Interactive Anomaly Detection for Articulated Objects via Motion Anticipation - Supplementary

Ankan Bhunia **Changjian Li** **Hakan Bilen**
University of Edinburgh

A Appendix / supplemental material

In this supplementary material, we present details of the dataset creation pipeline in Appendix A.1, additional implementation details in Appendix A.2 and additional results in Appendix A.3. We provide **interaction videos** with this material to visualize the interaction steps, as well as the predicted and observed trajectories for various examples. These videos supplement Figures 1 and 5 in the main paper, along with additional examples.

A.1 PartNet-IAD Dataset Generation Details

Our proposed PartNet-IAD dataset is built upon the publicly available articulated 3D shape dataset, PartNet-Mobility [4] (MIT license). We provide the full list of categories and their splits in Tab. A1. Our PartNet-IAD dataset is licensed under Attribution-NonCommercial-ShareAlike 4.0 International.

Category selection. We use 11 training categories (Refrigerator, FoldingChair, Laptop, Stapler, TrashCan, Microwave, Toilet, Window, StorageFurniture, Switch, Kettle) and 10 testing categories (Box, Phone, Dishwasher, Safe, Oven, WashingMachine, Table, KitchenPot, Bucket, Door) from the PartNet-Mobility dataset, excluding objects that require multi-gripper collaboration (e.g., pliers, scissors) or are too small (e.g., pens, USB drives).

Anomaly types. We carefully design anomaly types tailored to each category based on object geometry and articulation patterns. We consider mainly four types of anomalies: 1) *Tilted axis of motion*: This anomaly involves rotating the axis of motion (for both revolute and prismatic joints) within a range of [0.1,0.4] radians along a randomly chosen axis (X, Y, or Z). To ensure the anomaly does not cause significant collisions with the main body, we evaluate 50 random articulation states within the originally specified range and retain the anomaly only if collisions occur in fewer than 25% of the states. Additionally, we manually create anomalies by altering the rotation axis to be orthogonal to its original axis. 2) *Different joint type*: This anomaly involves changing the joint type from revolute to prismatic or vice versa, depending on the feasibility of the object geometry. For instance, a drawer that rotates instead of sliding. We also introduce spherical joints to induce free rotation in 3 degrees of freedom (3DOF), creating anomalies where parts move unpredictably. 3) *Detachable Part*: For example, a cabinet door that falls off its hinges and can move without any constrained. To implement this, we modify the joint type to planar. This results in parts detaching from their joints and moving freely without constraints. 4) *Restricted motion*: This anomaly reduces the range of motion for both revolute and prismatic joints. We randomly select a smaller range than the original (less than 3/4 of the original range) to restrict movement. For example, a drawer that slides only partially open or a door that swings only slightly. For parts that do not move at all, we implement this by changing the joint type to fixed.

A total of 157 anomalous objects are created for the training categories, and 281 for the testing categories. The modified articulations are stored directly in URDF files written in XML format. We use Pybullet [1] to create simulation environment. Pybullet is released under the zlib license.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

A.2 Additional Implementation Details

Training details. We use PTv3 (MIT License) as our feature extractor. For input to the PTv3, we first normalize the point clouds and sample points with a voxel size of 0.02, ensuring at most one point per voxel. Voxels are serialized using various serialization methods [3], applied across layers as in PTv3. Both MLP heads consist of four layers. We train the model with a batch size of 64, where each object in the batch is associated with 256 sampled action-motion interaction data points for loss computation. To address the imbalance caused by the majority of action data not inducing motion, we apply data balancing to ensure each batch includes sufficient action-motion pairs involving part movement. For generalization, we apply data augmentations, including random rotation around the z-axis, flipping, jittering, chromatic auto-contrast, chromatic translation, and chromatic jitter. These augmentations prevent over-reliance on object pose and color, encouraging the model to leverage 3D geometric cues instead. We use the Adam optimizer with an initial learning rate of 5×10^{-4}, reducing it by a factor of 10 every 20 epochs. We use a single NVIDIA A40 GPU with 48 GB GPU memory to train the model. Training is performed for 50 epochs and took 2.5 days to complete. The loss parameters λ_M, λ_e, and λ_c are set to 1.0, 1.0, and 0.05, respectively, and remain fixed for all experiments. The standard deviation of all our experiments (including the ablations and our method) under multiple runs is less than 0.5.

Online data sampling: While offline data generation using random actions can partially explore the action space, it is often inefficient due to the large size of the action space. Focusing on actionable interactions (*i.e*. actions likely to induce motion) yields more informative training data. Online data sampling provides an efficient way of collecting actionable interaction data by leveraging the knowledge already acquired by the network. Specifically, we sample actions using the normalized movability score distribution in Eq. (8) and simulate additional interactions for those high-likelihood actions. We skip noise filtering in this step to sample actions faster, since this process is not sensitive to outliers. This online data collection strategy significantly increases the proportion of interaction pairs that induce positive motion.

Table A1: Statistics of the PartNet-IAD.

Training Categories					Testing Categories				
Name	# Normal	# Anomaly	Revolute	Prismatic	Name	# Normal	# Anomaly	Revolute	Prismatic
Fridge	9	12	×		Box	13	15	×	
FoldingChair	4	6	×		Phone	3	4	×	
Laptop	9	9	×		Dishwasher	41	42	×	×
Stapler	5	7	×		Safe	28	30	×	
TrashCan	10	15	×	×	Oven	24	27	×	
Microwave	2	8	×		WashingMachine	17	20	×	
Toilet	7	10	×		Table	77	78	×	×
Window	11	12	×	×	KitchenPot	25	26		×
StorageFurniture	60	64	×	×	Bucket	6	12	×	
Switch	2	4	×	×	Door	27	27	×	
Kettle	5	10	×	×					
Total	124	157			Total	261	281		

A.3 Additional Results

Additional real-world visualization of the motion prior network. Fig. A1 shows additional results of our normal motion estimation model (pretrained on the PartNet-Mobility dataset) applied to three categories from AKB-48: Bucket, Box, and Trashcan. As shown, our model performs favorably even on shapes with significant geometric differences and realistic textures, without any finetuning.

Robustness analysis of the motion prior network. Table 2 (main paper) quantitatively and Fig. A2 qualitatively show the impact of multi-step noise filtering on the motion estimation performance. We also conduct ablation studies on the test categories to assess the individual contributions of the confidence-based loss and noise filtering strategy to our IAD task, using the AUROC metric as shown in Tab. A2.

Failure cases of the motion prior network. While our motion prior is generally robust, it can yield incorrect predictions in the presence of ambiguity and self-occlusion. Fig. A3 illustrates a few failure cases: (a) the trashcan lid is predicted to open along an incorrect hinge axis; (b) the drawer

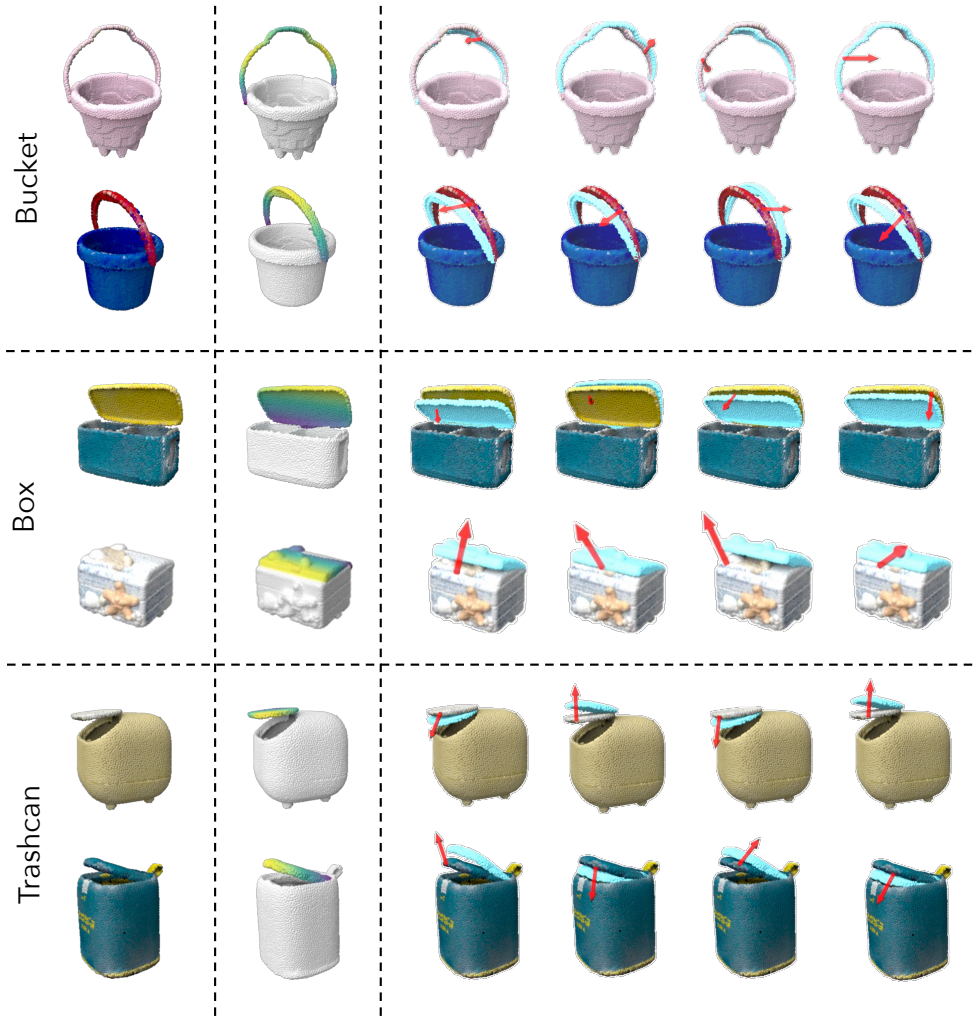

Figure A1: **Real-world visualization of the motion prior network on the AKB-48 [2] dataset.** Only normal objects are used for this visualization. For each row, we show the input point cloud, the movability score map, and four sampled action-motion pairs, respectively. The sampling of action-motion pair is performed based on the normalized scores in Eq. 8. Red arrows indicate the input actions, and the anticipated motions are shown in cyan.

Table A2: AUROC performance for noise filtering strategies.

Method	AUROC (%)
w/o filtering	81.4
w/ conf.-based noise filtering	82.7
w/ conf.-based + DBSCAN-based noise filtering (our method)	83.9

is predicted to rotate instead of translate; (c) the bucket handle is slightly misaligned, likely due to limited visibility; (d) the sliding window is misclassified as having a revolute joint; (e) the stapler's motion is kinematically correct but ignores collisions with adjacent parts, resulting in unrealistic overlapping; (f) the washing machine door's motion is incorrect due to occlusion of its hinge.

Training category set selection. Our chosen category split follows the same protocol as in [5]. We also conduct experiments with different combinations of training categories (Tab. A3). While selecting these combinations, we ensure that each combination contains a sufficient number of articulated objects covering representative articulation types, such as both revolute and prismatic joints. The *StorageFurniture* category is included in all sets, as it is the most representative category in the

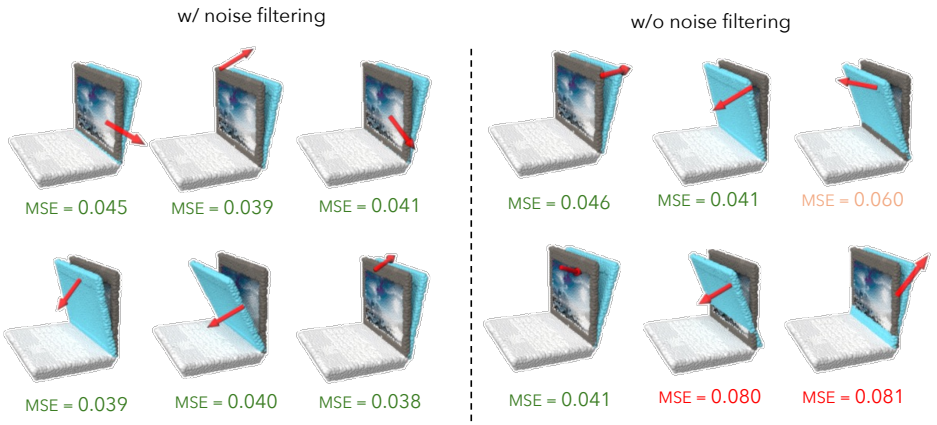

Figure A2: We compare six randomly sampled action-motion pairs with and without noise filtering. We compute the accuracy of the anticipated motions by MSE against the ground-truth motions. Higher MSE (indicated in red) means incorrect motion predictions. Red arrows indicate the input actions, and the anticipated motions are shown in cyan.

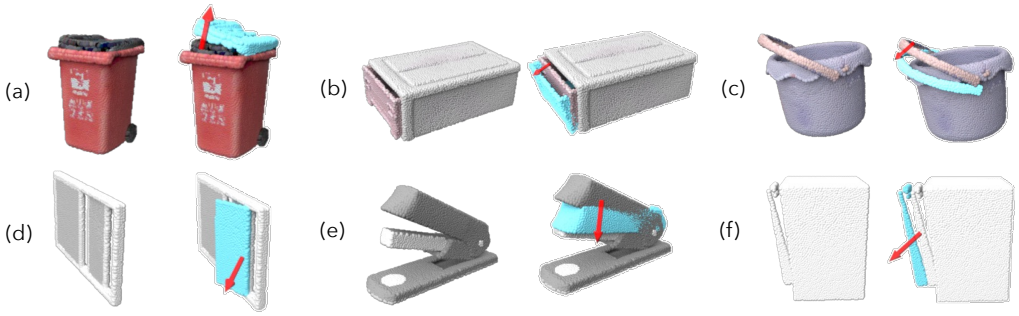

Figure A3: **Failure cases of the motion prior network.** Examples (a–c) are from the AKB-48 dataset, and (d–f) are from the PartNet-Mobility dataset. See Appendix A.3 for a detailed discussion of each case. Red arrows indicate the input actions, and the anticipated motions are shown in cyan.

PartNet-Mobility dataset. The test categories remain consistent across all experiments. As shown in the results below, our choice of training categories has a relatively low impact on overall performance, although some test categories may benefit from the presence of functionally similar categories in the training set (*e.g.* Microwave and Safe in SetA). Note that while the overall performance is slightly lower than in Table 1 in the main paper, this is expected since training was conducted on a smaller subset of the data.

Table A3: Performance of our method with different training category combinations.

	Training categories										
SetA	{StorageFurniture, Microwave, Switch, Stapler}	69.1	88.3	79.8	**84.1**	79.0	84.5	88.1	75.7	72.3	**73.9**
SetB	{StorageFurniture, Refrigerator, Laptop, Window}	71.4	91.5	**82.4**	81.1	**81.6**	85.8	**88.9**	80.5	73.4	72.1
SetC	{StorageFurniture, TrashCan, Toilet, Kettle}	**73.7**	**92.6**	80.2	81.0	80.3	**86.1**	88.4	**87.6**	**79.3**	72.3

References

[1] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

[2] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. Akb-48: A real-world articulated object knowledge base. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14809–14818, 2022.

[3] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024.

[4] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[5] Zhenjia Xu, Zhanpeng He, and Shuran Song. Universal manipulation policy network for articulated objects. *IEEE robotics and automation letters*, 7(2):2447–2454, 2022.