7 Supplementary

7.1 Assumptions and Future Research

7.1.1 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.

7.1.2 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.

7.2 Dataset Generation

7.2.1 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:

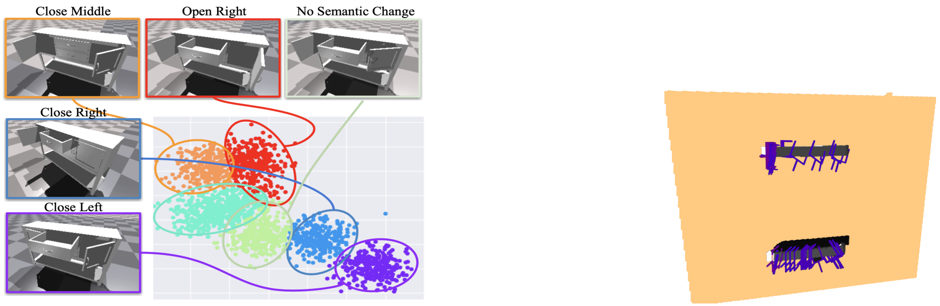
$$z_j = v_j^{init} - v_j^{final} = \mathcal{E}_O(O_j^{init}) - \mathcal{E}_O(O_j^{final}) \tag{9}$$

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 5: (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 hyperparameter 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:

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

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 $(O, a, \epsilon)_j$.

7.2.2 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.

7.3 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 $\epsilon > \bar{\epsilon}$
  $D \leftarrow \emptyset$                                    ▷ Set the initial dataset to be empty
  while $D$ not have enough data do
      $D_r = \{(a, o)_i\} \sim$ RandomSampling                 ▷ Random Sampling
      $G = \{g_i\} \sim$ GraspNet($O^i$)                         ▷ Sample Grasp using GraspNet
      $D_g = \{(a, o)_i\} \sim$ GenerateTraj($G$)               ▷ Gnerate trajectory based on grasp
      $D \leftarrow D \cup D_r \cup D_g$
      $\epsilon_i \sim D$                                      ▷ Compute task embedding in current $D$
      Cluster $\epsilon_i$ with GMM, assign cluster label on each trajectory
      $\{D_j | j = 1, ..., K\} \leftarrow D$
      $D_{GMM} \leftarrow \emptyset$
      for $j$ in range $K$ do
          Extract $D_j$ in $D$ based on cluster label
          $p(D_j | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left( \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \right)$    ▷ fit GMM
          $\hat{D}_j \leftarrow \{(a, o)_i\} \sim p(D_j | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$    ▷ Sample action from GMM
          $D_{GMM} \leftarrow D_{GMM} \cup \hat{D}_j$
      $D \leftarrow D \cup D_{GMM}$
```

7.3.1 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

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

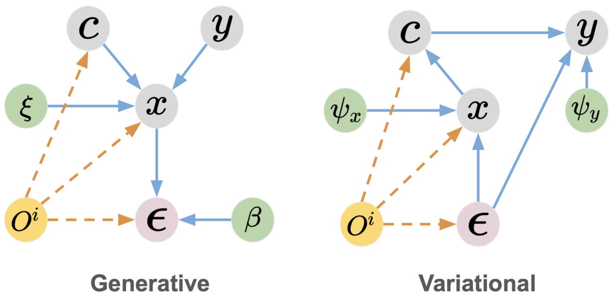


Figure 6: The graphical representations elucidate the Conditional Gaussian Mixture Variational Autoencoder (CGMVAE) framework, showcasing two distinct models: the generative model on the left and the variational family on the right. These graphical models serve to visually communicate the structural 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 6.

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 7.

7.3.2 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 9a 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 9b 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.

7.3.3 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 8.

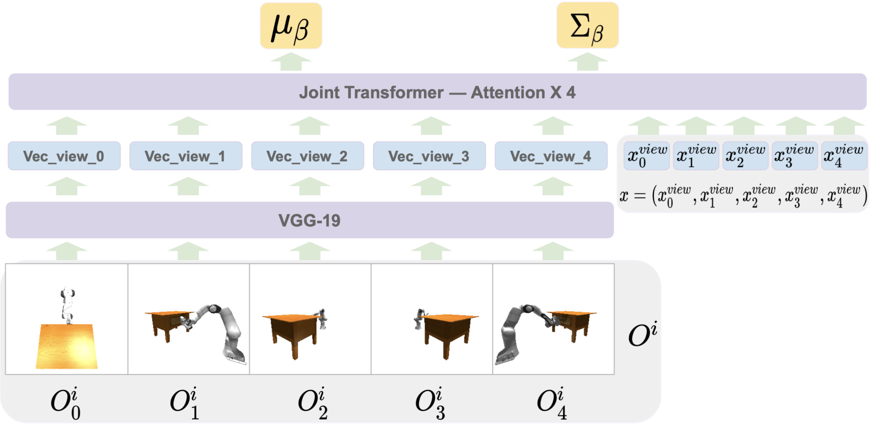

μ_β
Σ_β
Joint Transformer — Attention X 4
Vec_view_0
Vec_view_1
Vec_view_2
Vec_view_3
Vec_view_4
x_0^{view} x_1^{view} x_2^{view} x_3^{view} x_4^{view}
$x = (x_0^{view}, x_1^{view}, x_2^{view}, x_3^{view}, x_4^{view})$
VGG-19
O^i
O_0^i O_1^i O_2^i O_3^i O_4^i

Figure 7: 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.

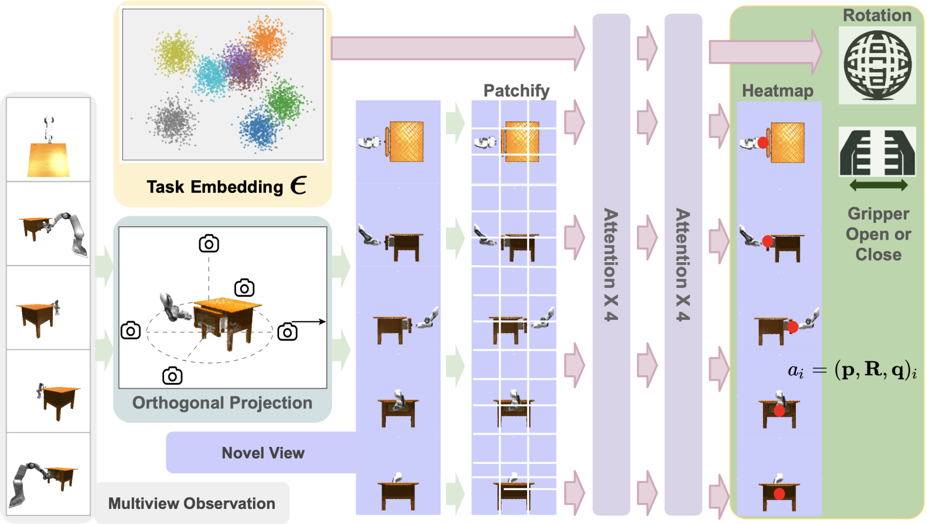
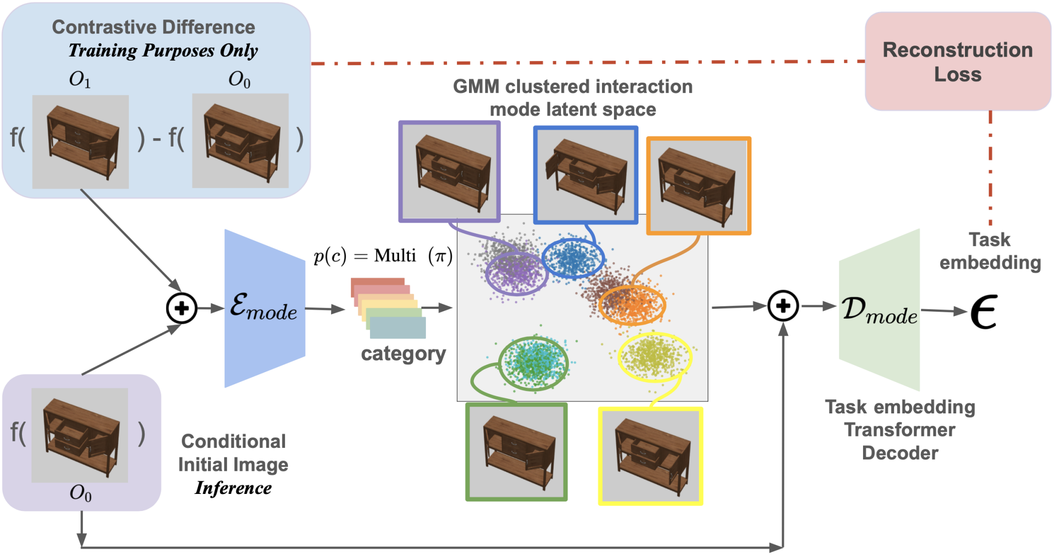


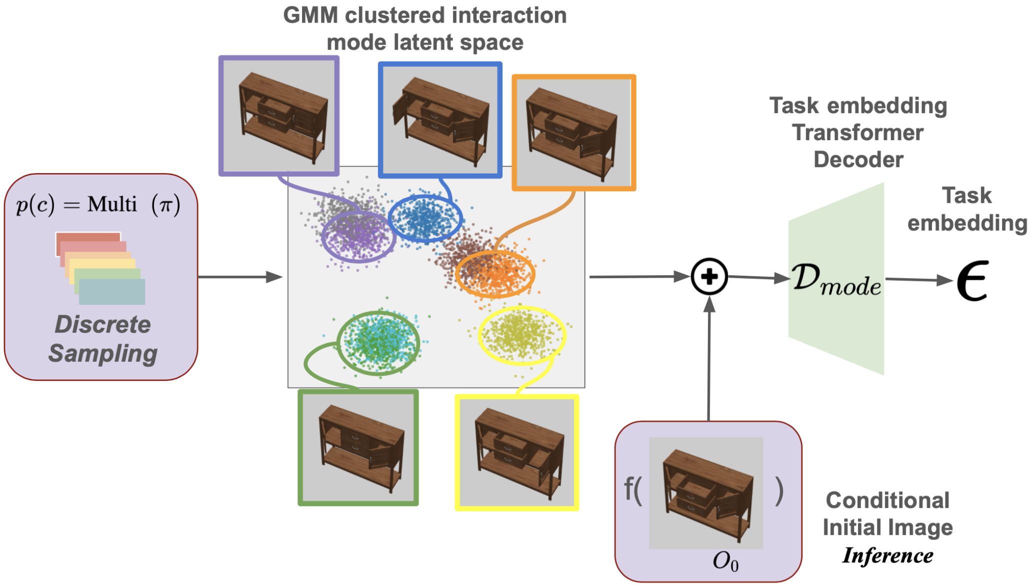
Figure 8: Action Predictor Architecture: This model integrates multi-view observations directly as input, sourced from predefined cameras within the scene. The process begins with the extraction of five RGBD images, which are subsequently transformed into RGB point clouds. These are then subject to orthogonal 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.

7.4 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.


GMM clustered interaction mode latent space
$p(c) = \text{Multi}\ (\pi)$
Discrete Sampling
Task embedding Transformer Decoder
Task embedding
$\mathcal{D}_{mode}$
ϵ
f( )
O_0
Conditional Initial Image
Inference

(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.

7.4.1 Mode selector TSNE plot Figure 14

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.

7.4.2 Action Predictor Qualitative Results

We present extensive qualitative results in Figure 15a, Figure 15b, Figure 16a, and Figure 16b, 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/.

7.4.3 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 10, 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.

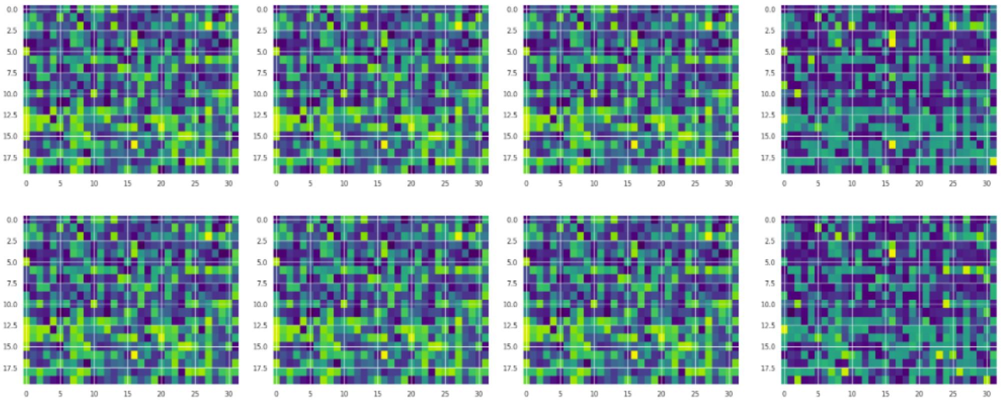
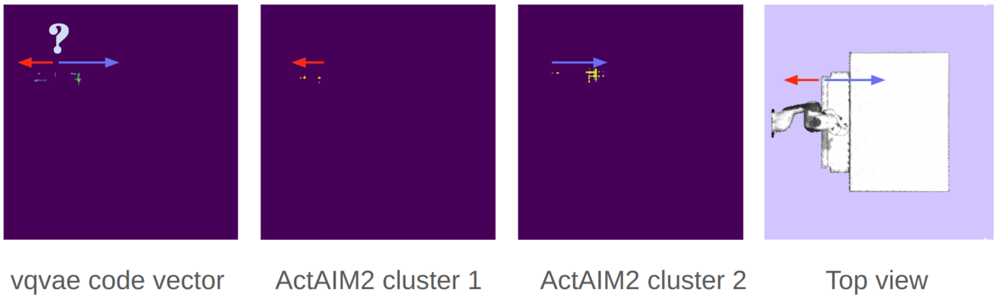


Figure 10: Visualization of Latent Space Clustering in VQVAE-RVT: This figure illustrates the distribution of eight code vectors within the latent space, categorized into two distinct clusters. These clusters are intended to represent the discrete interaction modes of opening and closing a drawer. The 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 11) reveal a lack of diversity in the action predictions, undermining the practical utility of this model configuration.



vqvae code vector

ActAIM2 cluster 1

ActAIM2 cluster 2

Top view

Figure 11: 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 \algoName when sampling from one cluster, demonstrating a specific interaction mode. (3) Action heatmap from \algoName 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 \algoName's capability to discern and represent distinct action strategies through targeted cluster sampling.

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.

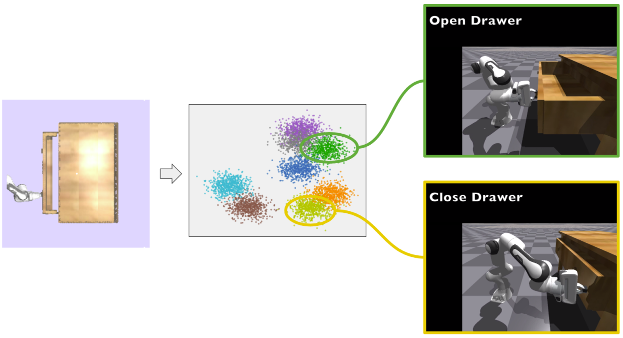


Figure 12: Opening and Closing a Drawer: This figure demonstrates the effective action sequence generated by ActAIM2 for a drawer. The left part of the image shows the drawer being opened, showcasing the robot's approach and grip adjustment. The right part of the image captures the drawer in a fully closed position, illustrating the final state after the action sequence execution.

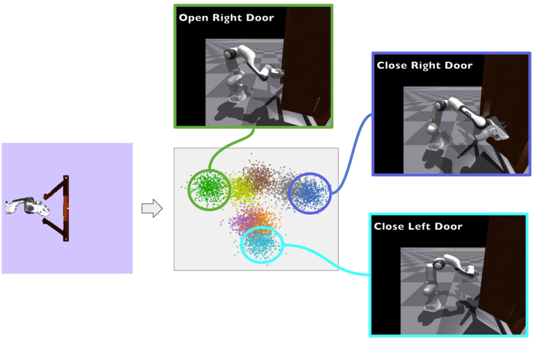


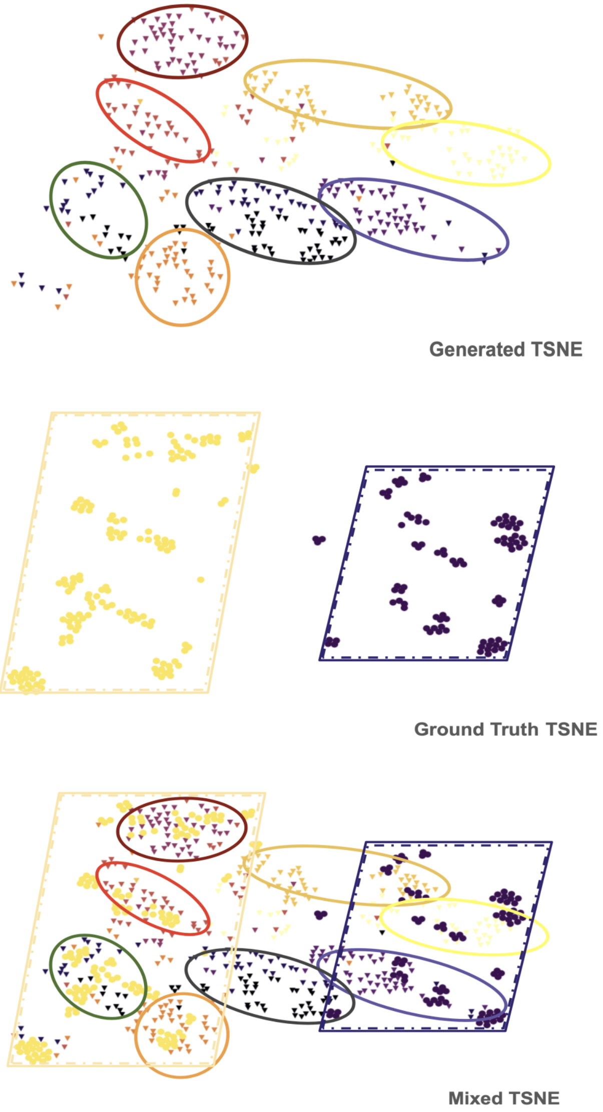
Figure 13: Opening and Closing a Door: This figure illustrates the ActAIM2's manipulation capability with a door. The left image displays the door being opened, highlighting the robot's positioning and the initial interaction phase. The right image shows the door completely closed, detailing the end of the manipulation sequence and the effectiveness of the action predictor.

8 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.
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 12 and Figure 13, 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.

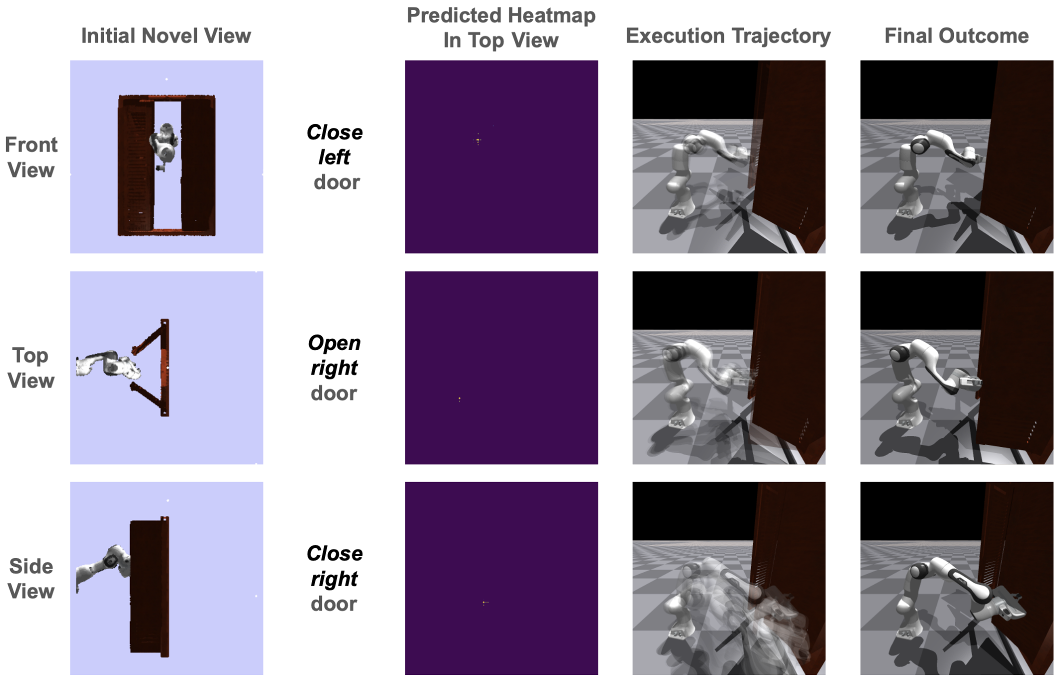


Generated TSNE

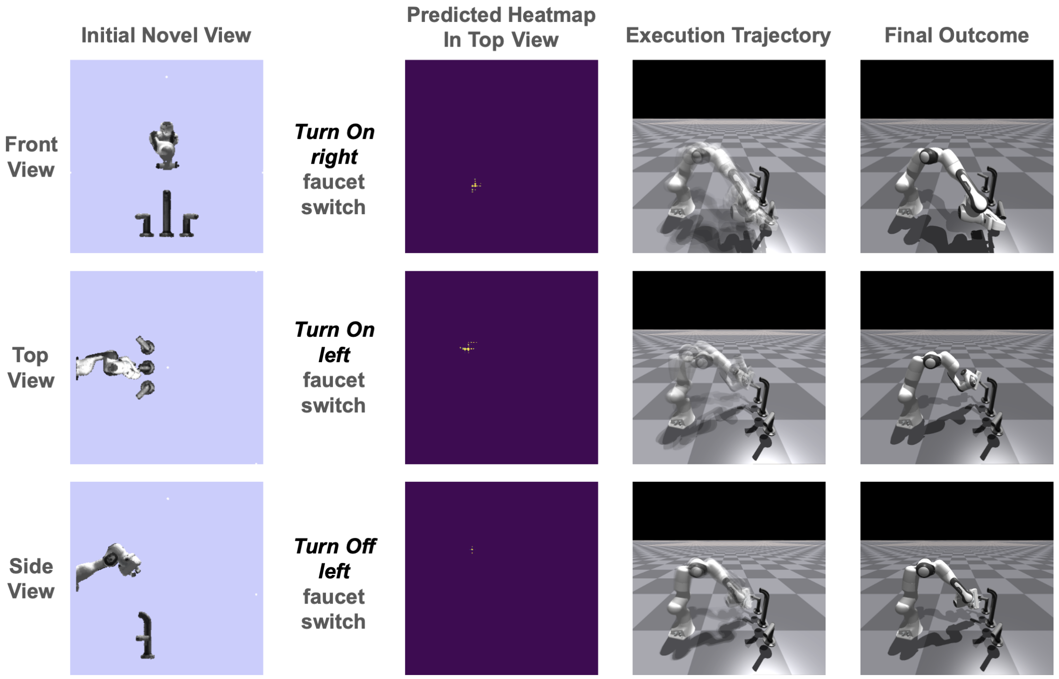
Ground Truth TSNE

Mixed TSNE

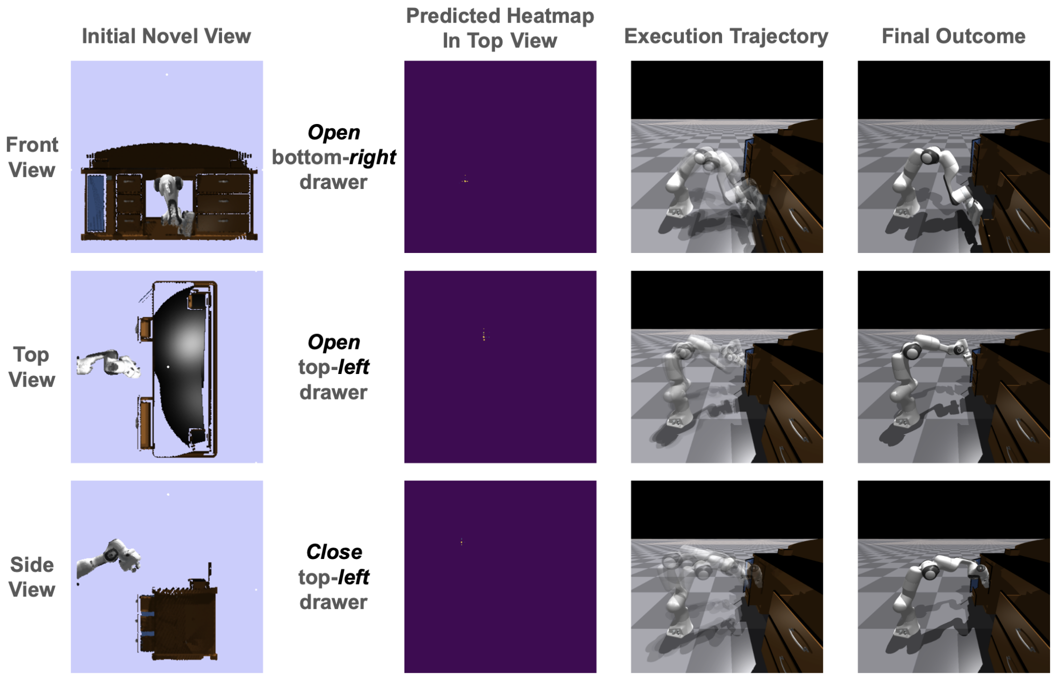
Figure 14: Disentanglement Visualization with CGMVAE: This figure illustrates the efficacy of the Conditional Gaussian Mixture Variational Autoencoder (CGMVAE) in disentangling interaction modes for the "single drawer" object (ID: 20411), using a t-SNE plot for visualization. Task embeddings ϵ_j, defined 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.



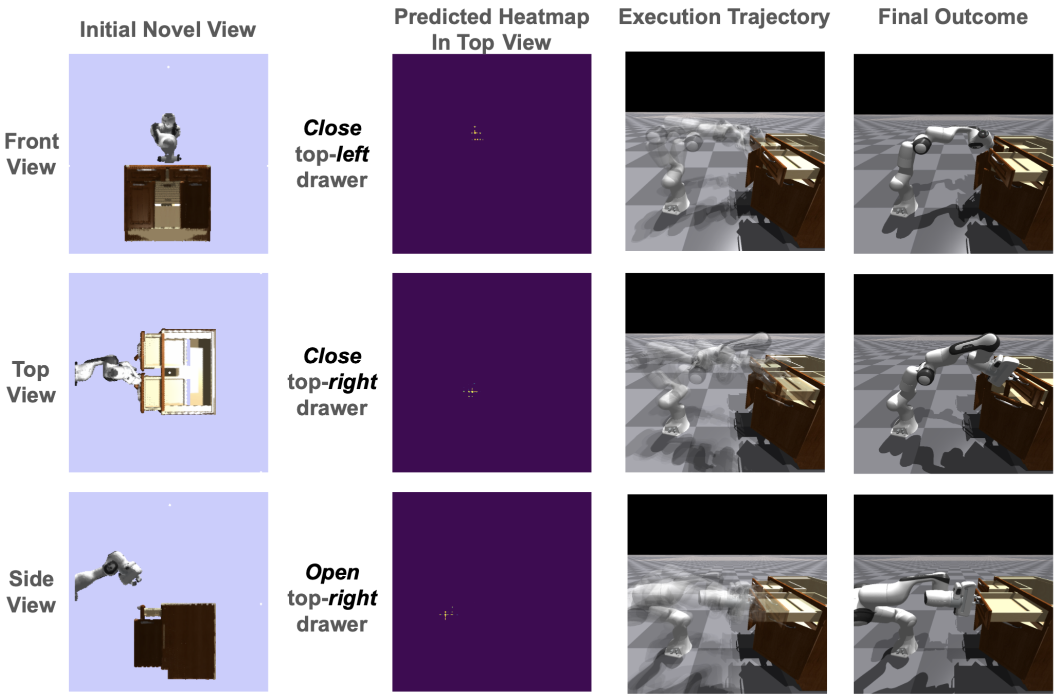
(a) **Door**, Object ID: **8961**


Initial Novel View
Predicted Heatmap In Top View
Execution Trajectory
Final Outcome
Front View
Turn On right faucet switch
Top View
Turn On left faucet switch
Side View
Turn Off left faucet switch

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



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


Initial Novel View
Predicted Heatmap In Top View
Execution Trajectory
Final Outcome
Front View
Close top-*left* drawer
Top View
Close top-*right* drawer
Side View
Open top-*right* drawer

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