

I. ROBOT SYSTEM DETAILS

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 Sawyer 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 programmatically 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}_{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 [3] 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}_{diff} converted to Euler angles multiplied by another constant to decrease the angular speed.

II. TRAINING DETAILS

A. Network Architecture

ArtFlowNet is based on the PointNet++ [5] 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 [1], a graph-learning framework based on PyTorch. Since we are doing regression, we use standard L2 loss optimized by an Adam optimizer [2].

B. 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.

C. Simulator Modifications

We heavily modify the ManiSkill [4] environment, which is a high-level wrapper of the SAPIEN [6] 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 [7]. 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.

D. 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.

III. SIMULATION EXPERIMENTS ILLUSTRATION

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.

IV. REAL-WORLD DATASET

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 I.

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 I: 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 articulation parameters.

In Fig. 2, we show the 14 objects individually for more clarity.

V. RESULTS IN THE UMPNET ENVIRONMENT

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 environment (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 II and III, 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 [7]. The numbers here are consistent with those in the UMPNet paper.

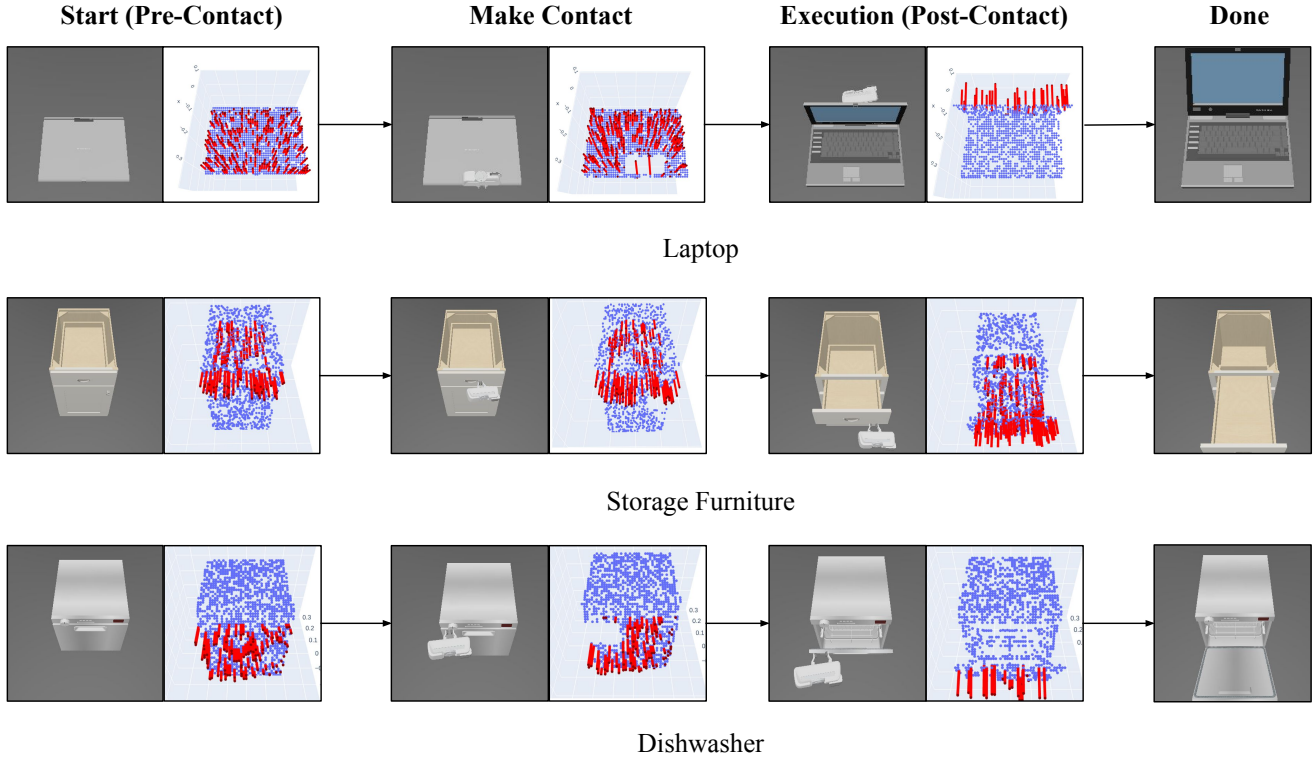


Fig. 1: Simulated rollout examples

	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 II: 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.

- 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.

VI. FULL TRIALS RESULTS

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






















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 III: 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.

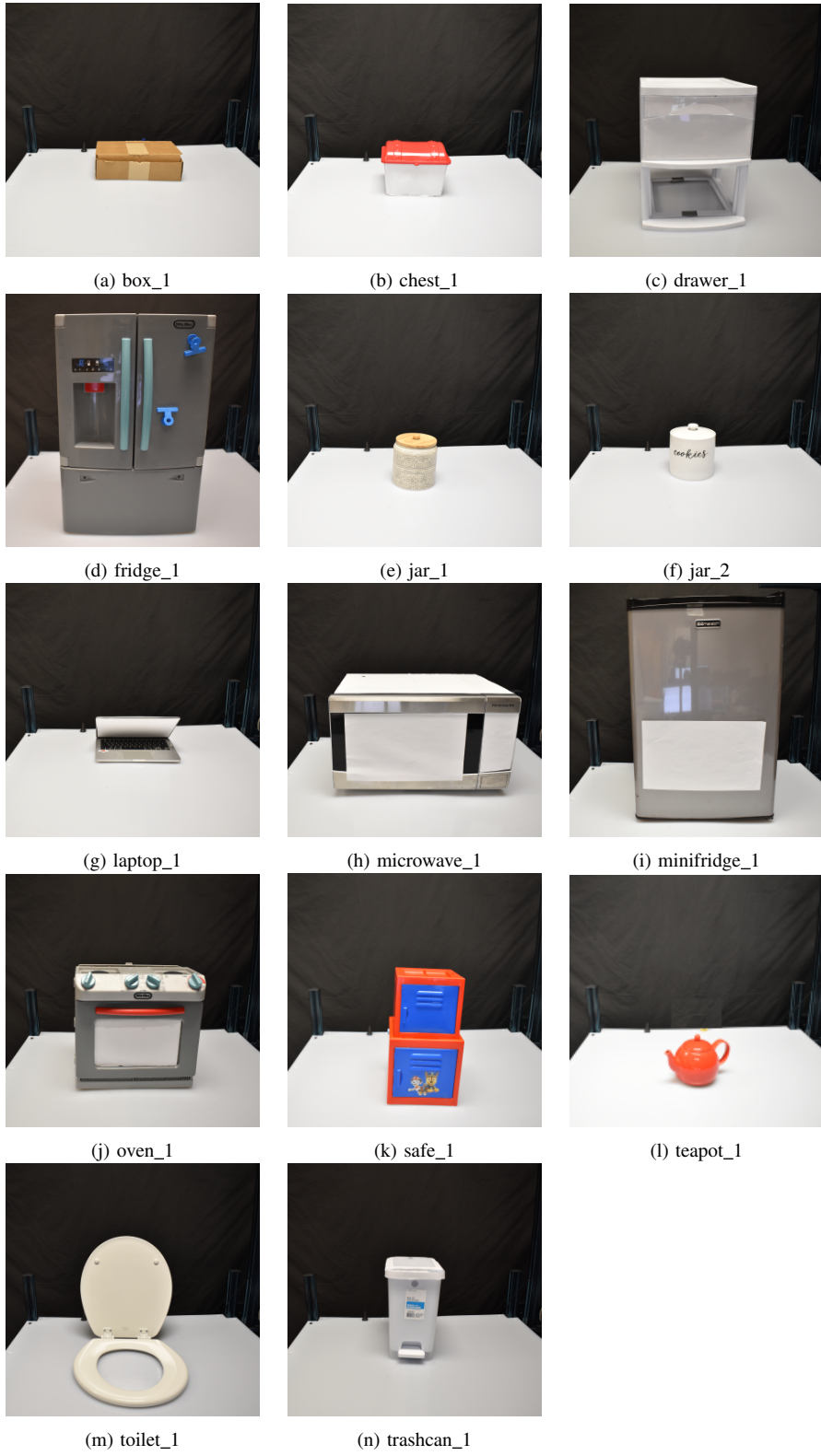


Fig. 2: Objects in the dataset for real world experiments

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 IV: 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 V: Real-World Trials for DAgger Oracle

REFERENCES

- [1] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [3] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.
- [4] 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.
- [5] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. June 2017.
- [6] 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.
- [7] Zhenjia Xu, He Zhanpeng, and Shuran Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022.