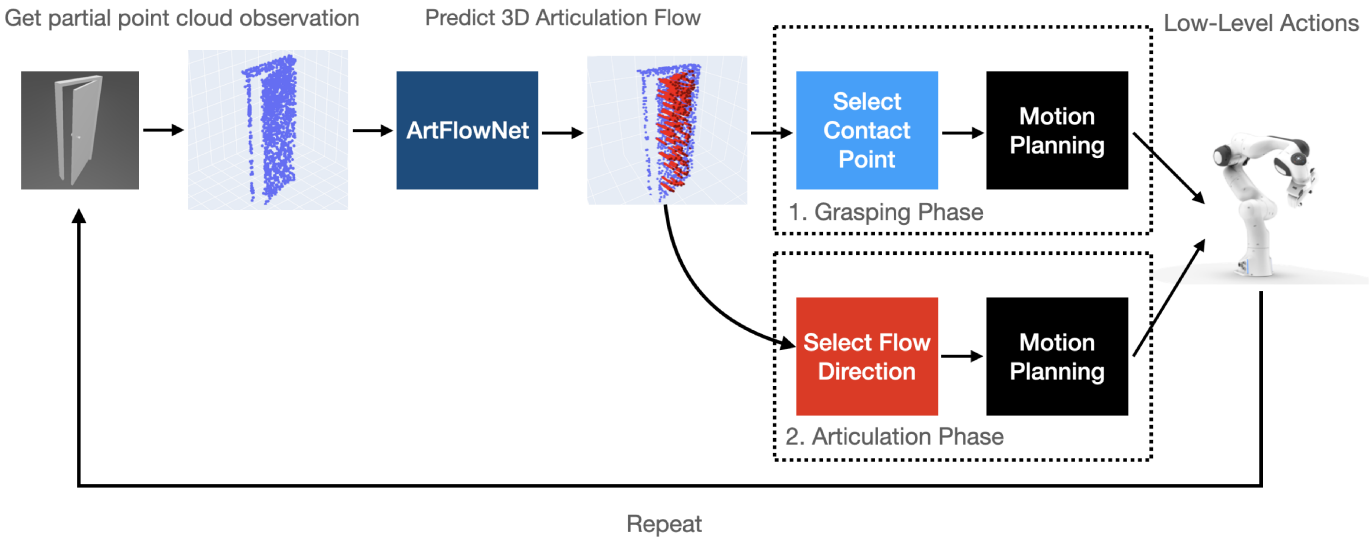


Fig. 3: FlowBot3D System Overview. Our system in deployment has two phases: the Grasp-Selection phase and the Articulation-Execution Phase. The dark red dots represent the predicted location of each point, and the light red lines represent the flow vectors connecting from the current time step's points to the predicted points. Note that the flow vectors are downsampled for visual clarity. In Grasp-Selection Phase, the agent observes the environment in the format of point cloud data. The point cloud data will then be post-processed and fed into the ArtFlowNet, which predicts per-point 3D flow vectors. The system then chooses the point that has the maximum flow vector magnitude and deploys motion planning to make contact with the chosen point using suction. In Articulation-Execution phase, after making suction contact with the chosen argmax point, the system iteratively observes the pointcloud data and predicts the 3D flow vectors. In this phase, the motion planning module would guide the robot to follow the maximum observable flow vector's direction and articulate the object of interest repeatedly.

Using this grasp selection method, we are able to execute our general articulation manipulation policy on a real robot. See the supplementary materials for other implementation details.

F. Training Details

We design a flow prediction network – which we refer to as ArtFlowNet – using the dense prediction configuration of PointNet++ [23] as a backbone, and train it using standard supervised learning with the Adam optimizer. We emphasize that we train a *single* model to predict 3DAF across all categories, using a dataset of synthetically-generated (observation, ground-truth flow) pairs based on the ground-truth kinematic and geometric structure provided by the PartNet-Mobility dataset [28]. During each step of training, we select an object in the dataset, randomize the state S of the object, and compute a new supervised pair (O_S, F_S), which we use to compute the loss and update the model parameters. During training, each object is seen in 100 different randomized configurations. Details of our dataset construction and model architecture can be found in the supplementary materials.

IV. Results

We conduct a wide range of simulated and real-world experiments to evaluate the FlowBot3D system.

A. Simulation Results

To evaluate our method in simulation, we implement a suction gripper in the ManiSkill environment [20], which serves as a simulation interface for interacting with the PartNet-Mobility dataset [28]. The PartNet-Mobility dataset contains 46 categories of articulated objects; following UMPNet [29], we consider a subset of PartNet-Mobility containing 21 classes, split into 11 training categories (499 training objects, 128 testing objects) and 10 entirely unseen object categories (238 unseen objects). Several objects in the original dataset contain invalid meshes, which we exclude from evaluation. We modify ManiSkill simulation environment to accommodate these object categories. We train our models (ArtFlowNet and baselines) exclusively on the training instances of the training object categories, and evaluate by rolling out the corresponding policies for every object in the ManiSkill environment. Each object starts in the "closed" state (one end of its range of motion), and the goal is to actuate the joint to its "open" state (the other end of its range of motion). For experiments in simulation, we include in the observation O_t a binary part mask indicating which points belong to the child joint of interest. Results are shown in Tables VI and VII[3].

Metrics. During our experiments, we calculate two metrics:

- *Normalized distance*: Following Xu et al. [29], we compute the normalized distance travelled by a specific child link through its range of motion. The metric is computed based on the final configuration after a policy rollout ($\mathbf{j}_{\text{end}}$) and the initial configuration ($\mathbf{j}_{\text{init}}$):

$$\mathcal{E}_{\text{goal}} = \frac{||\mathbf{j}_{\text{end}} - \mathbf{j}_{\text{goal}}||}{||\mathbf{j}_{\text{goal}} - \mathbf{j}_{\text{init}}||}$$

[3]Categories from left to right: stapler, trash can, storage furniture, window, toilet, laptop, kettle, switch, fridge, folding chair, microwave, bucket, safe, phone, pot, box, table, dishwasher, oven, washing machine, and door. Clipart pictures are borrowed from UMPNet paper with the authors' permission.

	Novel Instances in Train Categories													Test Categories										
	AVG.													AVG.										
Baselines																								
UMP-DI	0.29	**0.32**	0.33	0.16	0.18	0.37	0.14	0.19	**0.28**	0.72	**0.00**	0.55		0.36	0.32	0.62	0.15	0.00	0.41	0.61	0.17	0.34	0.38	0.58
Normal Direction	0.40	0.52	0.67	0.16	0.19	0.51	0.60	0.13	0.11	0.55	0.61	0.32		0.39	0.69	0.57	0.04	0.00	0.67	0.19	0.66	1.00	0.43	0.26
Screw Parameters	0.40	0.42	0.40	0.42	0.18	0.57	0.45	0.27	0.59	0.51	0.58	0.06		0.18	0.19	0.26	0.08	0.08	**0.23**	**0.06**	0.14	0.21	0.34	0.24
BC	0.74	0.59	0.91	0.63	0.75	0.57	1.00	1.00	0.98	0.62	0.96	0.10		0.87	0.81	0.63	1.00	0.93	0.99	0.74	0.95	0.96	0.83	0.88
DAgger E2E	0.64	0.39	0.85	0.61	0.73	0.50	1.00	0.96	0.90	0.54	0.48	0.10		0.83	0.73	0.62	1.00	0.80	0.95	0.73	0.83	0.98	0.81	0.85
DAgger Oracle	0.51	0.54	0.55	0.20	0.41	0.96	0.64	0.14	0.64	0.47	0.85	0.16		0.56	0.93	0.58	0.61	0.64	0.91	0.27	0.23	0.34	0.27	0.79
Baselines w/ Flow																								
BC + F	0.83	0.59	1.00	0.61	0.91	1.00	0.97	1.00	1.00	0.69	1.00	0.39		0.91	1.00	0.96	1.00	0.89	0.77	0.71	0.95	0.96	1.00	0.89
DAgger E2E + F	0.76	0.59	0.86	0.60	0.76	0.95	1.00	0.86	0.77	0.65	1.00	0.36		0.91	1.00	0.88	1.00	0.76	0.95	0.68	1.00	0.96	1.00	0.88
DAgger Oracle + F	0.50	0.59	0.53	0.25	0.51	0.58	0.86	0.17	0.65	0.56	0.48	0.38		0.60	0.77	0.71	0.62	0.73	0.91	0.28	0.43	0.47	0.31	0.73
Ours																								
FlowBot3D	**0.12**	0.32	0.23	0.11	0.09	**0.02**	**0.00**	**0.09**	0.32	**0.04**	0.13	**0.00**		**0.15**	**0.00**	0.21	**0.00**	**0.00**	0.33	0.22	0.09	0.07	**0.19**	0.35
FlowBot3D w/o Mask	0.17	0.32	0.37	0.10	0.11	0.15	0.00	0.11	0.33	0.05	0.07	0.29		0.19	0.16	0.24	0.05	0.00	0.24	0.24	0.17	0.27	0.19	0.37
FlowBot3D w/o Mask (+VPA)	0.16	0.33	**0.09**	**0.07**	**0.07**	0.16	0.00	0.14	0.49	0.27	0.11	0.00		0.16	0.11	**0.17**	0.23	0.00	0.53	0.10	**0.05**	**0.00**	0.23	**0.20**
Oracle w/ GT 3DAF	0.05	0.10	0.10	0.03	0.11	0.06	0.00	0.12	0.00	0.00	0.02	0.00		0.16	0.00	0.12	0.95	0.00	0.12	0.14	0.02	0.00	0.13	0.12

TABLE I: Normalized Distance Metric Results (↓): Normalized distances to the target articulation joint angle after a full rollout across different methods. The lower the better.

	Novel Instances in Train Categories													Test Categories										
	AVG.													AVG.										
Baselines																								
UMP-DI	0.52	0.60	0.33	0.65	0.73	0.29	0.67	0.80	**0.50**	0.11	**1.00**	0.00		0.45	0.83	0.03	0.50	1.00	0.31	0.29	0.78	0.33	0.31	0.20
Normal Direction	0.31	0.40	0.00	0.51	0.71	0.00	0.00	0.80	0.50	0.00	0.00	0.50		0.31	0.00	0.00	0.50	1.00	0.00	0.55	0.00	0.00	0.10	0.64
Screw Parameters	0.50	0.50	0.53	0.51	0.80	0.21	0.55	0.60	0.17	0.37	0.43	0.80		0.67	0.17	0.63	0.67	0.92	**0.69**	**0.92**	0.83	0.75	0.50	0.72
BC	0.14	0.40	0.00	0.20	0.18	0.14	0.00	0.00	0.00	0.11	0.00	0.50		0.04	0.17	0.00	0.00	0.04	0.00	0.15	0.00	0.00	0.00	0.00
DAgger E2E	0.14	0.60	0.00	0.26	0.09	0.28	0.00	0.00	0.00	0.00	0.25	0.00		0.04	0.00	0.00	0.00	0.20	0.00	0.17	0.02	0.00	0.00	0.00
DAgger Oracle	0.29	0.40	0.33	0.80	0.27	0.00	0.00	0.80	0.00	0.11	0.00	0.50		0.20	0.00	0.00	0.00	0.36	0.00	0.29	0.49	0.33	0.31	0.20
Baselines w/ Flow																								
BC + F	0.11	0.40	0.00	0.26	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.50		0.03	0.00	0.00	0.00	0.08	0.00	0.15	0.05	0.00	0.00	0.00
DAgger E2E + F	0.14	0.40	0.00	0.00	0.09	0.00	0.00	0.00	0.50	0.00	0.00	0.50		0.04	0.00	0.00	0.00	0.20	0.00	0.15	0.05	0.00	0.00	0.00
DAgger Oracle + F	0.33	0.40	0.33	0.58	0.45	0.00	0.00	0.80	0.00	0.11	0.50	0.50		0.16	0.00	0.00	0.00	0.24	0.00	0.34	0.41	0.33	0.12	0.16
Ours																								
FlowBot3D	**0.77**	0.57	0.56	0.88	0.82	**0.86**	**1.00**	**0.80**	0.50	0.78	0.75	**1.00**		0.69	**1.00**	0.56	**1.00**	**1.00**	0.38	0.43	0.84	0.83	0.43	0.44
FlowBot3D w/o Mask	0.72	**0.67**	0.55	0.85	0.82	0.57	1.00	0.80	0.50	**0.89**	0.75	0.50		0.62	0.83	0.59	0.50	1.00	0.62	0.28	0.76	0.58	0.56	0.50
FlowBot3D w/o Mask + VPA	0.73	0.60	**0.67**	**0.88**	**0.91**	0.43	1.00	0.80	0.50	0.56	0.72	1.00		**0.70**	0.50	**0.63**	0.50	1.00	0.23	0.90	**0.88**	**1.00**	**0.63**	**0.72**
Oracle w/ GT 3DAF	0.92	0.80	1.00	0.97	0.82	0.71	1.00	0.80	1.00	1.00	1.00	1.00		0.82	1.00	0.85	0.00	1.00	0.85	0.86	0.98	1.00	0.81	0.88

TABLE II: Success Rate Metric Results (↑): Fraction of success trials (normalized distance less than 0.1) of different objects' categories after a full rollout across different methods. The higher the better.

- *Success*: We also define a binary success metric, which is computed by thresholding the final resulting normalized distance at δ: Success $= \mathbb{1}(\mathcal{E}_{\text{goal}} \leq \delta)$. We set $\delta = 0.1$, meaning that we define a success as articulating a part for more than 90%.

Baseline Comparisons: We compare our proposed method with several baseline methods:

- **UMP-DI**: We implement a variant[4] of UMPNet's Direction Inference network (DistNet) [29], where instead of bootstrapping an action scoring function from interaction, we learn the scoring function by regressing the cosine distance between a query vector and the ideal flow vector for a contact point. At test time, we select the contact point based on ground-truth 3DAF, and after contact has been achieved we use CEM to optimize the scoring function to predict the action direction at every timestep.
- **Normal Direction**: We use off-the-shelf normal estimation to estimate the surface normals of the point cloud using Open3D [34]. To break symmetry, we align the normal direction vectors to the camera. At execution time, we first choose the ground-truth maximum-flow point and then follow the direction of the estimated normal vector of the surface.
- **Screw Parameters**: We predict the screw parameters for the selected joint of the articulated object. We then generate 3DAF from these predicted parameters and use the FlowBot3D policy on top of the generated flow.
- **Behavioral Cloning (BC)**: The agent takes as input a point cloud and outputs the action of the robot. The agent uses the PointNet-Transformer architecture proposed in [20]. The agent is trained end-to-end via L2 regression

[4]We could not yet compare directly to UMPNet, as their model and simulation environment had not yet been released at the time between submission and publication.

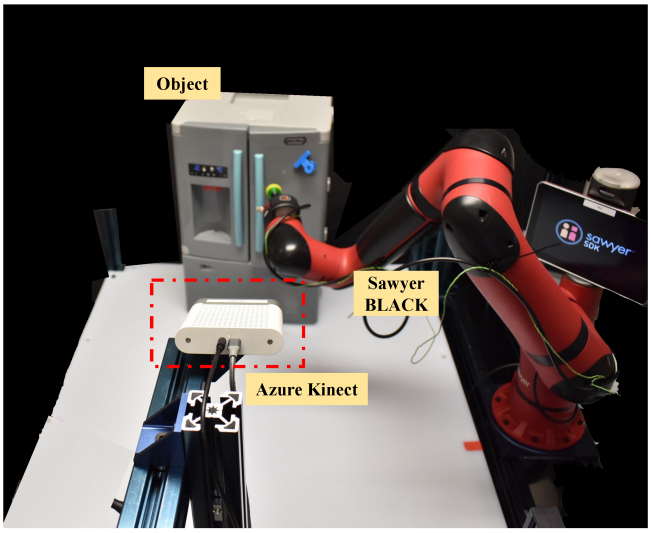


Fig. 4: Workspace setup for physical experiments. The sensory signal comes from an Azure Kinect depth camera, and the agent is a Sawyer BLACK robot.

Fig. 5: Fourteen test objects for our real-world experiments. Please refer to Supplementary Material for the exact category of each object.

on trajectories provided by an oracle version of GT 3DAF.

- **BC + F**: Same as BC, but with ground-truth flow at input.
- **DAgger E2E**: We also conduct behavioral cloning experiments with DAgger [24] on the same expert dataset as in the BC baseline. We train it end-to-end (E2E), similar to the BC model above.
- **DAgger E2E + F**: Same as DAgger E2E, but with ground-truth flow as an input.
- **DAgger Oracle**: A two-step policy, where we first use ground-truth flow to select a contact point using the Generalized Articulation Policy heuristic, and train DAgger on expert trajectories generated after the point of contact.
- **DAgger Oracle + F**: Same as DAgger Oracle, but with ground-truth flow at input.
- **Oracle w/ GT 3DAF**: An oracle version of FlowBot3D that uses ground truth 3DAF vectors instead of the predicted ones for both phases. This serves as an upper bound of FlowBot 3D's performance.[5]

Each method above consists of a single model trained across all PartNet-Mobility training categories. For a more straightforward comparison, we dedicate Table VI and VII to evaluations in the SAPIEN simulator and we defer the comparison between UMPNet and FlowBot3D to the supplementary material.

Analysis: We can draw two conclusions from our simulated evaluation. First, our formulation of FlowBot3D has a very high success rate across all categories, including test categories, which are completely novel types of objects (but may contain similar parts and articulation structures). This is evidence that the ArtFlowNet network is learning salient geometric features to predict the location and character of articulated points. Based on visual interpretation of actual predicted flows, ArtFlowNet is particularly adept at recognizing doors, lids, drawers, and other large articulated features. One might have thought that 3DAF is essentially estimating normal directions, but this is not the case, as seen in the results of the Normal Direction baseline. Normal Direction estimation suffers from occlusion issues and the normal is not always the correct direction to actuate the object (for example, for the spherical-shaped lid of a teapot). Additionally, our method's accuracy increases when the object is at least partially open, because there is less ambiguity about object structure than when an object is fully "closed". The UMP-DI baseline exhibits similar properties, but the implicit optimization yields noisier direction predictions. Second, none of the Behavior Cloning and DAgger policies, nor their flow-based variants, perform well. The best BC baseline, DAgger Oracle + F, is only able to fully articulate objects 33% of the time.

UMPNet Pybullet Environment: The simulation environment used in the original UMPNet evaluations [29] is a PyBullet-based environment with different physical and collision parameters. However, the source code to run the UMPNet environment was not available for us to run until after this paper was submitted for review; we have since obtained a copy of this environment, and evaluate our method on their environment in the supplementary materials.

B. Real-World Experiments

To evaluate the performance of FlowBot3D when executed in a real robotic environment, we design a set of of real-world experiments in which we attempt to articulate a variety of different household objects using the Sawyer robot in our workspace, as shown in Fig. 4. Our experiment protocol is thus: for each object in the dataset, we conducted 5 trials of each method. For each trial, the object is placed in the scene at a random position such that the articulations are visible and the robot can reach every position in the range of motion of each articulation. The policy is then executed for at most 10

[5]The description files of the phone meshes contain wrong rotation axis, thus the poor performance of the oracle policy on that category.

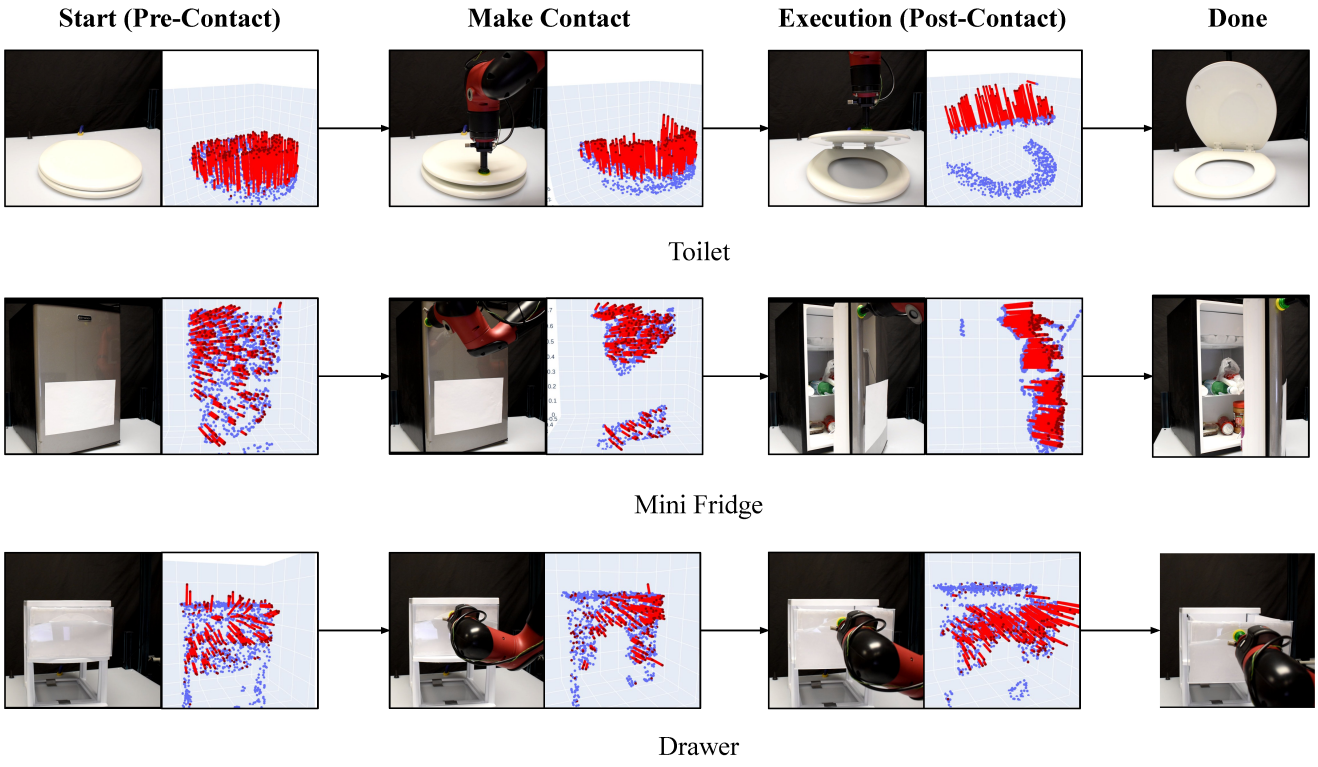


Fig. 6: Real world examples of FlowBot3D executing an articulation policy based on predicting 3D Articulated Flow. Notice that even with occlusions, such as in the intermediate mini-fridge observation, the network is able to predict reasonable 3D articulation flow vectors for downstream policy.

steps, terminating earlier if success has been achieved or if the policy predicts an action that cannot be executed safely (this case is infrequent). We conducted one round of evaluation (70 trials in total) for each of the following methods:

- **FlowBot3D**: The version of our generalized articulation policy as presented in Section III. In experiments, we use an ArtFlowNet trained without a part mask in the observation space. In addition, since the camera position in reality is different from that in the ManiSkill environments, we apply a viewpoint augmentation (VPA) at training time, where we render synthetic point clouds from various camera angles in simulation.
- **DAgger Oracle**: The DAgger model trained in simulation to produce closed-loop motion directions, but with the contact selection predicted by the FlowBot3D model.

As in our simulated experiments, we use a single model trained in simulation across multiple object categories without any further finetuning.

# Objects	2	1	1	2	1	1	1	1	1	1	2

TABLE III: Real-world objects used during our experiments.

Objects: We assemble a set of real-world objects that are representative of typical articulations a human may encounter in the real world: doors, drawers, hinges, etc. The objects were selected before experimentation began, and the only criteria for inclusion were 1) that it fit in the workspace, 2) it had a surface that a suction gripper could attach to and actuate, and 3) it wasn't too dark or reflective, so as to be seen by the Azure Kinect's depth camera. Each object falls into one of either the training or test classes we selected from the PartNet-Mobility. We also include several jars with lids, which, while not strictly articulated as 1-DoF joints, can be articulated like a prismatic joint. See Figure 5 and Table III for a summary of the dataset, and the supplementary materials for specifics for each object.

Metrics: During our trials, we compute the following metrics for each policy:

- Overall Success: Was the object articulated more than 90% of its range of motion (defined per-object)?
- Contact Success: Was the contact point chosen on an a joint that can move, and was the suction tip able to successfully form a seal at that point?
- Average Distance: Conditioned on a successful contact, what was the average distance from the end of the object's range of motion after the policy terminated?
- Motion Success: After successful contact, was the object articulated more than 90 % of its range of motion?

Details about how our trials are conducted and measurements computed can be found in the supplementary materials.

Quantitative analysis: We present summary metrics in Table IV, and a per-object summary in our supplementary materials. Across all metrics, FlowBot3D performs substantially

Method	Overall Succ.	Contact Succ.	Avg. Dist.	Motion Succ.
FlowBot3D	**45/70 (64.3%)**	64/70 (91.4%)*	**0.22**	**45/64 (70.3%)**
DAgger Oracle	10/70 (14.3%)	**68/70 (97.1%)***	0.73	10/68 (14.7%)

TABLE IV: Trials for FlowBot3D. *Note that both methods in the Contact Success column use the same FlowBot3D contact prediction and execution policy.

better than the DAgger baseline. In absolute terms, the policy succeeds a high fraction of the time (64%); the policy selects a suitable contact point on the object 91% of the time, and succeeded 70% of the time after contact was established.

In contrast, the baseline policy succeeded in a very small number of cases, only 14% of the time. While contact rates were comparable to the trials conducted for FlowBot3D (they use the same contact selection method), the motions predicted were almost always unsuccessful.

Qualitative analysis of FlowBot3D: A major goal of our real-world trials was to evaluate how well the Flowbet3D policy transfers from simulation to reality without any retraining. We find that the overall policy performs surprisingly well, and the ArtFlowNet module – trained exclusively on point clouds rendered in simulation – generalizes impressively to real-world objects, producing high-fidelity flow predictions on a range of real objects. This is because there isn't much of a domain shift in the point cloud observations, and the geometric features that signal an articulation are fairly consistent.

Failure modes: We have found that the majority of trial failures were due to two reasons in real world: flow prediction error and contact failure. For flow prediction errors, after making contact with the object, executing an incorrect 3D articulation flow vector will drive the gripper away from the object, causing the gripper to lose contact with the object. The bulk of flow prediction errors happen either because the robot occludes too much of the scene (which might be rectified by multiple viewpoints, temporal filtering, or a recurrent policy), or because the robot fails to detect the presence of articulations (this occurs on the real oven, for instance). For contact failures, the contact selection heuristic might not filter out all ungraspable points and thus the robot might choose a contact point that is difficult or impossible to make a complete seal on during suction. Overall, we theorize that many of the failures could be mitigated by improving the compliance and control of the gripper, and including a stronger contact quality prediction module.

C. Simulation Ablations

We conduct two ablations on the design decisions made for ArtFlowNet:

- **Including a part mask** We study the effect of providing a segmentation mask of the articulated part of interest as input to ArtFlowNet; such a mask could theoretically be obtained by a segmentation method or provided by a human to specify the articulation task, as in the SAPIEN challenge [28]. We find while the inclusion of a mask improves predictions in ambiguous cases (i.e. when a door is closed and coplanar with its parent link), removing the mask only decreases performance a small amount (see Tables VI and VII).
- **Applying viewpoint augmentations during training**: We analyze how randomizing the camera viewpoint during the synthetic dataset generation step affects model performance. We find that augmenting the viewpoint has little effect on the performance in a simulated environment (see Tables VI and VII), but improves performance in sim-to-real transfer.

V. Conclusion

In this work, we propose a novel visual representation for articulated objects, namely **3D Articulation Flow**, as well as a policy – **FlowBot3D** – which leverages this representation to successfully manipulate articulated objects. We demonstrate the effectiveness of our method in both simulated and real environments, and observe strong sim-to-real transfer generalization.

While our method shows strong performance on a range of object classes, there is substantial room for improvement. One class of improvements is in system-building and engineering: with a more compliant robotic arm controller, as well as a more sophisticated contact prediction system, we believe we would be able to eliminate a wide class of failure modes. However, the remaining failure modes raise questions we would like to explore in future work. For instance, we would like to explore how our flow representation models might be used in an online adaptation setting, so that incorrect predictions can be corrected. We also would like to explore how our representation might be useful when learning from demonstrations, or in other more complex manipulation settings.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-1849154 (NSF S&AS), IIS-2046491 (NSF CAREER), and DGE-1745016 (NSF GRFP), as well as LG Electronics. We are grateful to Thomas Weng, Brian Okorn, Daniel Seita, Shikhar Bahl, and Russell Mendonca for feedback on the paper.

References

[1] Artemij Amiranashvili, Alexey Dosovitskiy, Vladlen Koltun, and Thomas Brox. Motion perception in reinforcement learning with dynamic objects. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 156–168. PMLR, 2018.

[2] Yahav Avigal, Vishal Satish, Zachary Tam, Huang Huang, Harry Zhang, Michael Danielczuk, Jeffrey Ichnowski, and Ken Goldberg. Avplug: Approach vector planning for unicontact grasping amid clutter. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1140–1147. IEEE, 2021.

[3] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

[4] Felix Burget, Armin Hornung, and Maren Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *2013 IEEE International Conference on Robotics and Automation*, pages 1656–1662, May 2013.

[5] Jae-Sook Cheong, A Frank Van Der Stappen, Ken Goldberg, Mark H Overmars, and Elon Rimon. Immobilizing Hinged Polygons. *Int. J. Comput. Geom. Appl.*, 17(01): 45–69, February 2007.

[6] Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *2010 IEEE International Conference on Robotics and Automation*, pages 1799–1806, May 2010.

[7] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.

[8] Siyuan Dong, Devesh K Jha, Diego Romeres, Sangwoon Kim, Daniel Nikovski, and Alberto Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6437–6443. IEEE, 2021.

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks, 2015.

[10] Berthold K P Horn and Brian G Schunck. Determining optical flow. *Artif. Intell.*, 17(1):185–203, August 1981.

[11] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Learning to predict part mobility from a single static snapshot. *ACM Trans. Graph.*, 36(6):1–13, November 2017.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[13] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. ScrewNet: Category-Independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677, May 2021.

[14] Dov Katz, Yuri Pyuro, and Oliver Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *Robotics: Science and Systems IV*. Robotics: Science and Systems Foundation, June 2008.

[15] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-Level articulated object pose estimation, 2020.

[16] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*, pages 5620–5627. IEEE, 2018.

[17] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[18] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[19] Kaichun Mo, Leonidas J Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6813–6823, 2021.

[20] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. ManiSkill: Learning-from-Demonstrations benchmark for generalizable manipulation skills. *arXiv e-prints*, pages arXiv–2107, 2021.

[21] Venkatraman Narayanan and Maxim Likhachev. Task-oriented planning for manipulating articulated mechanisms under model uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3095–3101, May 2015.

[22] Sudeep Pillai, Matthew R Walter, and Seth Teller. Learning articulated motions from visual demonstration. *arXiv preprint arXiv:1502.01659*, 2015.

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[24] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[25] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs field transforms for optical flow. In *Computer Vision – ECCV 2020*, pages 402–419. Springer International Publishing, 2020.

[26] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019.

[27] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In

Conference on Robot Learning, pages 192–202. PMLR, 2022.

[28] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, and Others. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[29] Zhenjia Xu, He Zhanpeng, and Shuran Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022.

[30] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. Rpm-net: recurrent prediction of motion and parts from point cloud. *arXiv preprint arXiv:2006.14865*, 2020.

[31] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *arXiv preprint arXiv:1809.07417*, 2018.

[32] Vicky Zeng, Timothy E Lee, Jacky Liang, and Oliver Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2020.

[33] Harry Zhang, Jeffrey Ichnowski, Yahav Avigal, Joseph Gonzales, Ion Stoica, and Ken Goldberg. Dex-Net AR: Distributed deep grasp planning using a commodity cellphone and augmented reality app. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 552–558, May 2020.

[34] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

APPENDIX

A. Hardware

In all of real-world experiments, we deploy our system on a Rethink Sawyer Robot and the sensory data (point cloud) come from an Azure Kinect depth camera. The robot's end effector is an official Saywer Pneumatic Suction Gripper with a suction cup with a diameter of 3 cm. The air supply of the suction gripper is provided by a California Air Tools compressor.

B. Workspace

We set up our workspace in a 1.08 m by 1.00 m space put together using Vention beams. We set up the Azure Kinect camera such that it points toward the center of workspace and has minimal interference with the robot arm-reach trajectory. Collision geometry are set up using MoveIt's collision box construction tool. We add a number of boxes representing the camera and Vention beams that can potentially be blocking the robot during motion planning.

C. Hand-Eye Calibration

For Hand-Eye Calibration, we are using the Easy-Hand-Eye ROS package that calculates the transformation from the camera frame to world frame using an ArUco marker fixed on the robot's end effector. The process requires about 30 samples of the robot pose and ArUco marker pose combinations.

D. Foreground Segmentation

In simulated experiments, we have access to segmentation masks that segment out tabletop and robot from the collected point cloud. In real-world experiments, however, we need to programatically segment out those points ourselves.

Tabletop. We segment out the tabletop plane by simply subtracting the points with z values less than 0.015 m from the collected point cloud after calibration because the table top is placed 1.5 cm below the robot base.

Robot. The robot points are masked out in real-time by rendering the robot 3D model using its URDF file. This is done through a ROS package called Real Time URDF Filter. This filter assumes a perfect calibration of the camera. When the calibration is slightly off, some trailing points from the robot might remain in scene. Thus, we also statistically remove the outliers from the resulting point cloud because the remaining robot points are sparser than the object's points.

E. Contact Point Heuristic

In simulation, the suction contact is modeled by a kinematic constraint between the gripper point and the contact point. Therefore, in simulation, we have a perfect contact that can almost always successfully grasp the desired part. In real-world experiments, due to the complication of the physics of the suction gripper and the geometry of the target part, we can not always guarantee a successful grasp. Therefore, we add an extra heuristic upon the max-flow selection when selecting which point to grasp. Specifically, we add an interior point selection procedure that calculates the curvature of the points using PCA and we choose the point with curvature

value smaller than a threshold. If the max-flow point has a curvature value higher than the threshold, we discard that point and choose the nearest low-curvature point at least 2 cm away from the max-flow point.

F. Grasp Selection Details

In the Grasp Selection phase of real-world experiments, we predict and estimate the part's 3D articulated flow vectors using FlowNet. We then use the aforementioned contact point heuristic to filter out points that have high curvature values. If the max-flow point is within the remaining points, we keep it and use it as the selected contact point. Otherwise, we choose the nearest low-curvature point at least 2 cm away from the max-flow point. Once we have selected the point, we have also selected the end effector's goal translation. For goal orientation, we align the end effector with the flow vector. The procedure is explained here: assume that the axis connecting the suction gripper tip to the robot hand is called $\mathbf{v}_1$ and the chosen flow vector is $-\mathbf{v}_2$, we aim to find a rotation that aligns $\mathbf{v}_1$ to $\mathbf{v}_2$ (because the robot approach direction is opposite to the flow direction). The difference of the rotation expressed in quaternion is calculated as follows:

$$\begin{aligned}
\phi_{12} &= \cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2) \\
\omega &= \mathbf{v}_1 \times \mathbf{v}_2 = [\omega_x, \omega_y, \omega_z] \\
q_x &= \omega_x \cdot \sin(\omega/2) \\
q_y &= \omega_y \cdot \sin(\omega/2) \\
q_z &= \omega_z \cdot \sin(\omega/2) \\
q_w &= \cos(\omega/2) \\
\mathbf{q} &= [q_x, q_y, q_z, q_w] \\
\mathbf{q}_{\text{diff}} &= \frac{\mathbf{q}}{||\mathbf{q}||}.
\end{aligned}$$

Therefore, when given the robot's starting rotation quaternion $\mathbf{q}_{\text{start}}$, the goal orientation of the robot end-effector $\mathbf{q}_{\text{goal}}$ is given by:

$$\mathbf{q}_{\text{goal}} = \mathbf{q}_{\text{diff}} \cdot \mathbf{q}_{\text{start}}$$

G. Robot Control Paradigm

In the Grasp Selection Phase of real-world experiments, the robot is controlled using position control by inputting the end-effector pose and solving for the trajectory using an RRTConnect-based [17] IK solver. One caveat about the Grasp Selection Phase in real world is that the robot does not make contact with the select point directly. Instead, the robot first aligns with the chosen flow vector and plans to a point 10 cm in the chosen 3D articulated flow direction away from the max-flow point. Then the robot switches the control mode to velocity control and approaches the proposed point in the aligned (negative selected flow) direction slowly until the force sensor of the robot reports reading greater than a threshold, meaning the robot makes contact with the object. Then in the Articulation-Execution Phase, the velocity controller takes as input the translational velocity represented by the current time step's normalized predicted articulation flow vector multiplied by a constant to decrease the speed and the rotational velocity as the aforementioned $\mathbf{q}_{\text{diff}}$ converted to Euler angles multiplied by another constant to decrease the angular speed.

H. Network Architecture

ArtFlowNet is based on the PointNet++ [27] architecture. The architecture largely remains similar to the original architecture except for the output head. Instead of using a segmentation output head, we use a regression head. The ArtFlowNet architecture is implemented using Pytorch-Geometric [10], a graph-learning framework based on PyTorch. Since we are doing regression, we use standard L2 loss optimized by an Adam optimizer [16].

I. Ground Truth 3DAF Generation

We implement efficient ground truth 3D Articulation Flow generation. At each timestep, the system reads the current state of the object of interest in simulation as an URDF file and parses it to obtain a kinematic chain. Then the system uses the kinematic chain to analytically calculate each point's location after a small, given amount of displacement. In simulation, since we have access to part-specific masks, the calculated points' location will be masked out such that only the part of interest will be articulated. Then we take difference between the calculated new points and the current time step's points to obtain the ground truth 3D Articulation Flow.

J. Simulator Modifications

We heavily modify the ManiSkill [23] environment, which is a high-level wrapper of the SAPIEN [32] simulator. Specifically, we add in a variety of PartNet-Mobility objects to the environment for more diverse training dataset. We obtain a list of training and testing objects from the authors of UMPNet [33]. We have filtered out some phone objects and door objects due to the collision of meshes in the SAPIEN simulator upon loading, but the dataset remains largely identical to the one used in UMPNet. Furthermore, we implement efficient online ground truth 3D articulation flow calculation in the ManiSkill environment for generating training data online. We also implement camera viewpoint sampling by randomizing the azimuth and elevation for the VPA model training. Instead of using a full robot arm, we only use a floating gripper with 8 DoF (x, y, z for translation, r, p, y for rotation, and speed parameter for each of the two fingers on the gripper) controlled by a velocity controller. The two gripper fingers' speed parameters are not learned in Behavioral Cloning as the two fingers remain closed. To simulate suction, we create a strong force between the gripper fingers and the target object since kinematic constraints are not directly supported in the SAPIEN simulator.

K. Hyperparameters

We use a batch size of 64 and a learning rate of 1e-4. We use the standard set of hyperparameters from the original PointNet++ paper.

Here we briefly illustrate FlowBot3D system in simulation. In simulation, the suction is implemented using a strong force between the robot gripper and the target part.

As shown in Section IV-B, we use 14 different objects in real world experiments. The objects labeled 1-14 in Fig. 5 are described here in Table V.

Label	ID	Category	Type
1	chest_1	Box	Revolute
2	teapot_1	Kettle	Prismatic
3	toilet_1	Toilet	Revolute
4	fridge_1	Refrigerator	Revolute
5	oven_1	Oven	Revolute
6	drawer_1	Storage	Prismatic
7	safe_1	Safe	Revolute
8	microwave_1	Microwave	Revolute
9	minifridge_1	Refrigerator	Revolute
10	jar_1	Kitchen Pot	Prismatic
11	jar_2	Kitchen Pot	Prismatic
12	trash_1	Trash Can	Revolute
13	laptop_1	Laptop	Revolute
14	box_1	Box	Revolute

TABLE V: Labels and their corresponding objects and the objects' articulation types shown in Fig. 5 of the paper. Note that jar_1 and jar_2 are not technically kitchen pots but they do have lids similar to kitchen pots in functionality and have identical ariculation parameters.

In Fig. 8, we show the 14 objects individually for more clarity.

We perform a direct evaluation of FlowBot3D in the UMPNet simulation environment, which was only made public after this manuscript was accepted for publication. There are several major differences between our main simulation environment and the UMPNet evaluation environment:

- The UMPNet environment uses the PyBullet physics simulator, whereas we use the SAPIEN environemnt (backed by PhysX).
- The UMPNet environment disables collisions between the gripper geometry and the rest of the object (except for the part where contact is made). We leave full contact enabled.
- The UMPNet environment has a hard contact constraint between the object and the gripper, whereas our contact is softer, acting more like a spring.

We use the UMPNet evaluation script without modification, with the exception that the chosen action is selected based on FlowBot3D instead. In Tables VI and VII, we present the results for the following methods:

- **UMPNet**: We run a pre-trained UMPNet model with the official UMPNet code following the exact same evaluation procedure listed in [33]. The numbers here are consistent with those in the UMPNet paper.
- **FlowBot3D in UMPNet Environment**: FlowBot3D trained and evaluated with the camera parameters and objects' placement randomization from UMPNet's PyBullet environment. Note that in test time, UMPNet takes as input a goal of the articulated object in its fully closed or fully open state, so we use the ground-truth goal to decide if we need to invert the output 3DAF directions (i.e. if the ground-truth goal is a fully closed state, we invert the output direction).

Overall, the two methods perform similarly on the task. However, while the ArtFlowNet was retrained on point clouds generated in PyBullet, the performance was not significantly tuned on the different task distribution in the UMPNet dataset.

In Table VIII and IX, we show the full trials results, which contains the metrics averaged over all 5 trials for each object.

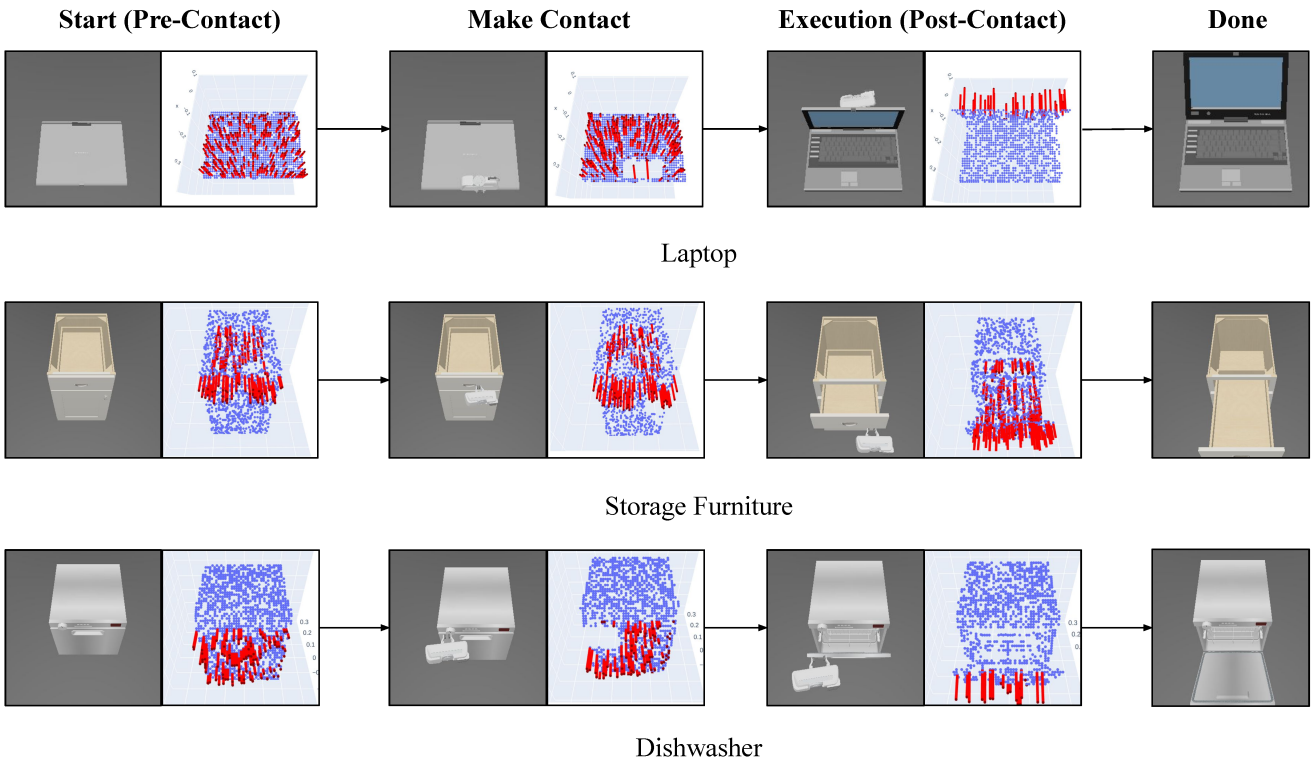


Fig. 7: Simulated rollout examples

	Novel Instances in Train Categories													Test Categories										
	AVG.													AVG.										
Baselines																								
UMPNet	0.18	**0.18**	**0.17**	0.32	0.32	0.05	0.06	**0.12**	**0.24**	0.23	0.18	0.08		**0.15**	0.23	**0.14**	0.04	**0.00**	0.25	0.27	0.09	0.21	**0.13**	**0.19**
Ours																								
FlowBot3D in UMPNet	**0.17**	0.42	0.22	**0.16**	**0.17**	**0.03**	**0.00**	0.20	0.51	**0.07**	**0.00**	**0.08**		0.21	**0.17**	0.29	**0.00**	0.06	**0.21**	**0.10**	**0.06**	**0.16**	0.29	0.73

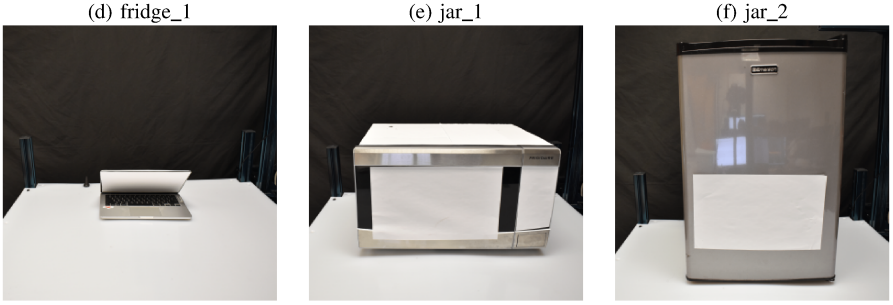
TABLE VI: Normalized Distance Metric Results: Normalized distances evaluated in the official UMPNet environment to the target articulation joint angle after a full rollout across different methods. The lower the better.

	Novel Instances in Train Categories													Test Categories										
	AVG.													AVG.										
Baselines																								
UMPNet	0.73	**0.73**	0.71	0.60	0.49	0.89	0.90	0.79	**0.60**	0.64	0.78	0.86		**0.75**	0.55	**0.80**	0.89	**1.00**	0.66	0.64	**0.77**	0.64	**0.75**	**0.76**
Ours																								
FlowBot3D in UMPNet	**0.81**	0.53	**0.74**	**0.81**	**0.82**	**0.96**	**0.99**	**0.79**	0.44	**0.90**	**1.00**	**0.89**		0.70	**0.69**	0.63	**1.00**	0.94	**0.67**	**0.89**	0.75	**0.66**	0.69	0.14

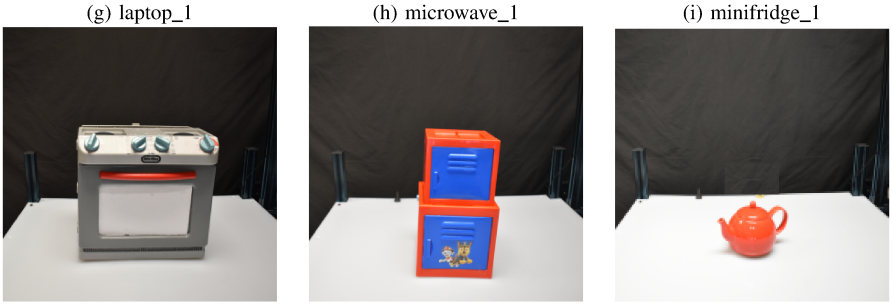
TABLE VII: Success Rate Metric Results: Fraction of success trials (normalized distance less than 0.1) of different objects' categories after a full rollout across different methods evaluated in the official UMPNet environment. The higher the better.

(a) box_1 (b) chest_1 (c) drawer_1

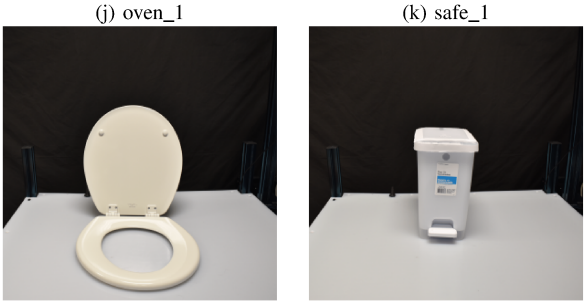
(d) fridge_1 (e) jar_1 (f) jar_2



(g) laptop_1 (h) microwave_1 (i) minifridge_1



(j) oven_1 (k) safe_1 (l) teapot_1



(m) toilet_1 (n) trashcan_1

Fig. 8: Objects in the dataset for real world experiments

Object ID	Object Category	Success/Total	Success %	Contact Success/Total	Distance	Motion-Only Success/Total
chest_1	Box	3/5	60%	4/5	0.22	3/4
teapot_1	Kettle	5/5	100%	5/5	0.00	5/5
toilet_1	Toilet	4/5	80%	5/5	0.02	4/5
fridge_1	Refrigerator	3/5	60%	3/5	0.11	5/5
oven_1	Oven	0/5	0%	5/5	1.00	0/5
drawer_1	Storage	3/5	60%	3/5	0.40	3/3
safe_1	Safe	1/5	20%	2/5	0.73	1/2
microwave_1	Microwave	3/5	60%	5/5	0.11	3/5
minifridge_1	Refrigerator	2/5	40%	5/5	0.155	2/5
jar_1	Kitchen Pot	5/5	100%	5/5	0.00	5/5
jar_2	Kitchen Pot	5/5	100%	5/5	0.00	5/5
trash_1	Trash Can	5/5	100%	5/5	0.02	5/5
laptop_1	Laptop	4/5	100%	5/5	0.07	4/5
box_1	Box	2/5	40%	5/5	0.28	2/5
SUMMARY	**-**	**45/70**	**64.3%**	**64/70**	**0.22**	**45/64**

TABLE VIII: Real-World Trials for FlowNet

Object ID	Object Category	Success/Total	Success %	Contact Success/Total	Distance	Motion-Only Success/Total
chest_1	Box	1/5	20%	5/5	0.80	1/5
teapot_1	Kettle	2/5	40%	5/5	0.60	2/5
toilet_1	Toilet	0/5	0%	5/5	0.78	0/5
fridge_1	Refrigerator	0/5	0%	5/5	1.00	0/5
oven_1	Oven	0/5	0%	5/5	1.00	0/5
drawer_1	Storage	1/5	20%	5/5	0.72	1/5
safe_1	Safe	1/5	20%	3/5	0.70	1/3
microwave_1	Microwave	0/5	0%	5/5	1.00	0/5
minifridge_1	Refrigerator	0/5	0%	5/5	1.00	0/5
jar_1	Kitchen Pot	3/5	60%	5/5	0.40	3/5
jar_2	Kitchen Pot	1/5	20%	5/5	0.80	1/5
trash_1	Trash Can	0/5	0%	5/5	1.00	0/5
laptop_1	Laptop	1/5	20%	5/5	0.81	1/5
box_1	Box	1/5	20%	5/5	0.80	1/5
SUMMARY	**-**	**10/70**	**14.3%**	**68/70**	**0.73**	**10/68**

TABLE IX: Real-World Trials for DAgger Oracle