(a) (b)

Figure 5: (a) Real World Setup: In the image, we demonstrate the real-world experiment setup using a single RGB-D camera (Azure Kinect) to capture visual information and a Kinova robot equipped with a parallel jaw for interacting with an articulated shelf object.
(b) Shelf Object Point Cloud Illustration: We present the point cloud of the shelf object with three movable drawers, extracted from the RGB-D camera.

7 Supplementary

7.1 Real Robot experiments

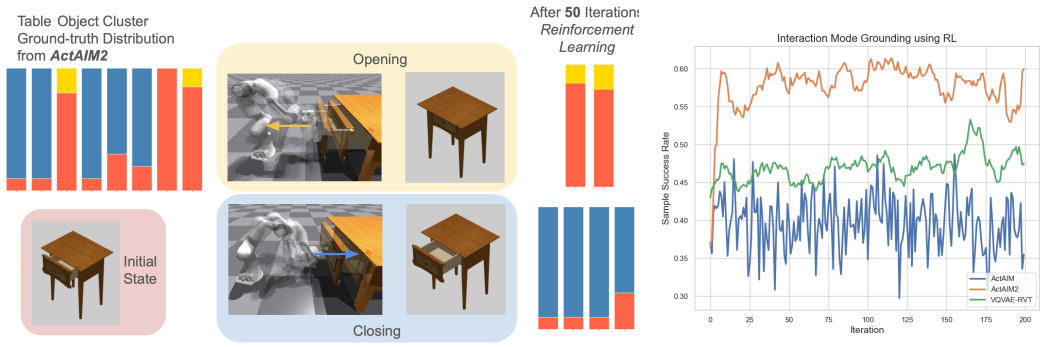
In this section, we evaluate ActAIM2 using real visual sensory data by training and testing the model in a real-world setup. For further insights into the model's performance, we have included attached videos, which are also available on our website.

Real World Setup – We employ a Kinova Gen2 robot arm with a spherical wrist, 7 degrees of freedom, and a parallel jaw gripper (Robot Type: j2s7s300), placed on a table alongside the articulated object shown in Figure 5a. The scene is captured using a statically mounted Azure Kinect (RGB-D) camera in a third-person view. We first calibrate the robot-camera extrinsics and transform the perceived color point clouds to the robot's base frame before processing them with ActAIM2. During evaluation, based on the target pose predicted by ActAIM2, we utilize the MoveIt motion planner to direct the robot to the target pose, incorporating trajectory generation and feedback control.

Tasks – We focus on manipulating a small shelf with three drawers showed in Figure 5b, defining two interaction modes for each half-open drawer: opening and closing. For the opening interaction, the gripper manually grasps the edge of the drawer and pulls it out. Conversely, for closing, the gripper grasps the edge and pushes the drawer in. We utilize a small shelf with three drawers as the articulated object and set three plausible initial states: the top, middle, and bottom drawers are each set to half-open. Task variation is introduced by altering the position of the small shelf on the table.

Data Collection and Training – Our evaluation focuses on a single task: opening and closing a half-open drawer on a small shelf. Initially, ActAIM2 is pre-trained using domain randomization on our simulated dataset. We currently employ ActAIM2 trained on the single-view dataset referenced in Table 4. For expert demonstration, we collect real-world interaction data via expert control to either open or close the drawer. The robot control strictly adheres to the simulator setup, utilizing four predefined action primitives: initializing, reaching, grasping, and interacting. To introduce variation in the training trajectories, each expert-controlled interaction randomly selects different parts of the drawer for the robot to grasp. Throughout these interactions, the articulated object remains stationary. Simultaneously, RGB-D images are captured as the robot moves to the target pose, creating a dataset of RGB-D frames paired with target pose annotations. For each initial state (top, middle, and bottom drawers half-open), we collect 10 expert data sets for each mode, totaling 60 trajectories (10 pushing and 10 pulling trajectories for each state).

Results – We initially trained ActAIM2 on simulated data and later fine-tuned it using real-world trajectories. This process yielded a notable success rate of 75% for both the pushing and pulling tasks involved in drawer interactions. Importantly, ActAIM2 accurately classifies the interaction modes of pushing and pulling, demonstrating a well-aligned discrete representation. To assess the performance,



(a) **(b)**

Figure 6: (a) Visualization of Reinforcement Learning on ActAIM2: ActAIM2's grounding through reinforcement learning is demonstrated, highlighting successful openings (yellow), closings (blue), and failures (red) across eight interaction clusters. After 50 iterations, clusters are accurately identified, enabling consistent trajectory generation.
(b) Reward Optimizing Plot: This plot displays reward optimization over iterations (x-axis) and average reward (y-axis, indicative of success rate for the targeted mode). ActAIM2's structured latent space shows clear advantages in sampling and convergence efficiency compared to ActAIM [12] and VQVAE-RVT.

we sampled from each of the 8 clusters designed to represent distinct interaction modes—either opening or closing the drawer. Clusters specific to pushing and pulling were found to have success rates exceeding 87.5%, mirroring the mode sampling evaluation. This improvement is largely due to incorporating human demonstration data, which helps correct any imbalance in interaction mode representation during training.

However, in our real robot experiments, we observed some failure cases where the gripper failed to grasp the edge of the drawer. These failures are linked to task variations, specifically the slight movement of the object along the x-axis, which led to the gripper failing due to the agent's incorrect determination of the object's position. To address this issue, we suggest implementing a more detailed coarse-to-fine tuning approach on the vision data and training with increased variation to improve accuracy in object positioning.

7.2 Interaction Mode Grounding with ActAIM2 using Reinforcement Learning

We further explore ActAIM2's capability as a prior in reinforcement learning to achieve specified interaction modes given a goal, iterating through all plausible interaction modes to effectively ground and distinguish each mode. In our setup, we used an unseen table (ID:20411) as the articulated object with a sparse reward function defined as $r = 1$ if $|d_i - d_g| < d_\epsilon$ else 0, where d_i is the selected ith degree of freedom (DoF), d_g is the target DoF, and d_ϵ is the threshold. Reinforcement learning was employed using ActAIM2 as the prior, framing the task as a Multi-Arm Bandit problem [49], which narrows the sampling space and enhances learning efficiency. For comparison, we used ActAIM and RVT-VQVAE as the prior in the baseline experiments, applying the same reward function but updating the sample task embedding via the cross-entropy method [50].

Qualitative and quantitative results are visualized in Figure 6a and Figure 6b. Figure 6a shows agents refining their understanding of interaction modes through repeated sampling, leading to self-supervised re-clustering. Figure 6b illustrates that using ActAIM2, the reward—equivalent to the average sample success rate—smoothly increases, whereas ActAIM struggles to ground the task embedding to a specific interaction mode representation, and VQVAE-RVT updates slowly due to its inability to provide distinct interaction mode representations.

7.3 Assumptions and Future Research

7.3.1 Assumptions

During the data collection phase, ActAIM2 operates under the assumption that: 1) the manipulations are straightforward enough to be captured using a limited set of action primitives such as grasping, pushing, or pulling; 2) an interaction mode is identified upon observing significant visual changes; 3) the interaction modes can be categorized into a few distinct types. A more detailed discussion of these assumptions is provided below.

Simple Action Space – We employ a scripted, self-supervised method to collect actions that encompass diverse interaction modes. The action space is sufficiently simple, focusing primarily on heuristic grasping and random actions. For more complex tasks, such as hammering, washing dishes, or cooking, our current method fails to collect adequate data. Addressing these more intricate tasks would require a more comprehensive and extensive dataset.

Significant Visual Change – Our data collection is entirely self-supervised, devoid of any expert data or privileged information. We define an interaction as successful if it results in a significant visual alteration to the targeted objects. This approach is effective for articulated objects in our studies, such as doors, windows, or tables, which typically remain stationary except for their movable components. However, challenges arise with objects like tools (e.g., hammers, cups, knives), where it is difficult to discern visual changes either in the tools themselves or the targeted objects (e.g., nails, cup holders, or deformable objects). Especially in tasks requiring repetitive actions, like continuously striking a nail or repeatedly wiping dishes, a more nuanced and generalized method is necessary to determine if meaningful interactions are occurring.

Discrete Interaction Modes – Articulated objects, by design, often have limited manipulation options. However, when dealing with other objects such as tools, the number of potential interaction modes significantly increases. The functionality of these objects can be diverse; for example, a hammer might be used not only for hammering but also for hooking or reaching. Even the act of grasping these objects presents countless variations, complicating the task of clustering them into discrete modes.

7.3.2 Future Research

Based on the assumptions discussed earlier, we have identified two primary avenues for extending our current research: long-horizon planning tasks and enhancing tool manipulation strategies.

Long-horizon Planning Tasks – Leveraging the discrete representation of interaction modes provided by ActAIM2, we propose its application to long-horizon planning tasks. Examples of such tasks include sequentially opening a table drawer, locating and opening a box within the drawer, and finally pressing a button inside the box. These tasks illustrate the potential of ActAIM2 to serve as a foundational prior, streamlining the process to discrete searches within complex sequences. To ensure the robustness of our approach, it is crucial that the model accurately predicts all feasible interaction modes based on the given scenario.

Extension to Tool Manipulation Tasks – Another direction for expansion involves applying our work to tool manipulation. Here, defining the interaction modes for various tools will be pivotal. A robust dataset specifically tailored for tool manipulation is essential to support this endeavour. Additionally, a more sophisticated scene descriptor is required to effectively determine which objects to manipulate and which to designate as targets. This development would facilitate more nuanced and effective tool interactions in automated systems.

7.4 Dataset Generation

7.4.1 Iterative Data Collection Method

When collecting data, we employ a strategy of random sampling, subsequently filtering successful actions as determined by our vision model without resorting to any privileged information. Drawing inspiration from [43, 51], we delineate the task of manipulating articulated objects into four fundamental poses: initiation, reaching, grasping, and manipulating. Throughout these stages, we capture the robot's key action poses $a_i = (\mathbf{p}, \mathbf{R}, \mathbf{q})_i$ and RGBD observations O_i from a configuration of five cameras encircling the articulated object. Upon collecting the trajectory $T_j = \{(a_i, O_i)|i = 0, 1, 2, 3\}_j$, we also archive the initial and final observations, O_j^{init} and O_j^{final}, respectively, captured from the multi-view cameras with the robot occluded, to facilitate manipulation success evaluation.

We introduced our method of identifying successful interacted trajectories, which can be purely from vision data, specifically the initial and final observation. For each trajectory, characterized by the initial observation O_j^{init} and final observation O_j^{final}, we utilize a pre-trained image encoder $\mathcal{E}_O$ to transform the image observations into a latent vector v. The task embedding z_j for each trajectory T_j is defined as follows:

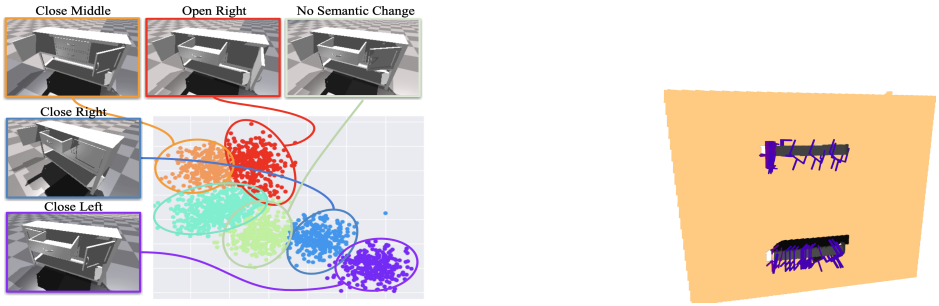
$$z_j = v_j^{init} - v_j^{final} = \mathcal{E}_O(O_j^{init}) - \mathcal{E}_O(O_j^{final}) \tag{9}$$

In our implementation, we employ a pre-trained VGG-19 network [52], without the final fully connected layers, to serve as our image encoder $\mathcal{E}_O$. To determine the success or failure of a manipulation, we introduce a threshold $\bar{z}$, defining a trajectory T_j as successful if $z_j > \bar{z}$. It is important to note that this process does not rely on any privileged information. To illustrate the validity of our method, we define the trajectory's success as a 30% change in the ground-truth DoF value. The efficacy of this criterion is validated against the ground-truth DoF values, demonstrating a 97.4% accuracy rate across our training and testing dataset. The collected trajectories must exhibit the diversity of interaction modes of the articulated objects. Thus, we employ three distinct methods of action sampling, as outlined below. The final dataset is a composite of these three methods.

1. Random Sampling – We generate play data for manipulation without prior interaction. First, we select an interaction point $p_1 \in \mathbb{R}^3$ on the articulated object, ensuring it lies within the robot's workspace. Subsequently, we sample a uniformly random manipulation rotation $\mathbf{R}_0 \in SO(3)$ and a manipulation position p_2 within the valid area, applying filters to exclude any configurations that would result in a collision. The robot's initial position p_0 is also determined through random sampling, which is a specified distance from the interaction point p_1, ensuring a feasible starting position for the manipulation task. Based on the previous sampling, we define the randomly sampled action sequence as $\{(p_0, \mathbf{R}_0, 0), (p_1, \mathbf{R}_0, 0), (p_1, \mathbf{R}_0, 1), (p_2, \mathbf{R}_0, 1)\}$.

2. Heuristic Grasping Sampling – Heuristic grasping sampling is employed to select interaction points on the articulated object to enhance the precision of grasping actions. Utilizing the RGB-D observations, we crop the articulated object and transform it into an RGB point cloud, which undergoes preprocessing with DBSCAN clustering [53], aimed at identifying segments with significant geometric features, such as handles or buttons. After clustering, each segment is analyzed by a pre-trained GraspNet model [54] to generate a set of potential grasps. From this set, grasps with the highest scores are selected, with the grasp point designated as the interaction point and the grasp orientation as the gripper rotation for the trajectory. The initial and manipulation poses are determined using the previously described random sampling approach. This heuristic approach to grasping not only bolsters the stability of grasp actions but also enriches the dataset with a higher proportion of complex interaction modes, such as "grasp to open", enhancing the dataset's diversity and utility for training models to manipulate articulated objects in 'hard' interaction scenarios.

3. GMM-based Adaptive Sampling – To foster a wide array of interaction modes within our dataset, we implement GMM-based adaptive sampling inspired by the methodology outlined in [12]. Following the acquisition of M trajectory datasets $\{T_j|j = 1, 2, ..., M\}$ through random and heuristic grasping sampling from previous interactions, we compute the task embeddings $\{z_j|j = 1, 2, ..., M\}$ based on Equation 9. A Gaussian Mixture Model (GMM) prior is constructed from these task



(a) **(b)**

Figure 7: (a) GMM clustering adaptive sampling: his figure illustrates the visualization of using GMM to represent different interaction modes.
(b) Visualization of Heuristic Grasping: We illustrate the proposed grasping using our predefined heuristic with ContactGraspNet [55].

embeddings, denoted as $\mathbb{P}(z|\theta) = \sum_{k=1}^{K} \kappa_k p(a|\theta_k)$, where θ_k represents the parameters of each Gaussian component within the mixture. The choice of K, the number of clusters, is a hyper-parameter that reflects the presumed number of interaction modes inherent to the object.

Subsequently, we cluster the task embeddings z_j, assigning a unique cluster label to each corresponding trajectory. We found that the task embeddings from different trajectories grouped within the same cluster indicate a similar interaction mode, as they share proximate visual characteristics from initial and final observation. Upon clustering, a new GMM is formulated for each cluster, based on the action sequences, represented as $\mathbb{P}_k(a|\phi) = \sum_{l=1}^{L} \beta_l p_k(a|\phi_l)$. We then aim to sample an equal number of actions from each cluster, ensuring that the representation of actions—and, by extension, interaction modes—within the dataset are as diverse as possible, thus facilitating a comprehensive exploration of the articulated object's potential interactions.

Utilizing these sampling methodologies, we concurrently collect data across all articulated objects within our dataset, culminating in a dataset denoted as:

$$D = \{T_j\}_{\text{random}} \cup \{T_j\}_{\text{grasp}} \cup \{T_j\}_{\text{GMM}} \tag{10}$$
$$= \{(a_i, O_i)_j\}_{\text{random}} \cup \{(a_i, O_i)_j\}_{\text{grasp}} \cup \{(a_i, O_i)_j\}_{\text{GMM}} \tag{11}$$
$$= \{(O_i, a_i)\}_{\text{random} \cup \text{grasp} \cup \text{GMM}} \tag{12}$$

After data collection, we enrich each trajectory within our dataset by associating the respective task embedding with the data tuple (O, a), thereby forming atomic training data instances represented as $(O, a, \epsilon)_j$.

7.4.2 Data Collection Algorithm

The dataset we developed for training purposes is available on our official website. Our dataset was constructed through a combination of random sampling, heuristic grasp sampling, and Gaussian Mixture Model (GMM)-based adaptive sampling, featuring the Franka Emika robot engaging with various articulated objects across multiple interaction modes. It encompasses categories such as faucets, tables, storage furniture, doors, refrigerators, and switches, with 8 unique instances per category. For each instance, we collected 150 trajectories, ensuring comprehensive coverage of the objects' interaction modes. Objects were scaled to realistic size and initialized in a 'half-open' state, denoting a median value for each degree of freedom (DoF). The data collection methodology is detailed in Algorithm 1.

7.5 Model Architecture and Implementation Details

This section outlines the detailed implementation of the model architecture, encompassing both the mode selector and the action predictor components.

Algorithm 1 Data Collection Algorithm

Require: Initial observation O^i, Number of GMM component K, hyper-paramter M for GMM in each cluster
Ensure: All sampled trajectories are filtered successful by evaluating $\epsilon > \bar{\epsilon}$

```
D ← ∅                                                   ▷ Set the initial dataset to be empty
while D not have enough data do
    D_r = {(a, o)_i} ∼ RandomSampling                   ▷ Random Sampling
    G = {g_i} ∼ GraspNet(O^i)                           ▷ Sample Grasp using GraspNet
    D_g = {(a, o)_i} ∼ GenerateTraj(G)                  ▷ Gnerate trajectory based on grasp
    D ← D ∪ D_r ∪ D_g
    ε_i ∼ D                                             ▷ Compute task embedding in current D
    Cluster ε_i with GMM, assign cluster label on each trajectory
    {D_j | j = 1, ..., K} ← D
    D_GMM ← ∅
    for j in range K do
        Extract D_j in D based on cluster label
        p(D_j | π, μ, Σ) = ∏_{n=1}^{N} ( �sum_{m=1}^{M} π_m N(x_n | μ_m, Σ_m) )   ▷ fit GMM
        D̂_j ← {(a, o)_i} ∼ p(D_j | π, μ, Σ)             ▷ Sample action from GMM
        D_GMM ← D_GMM ∪ D̂_j
    D ← D ∪ D_GMM
```

7.5.1 Mode Selector Architecture and Implementation Detail

This section revisits the stochastic variables' definitions and distributions, as previously emphasized. The distributions of the model parameters are formalized as follows:

$$p(c) = \text{Multi}(\pi) \tag{13}$$

$$p(y) = \mathcal{N}(0, \mathbf{I}) \tag{14}$$

$$p_{\xi,\beta}(\epsilon, x, y, c|O^i) = p(y)p(c)p_\xi(x|y, c, O^i)p_\beta(\epsilon|x, O^i) \tag{15}$$

$$p_\xi(x|y, c, O^i) = \prod_{k=1}^{K} \mathcal{N}(\mu_{c_k}(y, O^i), \Sigma_{c_k}(y, O^i)) \tag{16}$$

$$p_\beta(\epsilon|x, O^i) = \mathcal{N}(\mu_\beta(x, O^i), \Sigma_\beta(x, O^i)) \tag{17}$$

Here, $\mu_{c_k}, \Sigma_{c_k}, \mu_\beta, \Sigma_\beta$ are the model parameters to be optimized. Furthermore, we delineate the generative model and compute the inference at test time by defining the posterior as follows:

$$q(x, y, c|\epsilon, O^i) = \prod_i q_{\psi_x}(x|\epsilon, O^i)q_{\psi_y}(y|\epsilon, O^i)q_{\psi_c}(c|x, y, O^i) \tag{18}$$

This necessitates the computation of three additional network parameters: $q_{\psi_x}, q_{\psi_y}, q_{\psi_c}$. We then elaborate on deriving the posterior $q_{\psi_c}(c|x, y, O^i)$ for categorical variables c, employing the Gumbel Softmax for the representation of categorical distributions.

Notice that c is a categorical parameter that $c \sim \text{Multi}(\pi)$. We defined that $c \in \mathcal{C} = \{c_1, c_2, ..., c_k\}$ and the each class probability is described as $\{\pi_1, \pi_2, ..., \pi_k\}$. We use the Gumbel Softax trick which provides a simple and efficient way to draw samples c from a categorical distribution with class probabilities $\{\pi_1, \pi_2, ..., \pi_k\}$. The following form represents the categorical c as,

$$c = \text{one-hot}(\text{argmax}_i[g_i + \log \pi_i]) \tag{19}$$

where $\{g_1, g_2, ..., g_k\}$ are i.i.d samples drawn from Gumbel(0,1). Assuming that categorical samples c are encoded as k-dimensional one-hot vectors ω lying on the corners of the $(k-1)$-dimensional simplex Δ^{k-1} We use the softmax function as a continuous, differentiable approximation to arg max, and generate k-dimensional sample vectors $\omega \in \Delta^{k-1}$. We defined ω as

$$\omega_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{i=1}^{k} \exp((\log(\pi_i) + g_i)/\tau)} \tag{20}$$

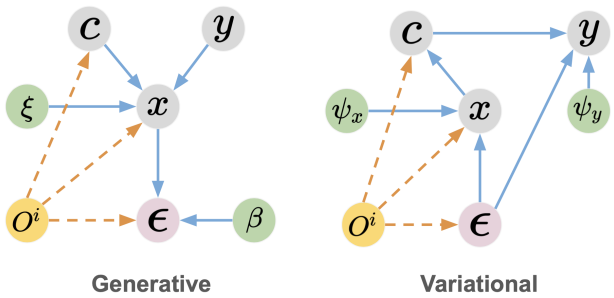


Figure 8: The graphical representations elucidate the Conditional Gaussian Mixture Variational Autoencoder (CGMVAE) framework, showcasing two distinct models: the generative model on the left and the variational family on the right. These graphical models serve to visually communicate the structural and functional relationships between variables within the CGMVAE, illustrating the data generation process and the approximation strategy employed by the variational family to infer latent variable distributions.

Where τ is the temperature as the hyperparameter. Therefore, we define the density of the Gumbel-Softmax distribution as,

$$p(c) = p_{\pi,\tau}(\omega_1, ..., \omega_k) = \Gamma(k)\tau^{k-1}\left(\sum_{i=1}^{k}\frac{\pi_i}{\omega_i^{\tau}}\right)^{-k}\prod_{i=1}^{k}\frac{\pi_i}{\omega_i^{\tau}} \tag{21}$$

Now, given the representation of the categorical distribution of c from Equation 21, we derive how we compute the posterior q_{ψ_c} for c. We consider the posterior $q_{\psi_c}(c = c_j|x, y, O^i)$ given $c = c_j$,

$$q_{\psi_c}(c = c_j|x, y, O^i) = \frac{p(c = c_j)p(x|c = c_j, y, O^i)}{\sum_{l=1}^{k} p(c = c_l)p(x|c = c_l, y, O^i)} \tag{22}$$

$$= \frac{\pi_j p(x|c = c_j, y, O^i)}{\sum_{l=1}^{k} \pi_l p(x|c = c_l, y, O^i)} \tag{23}$$

Therefore, we derive the posterior q_{ψ_c} directly and leave 2 posterior network q_{ψ_x}, q_{ψ_y} to be trained.

Based on the following discussion, we draw the generative model and variational model view as graphical models in the Figure 8.

In the implementation detail, we write parameters $p_\beta = (\mu_\beta, \Sigma_\beta)$ and $p_\xi = (\mu_{c_k}, \Sigma_{c_k})$ to generate a Gaussian distribution with each representing the mean and variance. We implement the network $\mu_{c_k}, \Sigma_{c_k}, \psi_x, \psi_y$ with a multi-layer ResNet and implement the network μ_β, Σ_β as a multi-view transformer since both O^i and ϵ represent multi-view information with the same number on the channel as the correspondent view number. We show our model μ_β, Σ_β architecture in Figure 9.

7.5.2 Mode Selector Training and Inference

We illustrate the functionality and application of our mode selector through two distinct plots, highlighting both the training process and the inference mechanism for task embedding generation.

Figure Figure 11a depicts the model's operation during training, where it processes the conditional variable O^i along with the ground truth data ϵ, to accurately reconstruct the task embedding.

Conversely, Figure Figure 11b demonstrates the inference stage, where the model, requiring only the initial observation O^i and a discretely sampled cluster (employing an 8-cluster configuration for implementation), successfully generates the corresponding task embedding ϵ.

7.5.3 Action Predictor

We provide the architecture of the action predictor which is a joint transformer that takes in task embedding ϵ and novel view as input. The detailed implementation is shown at Figure 10.

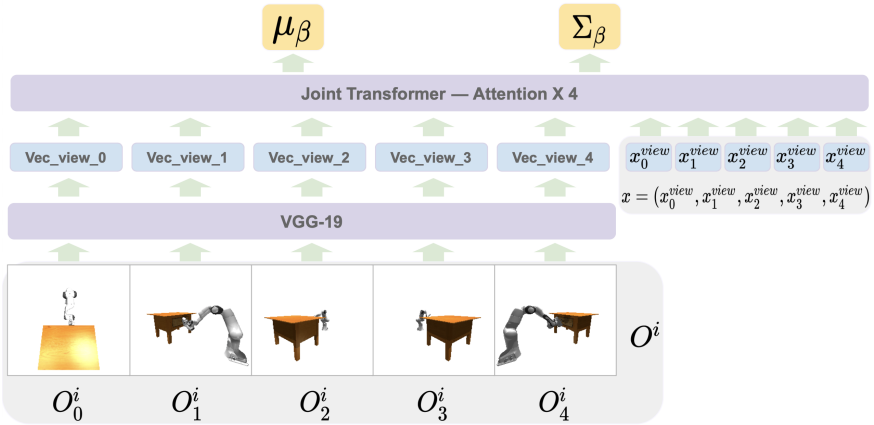


Figure 9: Mode Selector Decoder Architecture: The depicted architecture highlights the functionality of the mode selector decoder, which is designed to process two primary inputs: multi-view RGBD images $O^i = (O_0^i, O_1^i, O_2^i, O_3^i, O_4^i)$, and the Mixture of Gaussian (GMM) variable x. It is important to note that x can be represented as a multi-view feature vector, with our encoding approach preserving the separation of multi-view channels. Initially, the multi-view RGBD images are passed through a pre-trained VGG-19 image encoder to extract feature vectors for each view. Subsequently, these feature vectors, along with the GMM variable x, are inputted into a joint transformer. This transformer, featuring four attention layers, is tasked with producing the means and variances associated with the reconstructed task embedding ϵ.

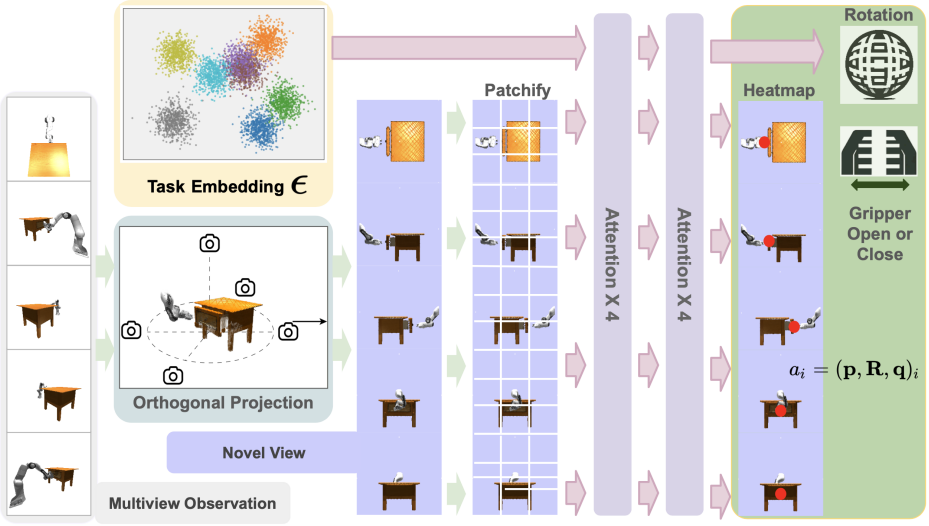
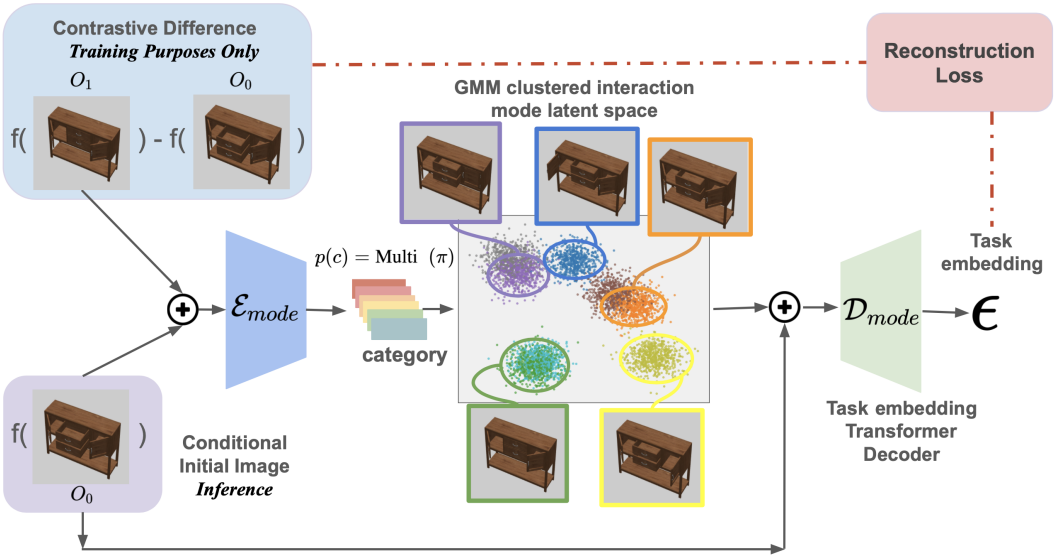


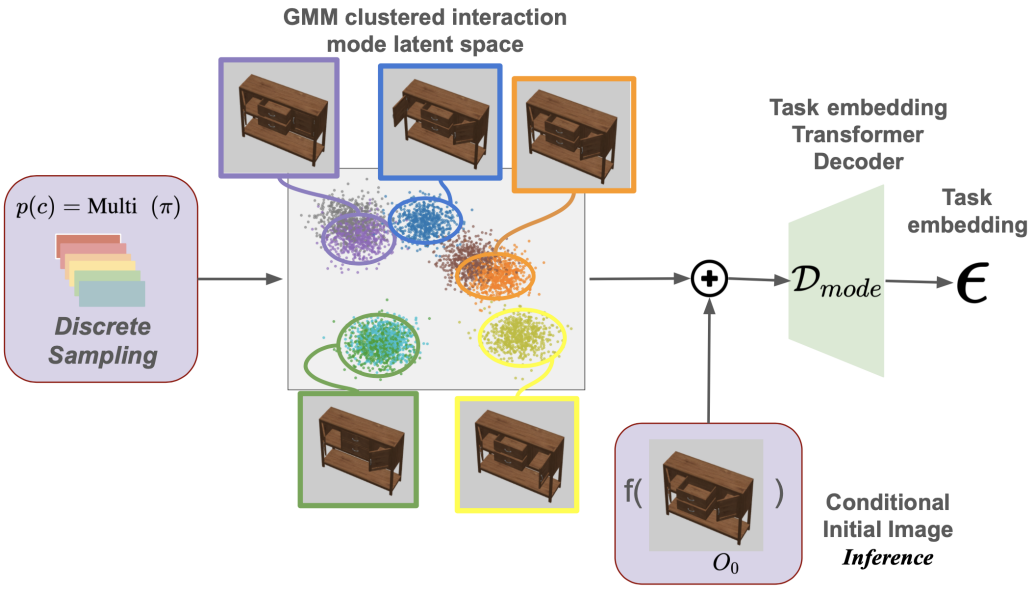
Figure 10: Action Predictor Architecture: This model integrates multi-view observations directly as input, sourced from predefined cameras within the scene. The process begins with the extraction of five RGBD images, which are subsequently transformed into RGB point clouds. These are then subject to orthogonal projection to generate five novel view images. Subsequently, these novel views are partitioned into smaller patches and fed into a joint transformer. This transformer, characterized by four attention layers, integrates the sampled task embedding derived from a Mixture of Gaussian distribution. The architecture of the joint transformer encompasses eight attention layers, culminating in the production of a heatmap. This heatmap delineates the action's translation, the discretized rotation, and a binary variable indicating the gripper's state—open or closed.

7.6 More Qualitative Results

We supplement our presentation with additional qualitative results, further elucidating the model's proficiency in learning the disentanglement of interaction modes. Initially, we demonstrate the efficacy of the mode selector through a t-SNE plot. This choice of visualization is motivated by our methodology of training the mode selector and action predictor independently, allowing for a focused examination of the mode selector's performance.



(a) Training Process of the Mode Selector: This figure illustrates the training procedure of the mode selector, mirroring the approach of a conditional generative model. It highlights the contrastive analysis between the initial and final observations—the latter serving as the ground truth for task embedding—to delineate generated data against the backdrop of encoded initial images as the conditional variable. The process involves inputting both the generated task embedding data and the conditional variable into a 4-layer Residual network-based mode encoder, which then predicts the categorical variable c. Following the Gaussian Mixture Variational Autoencoder (GMVAE) methodology, the Gaussian Mixture Model (GMM) variable x is computed and introduced alongside the conditional variable to the task embedding transformer decoder. This model is tasked with predicting the reconstructed task embedding, sampled from the Gaussian distribution as outlined in the architecture of the mode selector decoder, and calculating the reconstruction loss against the input ground truth data.



(b) Inference Process: In the inference phase, the agent discretely samples a cluster from the trained Gaussian Mixture Variational Autoencoder (GMVAE) model to calculate the Mixture of Gaussian variable x. This variable x, in conjunction with the conditional variable (initial image observation), is then inputted into the mode selector transformer decoder. The objective is to reconstruct the task embedding for inference, effectively translating the conditional information and sampled cluster into actionable embeddings.

Subsequently, we extend our qualitative analysis with figures akin to those presented in the main paper, offering a comprehensive view of the model's capabilities. These additional figures serve to reinforce the insights gained from the initial results, showcasing the model's nuanced understanding of interaction modes through the distinct visual representations of the data.

7.6.1 Mode selector TSNE plot Figure 16

Utilizing our pre-trained Conditional Gaussian Mixture Variational Autoencoder (CGMVAE) mode selector, we conduct disentanglement learning visualization on our comprehensive dataset. Specifically, we focus on the "single drawer" object (object ID: 20411), employing the mode selector to delineate the generated clusters and compare them with the ground truth task embeddings. The data for this visualization is derived from our dataset, and we calculate the task embedding ϵ_j for each data point as the difference between the initial and final object states, represented by

$$\epsilon_j = v_j^{init} - v_j^{final} = \mathcal{E}_O(O_j^{init}) - \mathcal{E}_O(O_j^{final})$$

.

Subsequently, we employ a t-SNE plot to simultaneously visualize the ground truth and generated task embeddings. In this visualization, distinct colors within the ground truth plot indicate data points originating from different interaction modes. Similarly, varied colors in the generated plot correspond to data points arising from disparate clusters within the Mixture of Gaussians model. Through this approach, we demonstrate that:

1. The ground truth task embeddings ϵ are distinctly clustered based on the interaction modes.
2. The CGMVAE model effectively generates clusters that categorize data points by their respective categories c.
3. The reconstructed data closely aligns with the ground truth data points, with the majority of the clustered data encompassed within the respective ground truth clusters.

This visualization underscores the efficacy of our generative model mode selector in extracting task embeddings for further application in the action predictor, highlighting the model's capability to discern and categorize interaction modes accurately.

7.6.2 Action Predictor Qualitative Results

We present extensive qualitative results in Figure 17a, Figure 17b, Figure 18a, and Figure 18b, demonstrating the model's ability to predict distinct interaction modes through discrete sampling. For each object, we explore three different clusters, each representing a unique interaction mode. The initial state of the robot and the articulated object is depicted from three perspectives: top-down, front, and side views. The heatmaps, derived from the top view during manipulation steps, highlight the variance in action space corresponding to different sampled interaction modes. Subsequent imagery illustrates the robot's movement within the simulator and the outcome following interaction with the articulated objects. It is important to note that comprehensive **video demonstrations** accompany this document and are accessible on our website, https://actaim2.github.io/.

7.6.3 Comparison of ActAIM2 and VQVAE-RVT

Inspired by the Genie [48] approach, we have compared our ActAIM2 with VQVAE-RVT to assess the efficacy of these models in discerning discrete interaction modes in robotic manipulation tasks. Our primary objective was to evaluate the distinction between interaction modes using a simplified scenario, a single-drawer table, which naturally exhibits two distinct interaction modes: opening and closing.

In our experiments, we visualized the latent spaces generated by both ActAIM2 and VQVAe-RVT. Particularly for VQVAE-RVT, the latent space visualization involved examining the distribution of eight code vectors. As depicted in Figure 12, these vectors clustered into two categories, which ideally should correspond to the two expected interaction modes of the drawer. This clustering pattern was anticipated and desired as it suggests a clear demarcation between the distinct modes of interaction.

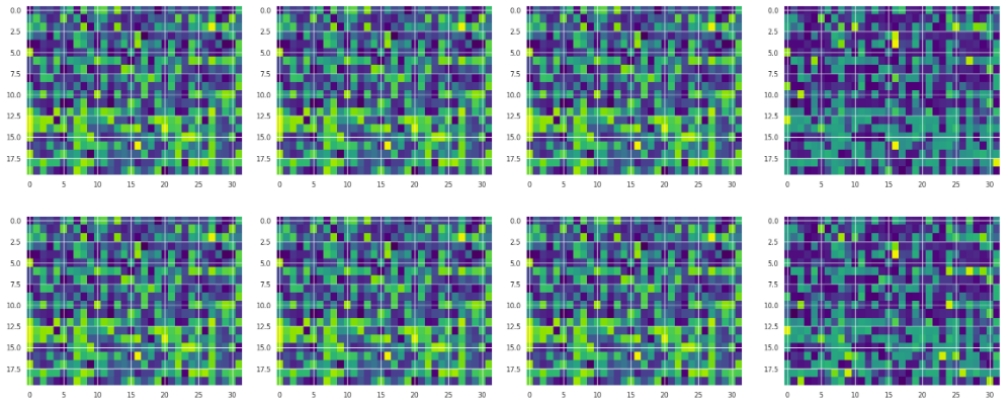


Figure 12: Visualization of Latent Space Clustering in VQVAE-RVT: This figure illustrates the distribution of eight code vectors within the latent space, categorized into two distinct clusters. These clusters are intended to represent the discrete interaction modes of opening and closing a drawer. The spatial arrangement highlights the expected separation of code vectors, symbolizing the potential for mode-specific action mapping in robotic manipulation tasks. Despite this apparent clustering, subsequent heatmaps (see Figure 13) reveal a lack of diversity in the action predictions, undermining the practical utility of this model configuration.

However, subsequent visualizations raised concerns about the practical efficacy of the VQVAE-RVT model in our application context. When we explored the heatmaps generated by the VQVAE-RVT model, we observed a critical limitation: all 8 code vectors produced essentially the same heatmap, despite their differing positions in the latent space. This heatmap, illustrated in Figure [Y], consistently depicted all plausible interaction modes for the drawer, regardless of the specific code vector used. This outcome was in stark contrast to the results from ActAIM2, where distinct heatmaps clearly indicated specific interaction actions like pushing or pulling, depending on the sampled cluster within the latent space.

These findings led us to conclude that merely replacing the GMVAE component with a VQVAE in the setup did not achieve the desired disentanglement of interaction modes. The VQVAE-RVT model failed to map the code vectors to unique, mode-specific interaction strategies, instead converging on a generalized representation that was not useful for distinguishing between the actionable options of opening and closing the drawer. Consequently, ActAIM2's ability to discriminate between distinct interaction modes via cluster-specific sampling proves superior in contexts demanding discrete and distinguishable action representations.

We also explain this from mathematics which provide an intuition of why GMVAE perform much better than VQVAE. Referring to Figure 8 and the ELBO loss computation in Equation 18 from the appendix, compared to the VQVAE, the most unusual term in our ELBO is the c-prior term. The c-posterior calculates the probability of assigning a data point to a cluster by measuring the distance between x and each cluster position from y. The c-prior helps reduce the KL divergence between the c-posterior and a uniform prior by adjusting both the clusters' positions and the encoded point x. This term aims to merge clusters by increasing their overlap and bringing their means closer together. However, like other KL regularization terms, it can conflict with the reconstruction term and may become too dominant with more training data. This over-regularization issue, common in traditional VAEs, also affects GMVAE clusters, leading to large, degenerate clusters.

7.7 Generation of Demonstration Videos

To illustrate the practical applications and effectiveness of ActAIM2, we generated demonstration videos by employing its inference mechanism. The process involves several key steps:

1. **Generative Mode Selection**: Initially, observations are inputted into the generative mode selector of ActAIM2. This component is responsible for reconstructing the task's latent space, which is modeled as a Mixture of Gaussians. This structure enables discrete sampling of clusters, which represent distinct interaction modes that the robotic system can execute.

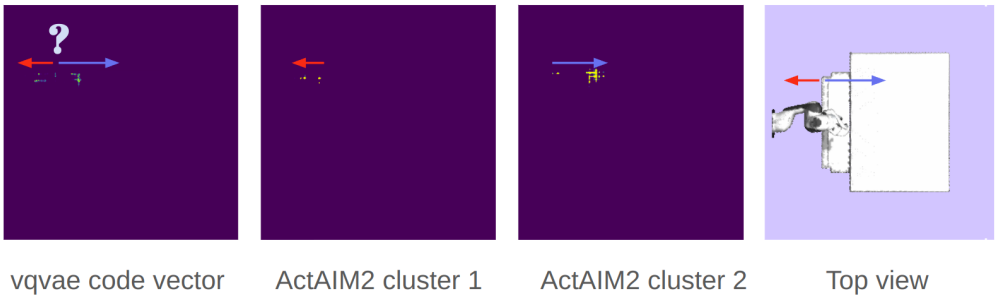


Figure 13: Comparative Visualization of Action Heatmaps and Observational Data From left to right: (1) VQVAE-RVT action heatmap synthesized using all eight code vectors, showing identical outcomes across the board, indicating a failure to differentiate interaction modes. (2) Action heatmap generated by \ActAIM2 when sampling from one cluster, demonstrating a specific interaction mode. (3) Action heatmap from \ActAIM2 when sampling from a different cluster, showcasing another distinct mode of interaction. (4) Top-view observation of the drawer, correlating with the spatial contexts of the heatmaps, providing a visual reference for the interaction zones mapped by the heatmaps. This series highlights \ActAIM2's capability to discern and represent distinct action strategies through targeted cluster sampling.

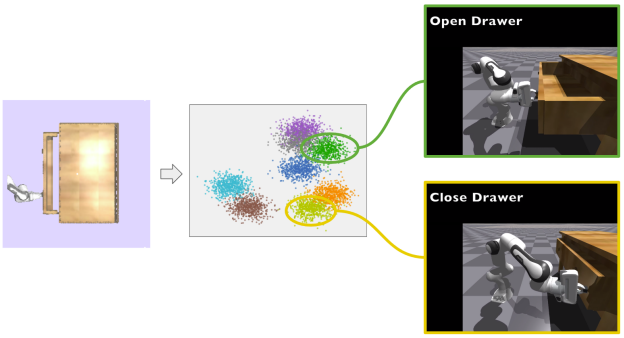


Figure 14: Opening and Closing a Drawer: This figure demonstrates the effective action sequence generated by ActAIM2 for a drawer. The left part of the image shows the drawer being opened, showcasing the robot's approach and grip adjustment. The right part of the image captures the drawer in a fully closed position, illustrating the final state after the action sequence execution.

2. **Sampling and Action Prediction**: From the reconstructed latent space, we sample the task embeddings by selecting a cluster within the Gaussian Mixture Model (GMM) and its corresponding Gaussian distribution. This sampled task embedding is then forwarded to the action predictor. The action predictor generates the specific actions needed to interact with the environment effectively.
3. **Simulation and Recording**: As depicted in Figure 14 and Figure 15, ActAIM2 reconstructs an object-based GMM and samples different task embeddings. Depending on the sampled task embedding, different interactions are reconstructed and executed within a simulator. We recorded the manipulation processes, which are detailed in the video provided in the supplementary files. Each video showcases how ActAIM2 navigates through different interaction scenarios, reflecting the diverse capabilities of the model in real-time applications.

This comprehensive demonstration not only validates the functionality of ActAIM2 but also provides a visual understanding of its potential in diverse robotic manipulation tasks. The videos highlight the nuanced interactions achievable through targeted sampling within the model's structured latent space.

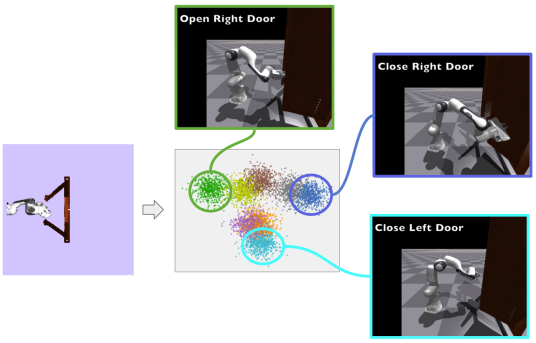


Figure 15: Opening and Closing a Door: This figure illustrates the ActAIM2's manipulation capability with a door. The left image displays the door being opened, highlighting the robot's positioning and the initial interaction phase. The right image shows the door completely closed, detailing the end of the manipulation sequence and the effectiveness of the action predictor.

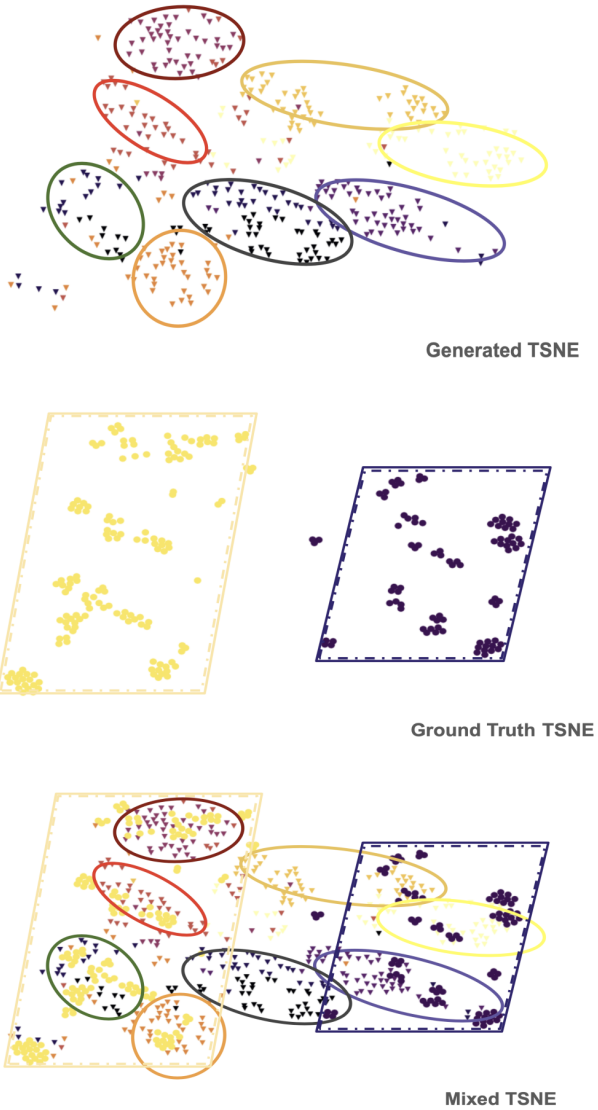
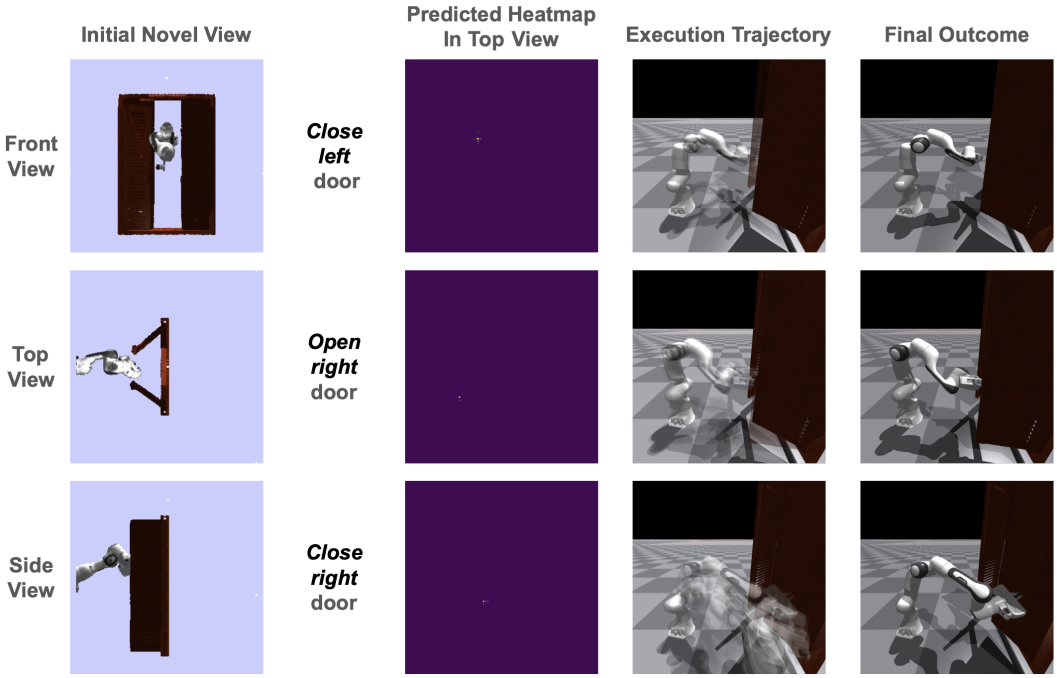
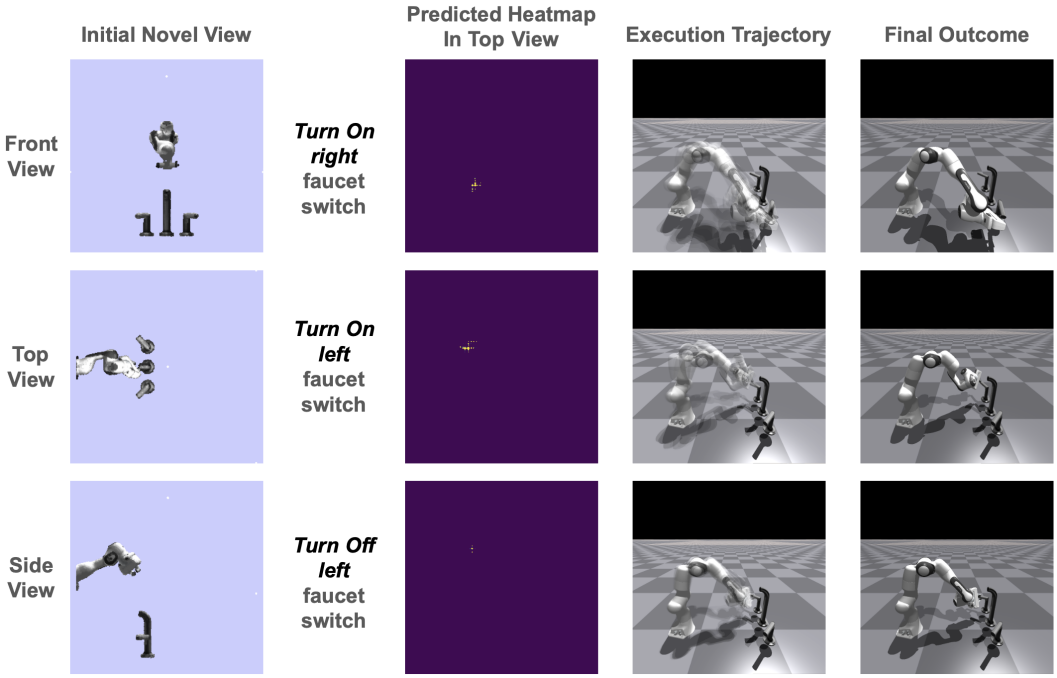


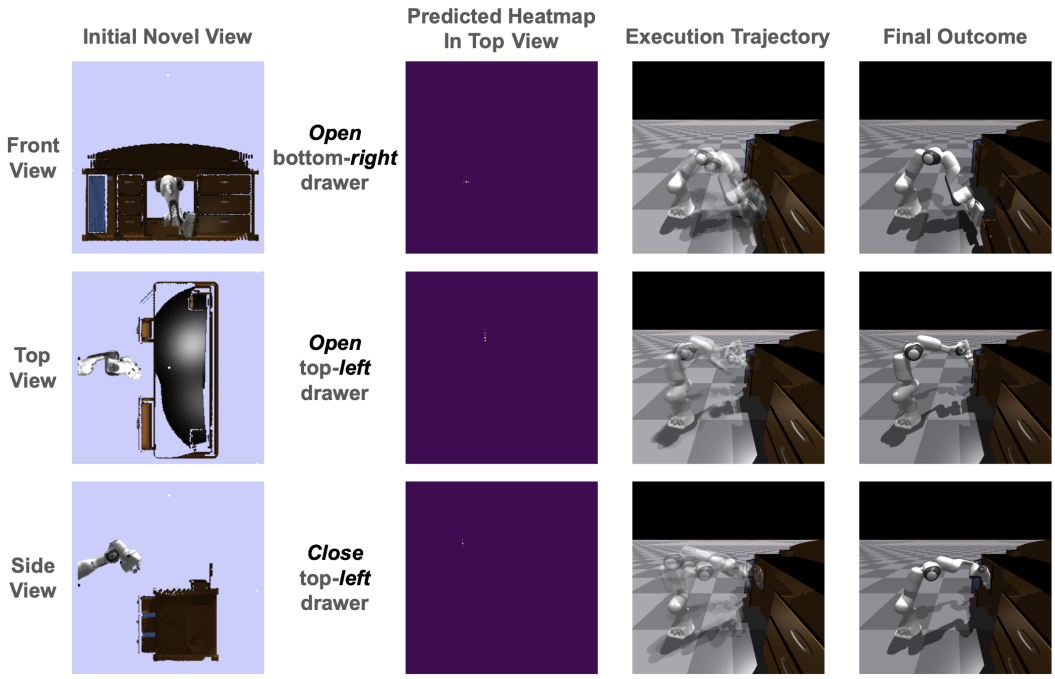
Figure 16: Disentanglement Visualization with CGMVAE: This figure illustrates the efficacy of the Conditional Gaussian Mixture Variational Autoencoder (CGMVAE) in disentangling interaction modes for the "single drawer" object (ID: 20411), using a t-SNE plot for visualization. Task embeddings ϵ_j, defined by the variance between initial and final object states, are visualized in distinct colors to denote various interaction modes and clusters. The sequence of figures demonstrates the CGMVAE's precision in clustering and aligning data points with their respective interaction modes: (1) Generated clusters from the CGMVAE mode selector reveal distinct groupings. (2) Ground truth task embeddings confirm the model's capacity for accurate interaction mode classification. (3) A combined visualization underscores the alignment between generated clusters and ground truth, showcasing the model's ability to consistently categorize tasks within identical interaction modes.



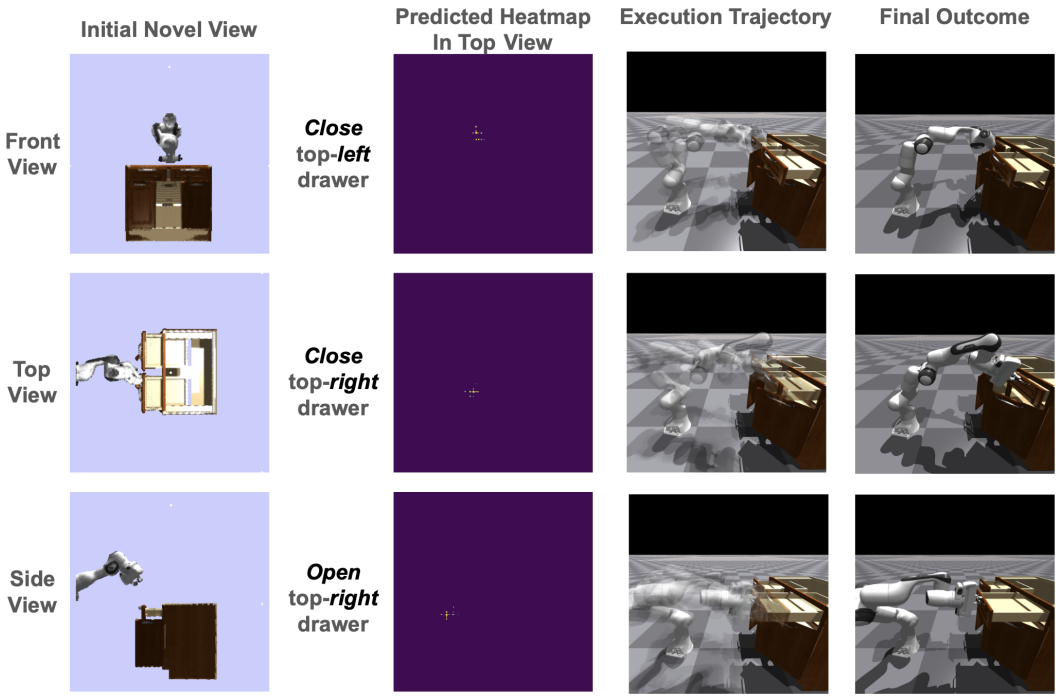
(a) Door, Object ID: **8961**



(b) Faucet, Object ID: **154**



(a) Table, Object ID: **19898**



(b) Storage, Object ID: **41083**

Table 3: Object Mode Sampling Evaluation with More Interaction Modes: We selected objects with more than two interaction modes and evaluated the mode sample success rates using ActAIM2. In practice, ActAIM2 executes actions sampled uniformly across 8 clusters and then reports the corresponding interaction mode success rates. The reported interaction modes are illustrated in Figure 19.

Object Category	Object ID	1st Mode% ↑	2nd Mode% ↑	3rd Mode% ↑	4th Mode% ↑
Faucet	154	65.4	16.3	6.2	0
Table	19898	40.1	16.2	13.2	6.3
Door	8961	79.3	43.2	14.2	10.2
Storage	41083	55.4	26.1	17.3	0

7.8 More Experimental Results

In this section, we have added two additional experimental results: object-specific mode sampling evaluation over ActAIM2, where the objects exhibit more than two interaction modes, and average sample success rate evaluation over ActAIM2 with limited perception information. These results aim to demonstrate the stability and robustness of ActAIM2 when predicting outcomes for more complex objects. Furthermore, we demonstrate that ActAIM2 can still accurately predict general outcomes with minimal visual information. This suggests the potential for simplifying the scene setup to adapt ActAIM2 for real-world experiments.

7.8.1 Mode Sampling Evaluation on objects with more Interaction Modes

In this section, we assess the mode sampling effectiveness on an object-wise basis as detailed in the experiment section of our paper (see Table 2). We focus on the first and second mode clusters identified by ActAIM2, presenting only two modes because we are examining the average mode sample success rate across categories, typically involving around eight different instances. For simpler articulated objects like a table with a single drawer or a faucet with a single switch, there are typically only two valid interaction modes. This is because these objects are initialized at the mean joint value $j_{\text{init}} = (j_{\text{max}} + j_{\text{min}})/2$. However, most articulated objects feature more than two joints, leading to multiple interaction modes.

To illustrate this, we select more complex objects from various categories, including faucets, tables, doors, and storage units. We first showcase plots depicting the interaction modes for each object from Figure 19. Following this, we present how ActAIM2 predicts the various interaction modes. These objects and their interaction demonstrations are also featured on our website through illustrative videos.

From Table 3, we find that ActAIM2 is capable of predicting multiple interaction modes for more complex objects and achieves a reasonable success rate among the demonstrated interaction modes.

7.8.2 ActAIM2 evaluation with less visual information

In the main paper, we utilize five cameras positioned around the articulated object to gather visual information. To assess the model's applicability in real-world scenarios, we evaluate its performance when trained with reduced visual information. Specifically, we conduct two additional experiments: one training with only depth information and another using a single view RGB-D camera. We employ the same dataset as in the experimental section but mask out extra information. For qualitative results, please refer to. Our results demonstrate that ActAIM2 can effectively train with less visual data, with only an acceptable level of performance decrease.

From Table 4, we find that training with only depth information does not significantly affect performance, with less than a 5% decrease in success rate when color is omitted. However, using only a single camera view impacts the success rate more substantially, particularly when the model is applied to larger objects. Scalability also significantly decreases with a single view camera. Nonetheless, performance on previously seen objects remains at a similar level.

Table 4: Robotic Interaction Mode Discovery with Less Vision We evaluate ActAIM2 under conditions of reduced visual information to demonstrate its stability and robustness. Specifically, we list experiments where ActAIM2 is trained using only depth data and single-view camera observations. Despite these constraints, it maintains a success rate drop within an acceptable range.

Test Set	Seen Objects							Unseen instances							Unseen Cats			
SSR % ↑							AVG							AVG				AVG
ActAIM2	65.3	43.2	52.1	69.2	25.3	36.2	48.6	44.9	41.2	41.5	60.2	20.1	24.4	38.7	34.3	28.9	34.1	32.4
Depth Only ActAIM2	61.5	41.9	49.3	61.5	26.3	35.1	45.9	44.3	42.6	34.2	43.7	21.4	20.4	34.4	30.2	21.2	29.5	27.0
Single View ActAIM2	55.3	32.1	35.2	45.3	21.4	27.9	36.2	32.5	26.1	29.7	32.1	15.3	13.2	24.8	15.3	13.2	12.3	13.6

7.9 Comparison between Prior Works

7.9.1 Comparison between ActAIM [12]

We state that the main differences between our work and [12] lie in 2 major aspects which are data collection (environmental setup) and method (discrete representation).

Data Collection For ActAIM, we utilized a floating parallel-jaw gripper for data collection, eliminating concerns related to invalid collisions or reachability issues. The action primitives learned by ActAIM include identifying the point to graspor interact and determining the direction to move post-grasp, akin to the approach used in Where2Act [43].

For ActAIM2, we employed a full Franka Emika robot, necessitating consideration of workspace constraints and potential collisions with the robot body. Directly applying the GMM adaptive data collection method from ActAIM proved challenging; hence, we incorporated a heuristic grasping strategy to aid in identifying graspable parts of the object. We developed an analytical technique to determine which segments of the point cloud are likely to correspond to handles on articulated objects. This was followed by using DBSCAN to identify and filter potential heuristic handles. Subsequently, we employed ContactGraspNet [55] to predict possible grasping points, as illustrated in Figure 5 of the Appendix. We combined these heuristic grasping actions with randomly sampled data to execute GMM adaptive sampling, similar to ActAIM, ultimately integrating these components into our final dataset.

Method Both ActAIM and ActAIM2 share the approach of decomposing the policy into a mode selector and an action predictor. The key difference lies in how they represent the interaction mode. In ActAIM, a simple Conditional Variational Autoencoder (CVAE) is used as the mode selector to represent the interaction mode for manipulating articulated objects. However, the latent space in this approach forms a Gaussian distribution, which cannot be easily sampled or clustered.

In Section 5.3, we experimented on interaction mode grounding, demonstrating that ActAIM struggles to associate task embedding with a specific interaction mode. The purpose of this experiment was to show that if the structure of the latent space is well-organized (meaning that all interaction modes are distinctly learned), the grounding should converge after a few interactions. The results revealed that ActAIM fails to provide a clear and discrete representation of interaction modes.

Furthermore, we visualized the interaction mode representation from ActAIM, as shown in Figure 14 of the Appendix. The visualization indicates that the interaction modes learned by ActAIM are noisy and disorganized, with different modes scattered throughout the space. This explains why the experiment in Section 5.3 did not successfully ground specific interaction modes.

7.9.2 Comparison between works with affordance prior

Interaction modes are characterized by significant visual changes in the observation space. In contrast to methods such as SayCan [7], VADER [56], or RLAfford [57], our approach focuses more on visual affordance without incorporating action data. The interaction mode seeks to predict and categorize the limited future states of articulated objects, such as opening or closing a partially open drawer. Ideally, the mode selector $p(\epsilon|o)$ could function as the encoder in a video generative model. In the context of articulated object manipulation, where visual information is readily available,

our mode selector model is trained using a pre-trained visual encoder (VGG-19 in our case). This mode selector could be substituted with any world model or video prediction model that forecasts possible future states. Ultimately, the interaction mode infers a distribution over plausible future states achievable through interaction, based on an input image of a scene. For articulated objects, any visual change in the object, especially if it occludes the robot, is significant. We aim to extract and cluster such information to facilitate learning in the action decoder.

7.10 Limitations

In this section, we discuss 3 major limitations of the current ActAIM2 which are imbalanced success rate in different interaction modes, lack of general scalability, and limited to articulated object manipulation. Moreover, we provide potential solution to all these limitations.

7.10.1 Imbalanced Success Rate

The experimental results demonstrate that while ActAIM2 can represent different interaction modes using discrete clusters, the success rates among these modes are imbalanced. The Mode Sampling Evaluation table shows that the average success rate in the second mode cluster is much lower than in the first mode cluster for ActAIM2. Additionally, Figure 6b illustrates this imbalance using a half-open drawer demonstration: pushing the drawer in has an over 80% success rate, whereas opening the drawer only achieves about a 20% success rate.

This imbalance in success rates could impact the application of ActAIM2, particularly in interaction mode grounding and long-term planning. As shown in section 5.3, the interaction mode grounding experiment using RL required nearly 50 iterations to distinguish between closing and opening modes, with the average success rate for opening still not being high enough.

In more complex scenarios with multiple articulated objects, this imbalance might affect the planning process if ActAIM2 is used as a prior. It can be difficult to determine whether the sampled interaction mode will lead to a "nothing happens" outcome or a rare interaction mode. Therefore, multiple sampling attempts are needed, reducing the efficiency of data use.

Based on our experience with model training, we believe that an imbalanced dataset is the cause of this issue. Although we use the GMM adaptive data collection method to increase interaction mode diversity, the challenge remains that modes requiring precise grasping need more accurate action labels. To address the problem of imbalanced success rates, we propose collecting more stable data. Specifically, we suggest incorporating more expert tele-operation control data into the dataset to achieve more reliable grasping, rather than relying solely on completely self-generated data.

7.10.2 Lack of General Scalability

In the experimental table, we observed a significant decrease in success rates from seen objects to unseen objects, and further to unseen categories. The success rate for generalizing across different categories dropped by nearly 50%. While ActAIM2 shows some level of scalability, it performs poorly when scaling across different categories.

To address this scalability issue, we suggest training our model on more data. Currently, ActAIM2 is trained on only 6 categories with 8 unique instances, and we have collected 150 trajectories for each articulated object. We believe that incorporating a greater variety of object categories is necessary to better demonstrate scalability across different types of articulated objects.

7.10.3 Limited to Articulated Object Manipulation

ActAIM2 is designed by decomposing the policy into a mode selector and an action predictor, where the mode selector defines distinct interaction modes. In this work, we define an interaction mode as a significant visual change in the articulated object, assuming that any visual change is meaningful if it pertains to the object. However, this definition doesn't hold true for tool manipulation tasks. For

example, in a hammering task, simply moving the hammer around on the table without grasping it is meaningless, yet it would still qualify as an interaction mode under the current definition.

This misalignment leads to inefficiencies during training when applying ActAIM2 to tasks like hammering. Additionally, data collection becomes a significant challenge due to the longer task duration and the need for more precise actions.

To address these issues, we suggest first improving the data collection process and redefining the concept of interaction modes for tool manipulation tasks. Specifically, we should introduce stable grasping as an additional interaction mode and utilize large video-language models to assess whether the resulting visual changes are meaningful.

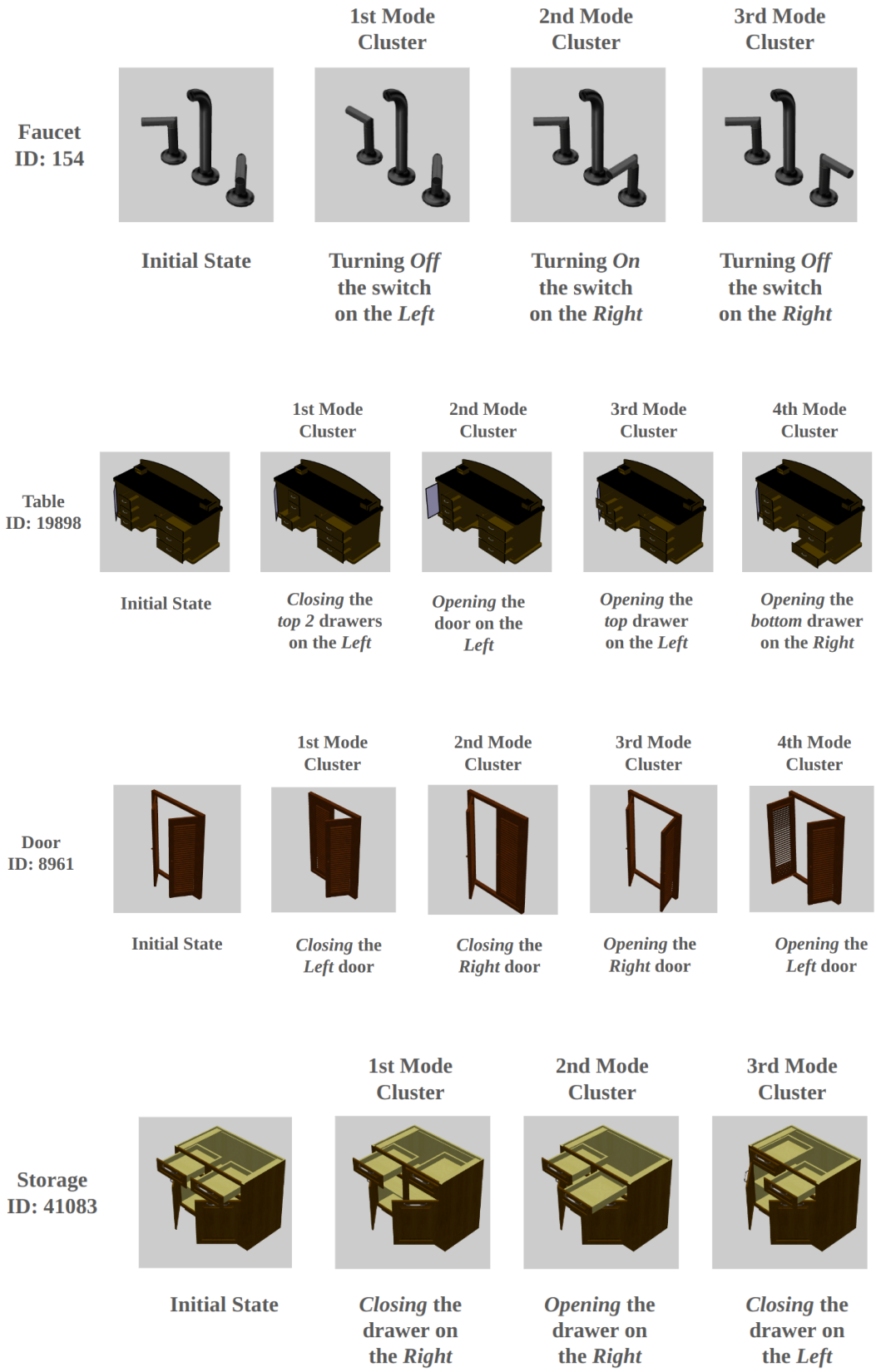


Figure 19: We illustrate objects from various categories that exhibit more complex interaction modes. These objects are qualitatively evaluated based on their sample success rates for each presented interaction mode. Video demonstrations of manipulating these objects are also available on our website.