Learning Multimodal Probabilistic Models of Manipulation Skill Effects

Siyeon Kim
University of Utah

Mohanraj D. Shanthi*
University of Utah

Yixuan Huang*
Princeton University

Tucker Hermans
University of Utah; NVIDIA

(*) indicates dual second author

1. Introduction

Although robotic manipulation has made significant progress in simulation and structured settings, it often struggles in real-world scenarios where uncertainty is inevitable. This challenge is particularly non-negligible for sim-to-real learning because a reality gap naturally emerges from modeling approximations and parameter mismatches between simulation and the real world [1, 10, 12]. The uncertainty arises not only from perception errors, including sensor noise, incorrect segmentation, and incorrect object recognition, but also from motion execution errors resulting from imperfect control [6] and calibration to unmodeled effects due to approximations in the dynamics [1]. Such uncertainty due to these unmodeled effects is often overlooked. For example, consider a robot pushing an object, which involves non-linear and non-smooth contact dynamics that are difficult to model exactly. Depending on the direction of the push, the point of contact, the object's geometry, and its inertial properties, the object can be slid, toppled, or rotated. Even a minor perturbation in any of the above parameters can result in significantly different and unintended effects on the object's poses. Therefore, it is necessary to learn a model to capture this multimodality of effects of skill to enable robust planning under uncertainty.

Prior work on learning skill effect models for robot manipulation [7, 11, 13] leverages deterministic models to predict the 3D position changes for multi-object interactions. Huang et al. [7] utilize a transformer network-based dynamics model and a two-layer MLP-based decoder to estimate the relative position change of objects at each time step. However, these deterministic models are unable to capture distinct multi-modal effects when uncertainty is high. Learning stochastic dynamics models has a rich history in robot manipulation, with PILCO defining a common approach using Gaussian process regression [5]. However, because of the Gaussian process assumption coupled with moment matching for efficient uncertainty propagation, common models cannot capture multi-modal outcomes.

Conkey and Hermans [4] advocate for leveraging probability distributions as a goal representation for planning under uncertainty. This work demonstrates its ability to capture state uncertainty by learning a mixture density network (MDNs) [2] of object dynamics, which can be used in planning to generate robust plans. However, the learned dynamics model assumes it's given a known object a priori.

We thus extend the MDN dynamics learning approach to learn uncertainty in a skill effect model using an input partial point cloud, rather than a known object. Additionally, our skill effect model adopts a latent dynamics model for robust prediction. We show that our model successfully deals with predicting skill effects under uncertainty with diverse object models from [3] for a push skill. Figure 1 illustrates the effect of learning a multimodal dynamics model over a deterministic and an unimodal model. Moreover, in Section 4, we showcase that our multimodal model outperforms deterministic and unimodal baselines on object pose prediction and log-likelihood in Table 1. It also maintains coherent predictions from low to high pushes as shown in Figures 1, 2, across YCB objects with different geometries.

We present the following contributions.

- **Modeling skill-effect uncertainty** - We introduce a mixture density network model to capture multimodal, discontinuous state uncertainty, and demonstrate that the model outperforms deterministic models in planning under uncertainty.
- **Latent dynamics for robust prediction** - Our model leverages the latent dynamics model for robust single-step prediction for a push skill
- **Object-agnostic perception** - Our model utilizes segmented object point cloud data without using any object priors.

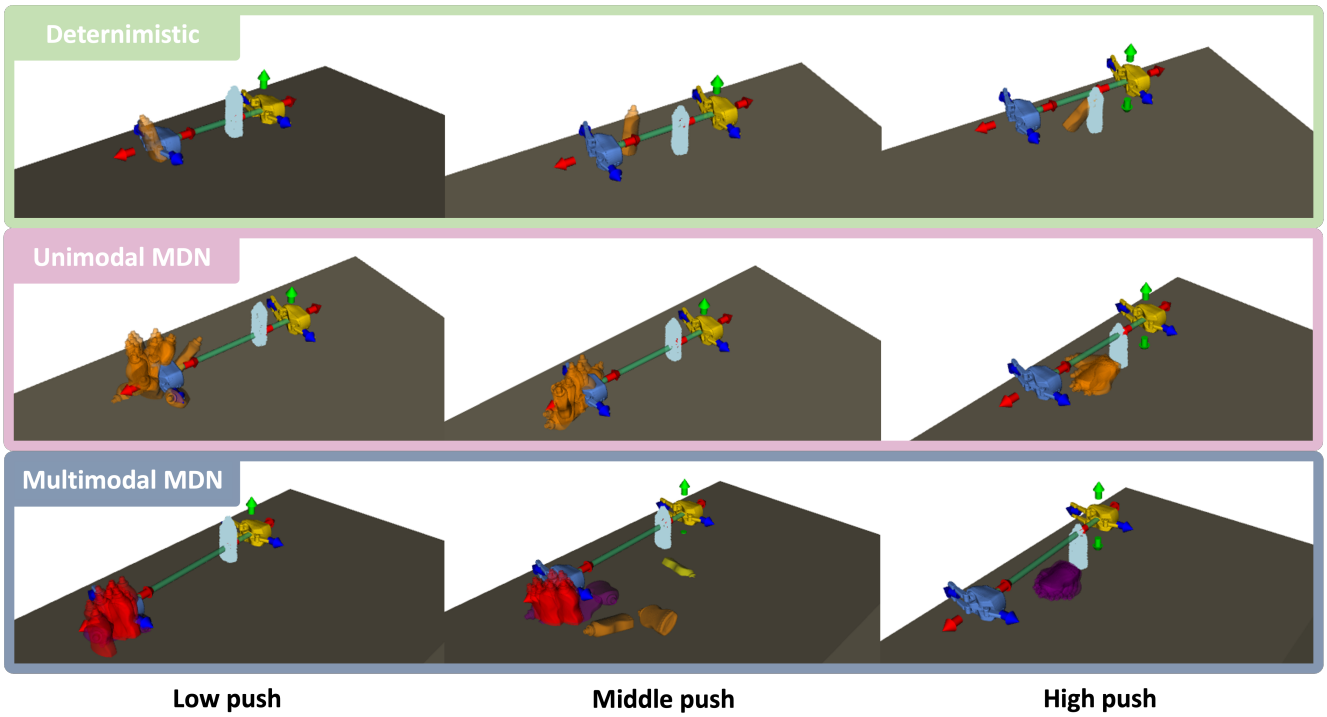

Figure 1. Qualitative comparison of the push skill at low, medium, and high heights comparing a deterministic model **(Top row)**, unimodal (1-component MDN) **(Middle row)**, and multimodal (5-component MDN) **(Bottom row)**. At test time, we predict the object's pose change from the input partial-view point cloud and apply it to transform the input cloud. Any full point cloud and object model shown here are for visualization only.

2. Problem Definition

We seek to learn a predictive skill-effect model that takes a partial-view point cloud S, along with an action A as input and outputs the change in pose for all objects in the scene resulting from the action. We assume the point cloud consists of N object and environment segments $O^i \subset O$, where $i = 0, \ldots, N$. Our skill effect model outputs corresponding $SE(3)$ object poses. We further assert that the model can be decomposed into an encoder Enc, a latent dynamics model T, and a decoder Dec.

We define our robotic planning task as finding a sequence of skills a_t and their parameters θ_t, $\tau = (A_0, \ldots, A_{T-1})$, where $A_t = (a_t, \theta_t)$. Then, we can formulate this planning task as an optimization problem for the skill sequence that has the minimum expected trajectory cost $J(\tau) = \mathbb{E}_{\mathbf{p}_{0:H-1} \sim p(\tau)}[c(\tau)]$ as follows:

$$\tau^* = \underset{\tau=(A_0,\ldots,A_{T-1})}{\operatorname{argmin}} \quad J(\tau)$$

$$= \underset{\tau}{\operatorname{argmin}} \int \prod_{k=0}^{H-1} \mathcal{P}(\mathbf{p}_{k+1} | \mathbf{z}_k, A_k) \sum_{j=1}^{H} c(\mathbf{p}_j, \mathbf{p}_g) d\mathbf{p} \quad (1)$$

$$\text{subject to } \delta \mathbf{z}_{t+1} = T(\mathbf{z}_t, A_t) \quad \forall t = 0, \ldots, H-1 \quad (2)$$

$$\mathbf{z}_0 = Enc(S_0) \quad (3)$$

$$\delta \mathbf{p}_t = Dec_p(\delta \mathbf{z}_t) \quad \forall t = 0, \ldots, H-1 \quad (4)$$

$$A_t \subset \mathcal{A} \quad \forall t = 0, \ldots, H-1 \quad (5)$$

$$\theta_{min} \leq \theta_t \leq \theta_{max} \quad \forall t = 0, \ldots, H-1 \quad (6)$$

3. Methods

We now provide details of our model architecture, the training procedure, and how we perform planning to solve the problem as stated in Section 2.

3.1. Model architecture

Encoders: Given input point cloud S, we first extract segment-specific feature vectors from each point cloud segment, $\mathbf{x}^i = Enc_{pc}(O^i)$, using a point cloud encoder Enc_{pc} from [14]. We use a learned positional embedding, $pos_i = \mathsf{Emb}_p(i)$, to identify each segment and concatenate it with segment feature vectors to create a latent state for each segment, $\mathbf{z}^i = pos_i \oplus \mathbf{x}^i$.

We encode discrete skill parameters, i.e., object identities

of interest, to which the action will be applied, into a one-hot embedding and concatenate them with embeddings of continuous skill parameters θ_t. For both push skills, we provide end effector poses when the action starts and ends; especially, we translate end-effector start pose with respect to target object pose at initial step and its end pose with respect to start pose: $\theta_t = ({}^{o_t}T_{ee_{start}}, {}^{ee_{start}}T_{ee_{end}})$. Furthermore, for continuous skill parameters, we parameterize 6-DoF relative skill poses $\delta\mathbf{p}$ as $(t, R) \in SE(3)$, where $t \in \mathbb{R}^3$ and $R \in SO(3)$. Like [9], we parametrize skill parameters and object poses as 9-DoF pose vectors: a translation $t \in \mathbb{R}$ and a continuous 6D rotation given by two unconstrained vectors $c_1, c_2 \in \mathbb{R}^3$, which are orthonormalized to yield $R \in SO(3)$. Unlike Euler angle and quaternion, this continuous rotational representation guarantees the continuity of orientation for our learning model, and the two vectors c_1, c_2 can be easily retrieved into the rotation matrix R by a Gram-Schmidt-like process [15].

Latent dynamics model: Like [7], given latent state and skill embeddings, our latent dynamics model T predicts a relative pose change between the current and following states of an object in latent space, $\delta\mathbf{z}_t = T(\mathbf{z}_t, A_t)$.

Decoder: In order to predict relative object pose from latent state and action embeddings, we employ the pose decoder Dec_p: $\delta\mathbf{p}_t = Dec_p(\delta\mathbf{z}_t)$. We then compute its corresponding transformation matrix $\omega(\delta\mathbf{p}_t)$, which transforms the current point cloud to the next point cloud, $O_{t+1} = \omega(\delta\mathbf{p}_t)O_t$. We define the centroid of the point cloud as the origin of itself and align the initial object frame's directions with the world frame, then the pose vector of point cloud can be also viewed as $\mathbf{p}_{t+1} = w(\delta\mathbf{p}_t)\mathbf{p}_t$. We evaluate that our model works well with an MDN-based decoder model in the following Section 3.2.

3.2. MDN Probabilistic Skill Effect Model

Like [4], we leverage an MDN [2] as a pose decoder to predict the distribution of $SE(3)$ object poses. Unlike the deterministic model-based decoder, this stochastic model is able to capture multi-modal skill effects, such as objects that stand up and topple down. The MDN defines the parameters of a Gaussian mixture model, $\mathcal{P}(\mathbf{p}_{k+1}|\mathbf{z}_k, \mathcal{A}_k) = \sum_{i=1}^{N} \alpha_i(\delta\mathbf{z}_t)\mathcal{N}(\mu_i(\delta\mathbf{z}_t), \Sigma_i(\delta\mathbf{z}_t))$ and α_i, μ_i, and Σ_i are mixing coefficients, mean, and covariance for each component, respectively, and are output from the learned MDN decoder. We make the dependence of these predictions on the predicted latent delta action $\delta\mathbf{z}_t$ explicit, to denote their structure as output of the neural network. We can ensure the mixing parameters sum to 1 by having them be output from a softmax. The mean parameters are standard linear output models, while we follow common practice to ensure the parameters of the covariance matrix are positive definite [2, 4].

3.3. Training

We utilize a combination of loss functions, $L_{tot} = a \cdot L_{reg} + b \cdot L_{pose}$, to train our model in an end-to-end manner. Firstly, we adopt the regularization loss from [7] to regularize latent states: $L_{reg} = \|\mathbf{z}_{t+1} - \hat{\mathbf{z}}_{t+1}\|_2^2$, where the ground truth of latent states at current and next time steps are $\mathbf{z}_t = Enc(O_t)$ and $\mathbf{z}_t = Enc(O_{t+1})$, respectively, and the predicted latent states at next step is $\hat{z}_{t+1} = z_t + \delta z_t$. Next, we define the second loss term $L_{pose} = g({}^{o_t}T_{o_{t+1}}, {}^{o_t}\hat{T}_{o_{t+1}})$ for predicting the change of object pose, ${}^{o_t}T_{o_{t+1}} = {}^{w}T_{o_t}^{-1} \cdot {}^{w}T_{o_{t+1}}$, by converting object following pose o_{t+1} frame from world w to current object pose o_t. We assume no covariance between position and orientation for simplicity. For the deterministic decoder model, we leverage a geodesic loss and, for the MDN-based decoder, compute position as a multivariate normal distribution in Euclidean space and orientation as an $SO(3)$ Riemannian manifold. Then, we sum their log-likelihood to resolve the difference in unit/scale in position and orientation.

3.4. Planning

We sample the 18-DoF action parameters using a Cross-Entropy Motion (CEM) planner [8] to solve our single-step planning problem Eq.(1). For this planning problem, we encode the latent dynamics as a constraint in Eq. (2), given a latent embedding encoded from input observation with Eq. (3). It also has a constraint that decodes the output latent embedding into an object pose change with Eq. (4), with constraints on action parameters in Eqs. (5)–(6). After solving this problem, it returns the optimal trajectory τ^* that has the lowest cost by computing the object pose, $\mathbf{p}_{t+1} = w(\delta\mathbf{p}_t)\mathbf{p}_t$.

We cannot compute the expected cost of each trajectory from Eq. (1) exactly for the case of the MDN density. Instead, we suggest a Markov Chain Monte Carlo (MCMC)-style sampler over trajectories using to approximate Eq. (1) as $\text{argmin}_\tau \frac{1}{N_\tau} \sum_{i=0}^{N_\tau} \mathcal{P}(\tau)c(\tau)$. We provide the detailed algorithm in Supplementary Material A.

4. Experimental Results

4.1. Data Collection

We generate a dataset for a push skill in the IsaacGym simulator using a 7-DOF KUKA iiwa arm. We train different skill-effect models to manipulate a bleach cleanser, a mustard bottle, and a cracker box from the YCB dataset [3]. For each object, we first collect random nominal skill execution and then create perturbed variants based on it. In addition, we add approximately 15% of negative examples. The dataset contains 6,500 to 7,300 trajectories for each object and approximately 20,000 trajectories in total.

4.2. Simulation Experiments

We compare our model with the deterministic regressor, unimodal (1-component MDN), and multimodal (5-component MDN) to assess prediction performance and uncertainty estimates. As the deterministic baseline, we adopt the latent dynamics of Points2Plans [7] to directly regress the object pose in SE(3).

Model prediction: As shown in Figure 1, the deterministic regressor reliably predicts the object's position; by contrast, it shows inaccurate prediction of orientation by averaging the orientations observed in the dataset, resulting in a half-toppled pose. The unimodal MDN produces a plausible prediction under high push, but gives dispersed samples for both low and middle push. In comparison, we observe that our multimodal MDN-based model gives coherent and reasonable predictions across all low, mid, and high pushes.

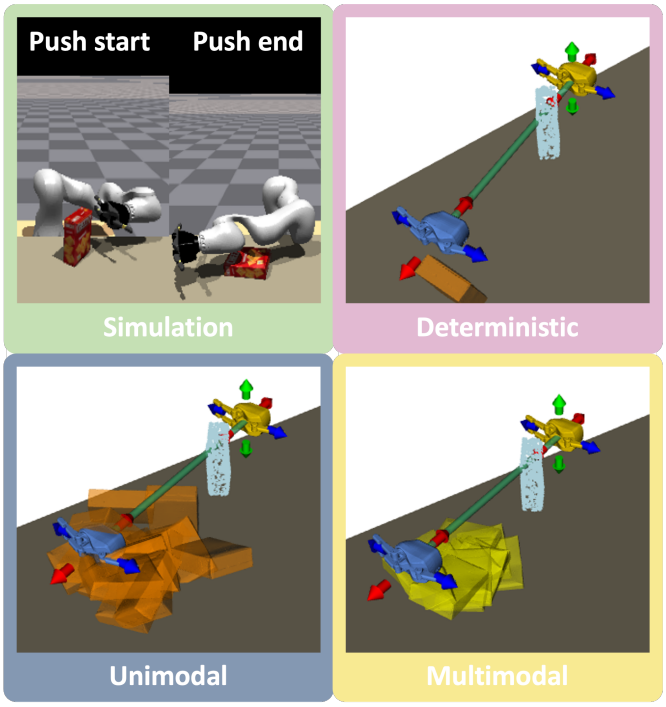

Figure 2. In order to show that our stochastic model is robust to uncertainty, we assess how close simulation rollouts and model predictions are with respect to the same actions. We show that our proposed stochastic model learn multimodal skill effects under uncertainty for both position and orientation.

Figure 3 further exhibits that our stochastic model makes consistent predictions across different object geometries. When pushing on the *wide face*, it is highly probable to predict the object toppled at all push heights. Meanwhile, when pushing on the *narrow face*, objects are more likely to be slid/rotated, remaining stable even at higher push heights. Therefore, it illustrates that our stochastic model contributes to learn the multi-modal skill effects well when there are different modes for object poses.

	Pos Err (m)↓	Rot Err (rad)↓	NLL↓
Deterministic	0.171 ± 0.048	0.733 ± 0.157	—
Unimodal	0.113 ± 0.027	0.758 ± 0.132	$(17.6 \pm 6.75) \times 10^3$
Multimodal	0.086 ± 0.018	0.463 ± 0.051	137 ± 41.7

Table 1. Mean pose error and log likelihood of object pose discrepancy between simulation results and predictions with the same actions. Position and rotation errors are represented as mean $\pm$ SE. The lower NLL values are better, and it's robust against state uncertainty.

Uncertainty quality: We assess how well the model captures uncertainty by comparing simulation rollouts and model predictions given the same actions, as shown in Figure 2. We execute ten actions for each object and evaluate them by using position error, rotation error, and negative log-likelihood (NLL). Table 1 indicates that the multimodal model achieves the lowest position/rotation errors as well as the lowest NLL for all objects. It demonstrates that multimodal MDN has both accurate predictions and a well-fitting predictive distribution. In contrast, the deterministic baseline produces only point estimates, resulting in substantially larger position and rotation errors, which means that it's unable to fully capture uncertainty. The unimodal MDN model has lower position error, but fails to model multimodal orientation outcomes, which leads to a very large NLL. Overall, our multimodal MDN model outperforms other baseline models in reflecting uncertainty in dynamics and shows more robust predictions.

5. Conclusion

We studied single-step outcome prediction for a probabilistic skill-effect model under uncertainty. Our approach learns a multimodal distribution over object pose change from partial point clouds and an action using a mixture density network (MDN) with latent dynamics. Across three YCB objects, the multimodal MDN separates different modes of object pose and outperforms deterministic and unimodal baselines in both pose accuracy and state uncertainty quality. In addition, we demonstrate that our stochastic model is capable of learning state uncertainty with different object geometries. As this paper focuses on single-step prediction, we will integrate the model into multi-step planning and validate it on a real robot. We will also extend to additional skills, such as pick-and-place, and evaluate generalization in cluttered, multi-object, and diverse environments.

References

[1] Jad Abou-Chakra, Lingfeng Sun, Krishan Rana, Brandon May, Karl Schmeckpeper, Niko Suenderhauf, Maria Vitto-

ria Minniti, and Laura Herlant. Real-is-sim: Bridging the sim-to-real gap with a dynamic digital twin, 2025. 1

[2] Christopher M Bishop. Mixture density networks. 1994. 1, 3

[3] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015. 1, 3

[4] Adam Conkey and Tucker Hermans. Planning under uncertainty to goal distributions, 2022. 1, 3

[5] Marc Deisenroth, Carl Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2012. 1

[6] Kaijen Hsiao, Matei Ciocarlie, Peter Brook, and Willow Garage. Bayesian grasp planning. In *ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, 2011. 1

[7] Yixuan Huang, Christopher Agia, Jimmy Wu, Tucker Hermans, and Jeannette Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics, 2024. 1, 3, 4

[8] Marin Kobilarov. Cross-entropy randomized motion planning. In *Robotics: Science and Systems*, pages 153–160. MIT Press, 2012. 3

[9] Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox. Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects, 2021. 3

[10] Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *CoRR*, abs/2111.00956, 2021. 1

[11] Chris Paxton, Chris Xie, Tucker Hermans, and Dieter Fox. Predicting stable configurations for semantic placement of novel objects. In *5th Annual Conference on Robot Learning*, 2021. 1

[12] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019. 1

[13] Mohit Sharma and Oliver Kroemer. Relational learning for skill preconditions. In *Conference on Robot Learning (CoRL)*, 2020. 1

[14] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019. 2

[15] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2020. 3

Learning Multimodal Probabilistic Models of Manipulation Skill Effects

Supplementary Material

A. CEM planner for MDN-based model

Algorithm 1 Monte Carlo (MC)-style CEM planner

Require: The number of action candidates N_{cand}, the number of elites N_{elite}, the number of MDN samples N_{mc}, the number of MDN components C, an initial segmented point cloud O_0 and a goal pose of a target object $\mathbf{p_G}$.

global: Enc_{pc}, Dec_p, T, ω, a uniform random sampling $\mathcal{U}$
function MDN_CEM($O_0, \mathbf{p_G}$)
  Sample initial actions: $\{\tilde{A}_t^n\}_{n=1}^{N_{cand}} \sim \mathcal{U}(-r, r)^d$ ▷ Initialize diverse action distribution
  $\mathbf{z}_0 = Enc_{pc}(O_0)$ ▷ Encode initial observation
  for $k = 1, \ldots, K$ **do**
    $\mathbf{z}_0^k = \mathbf{z}_0$
    $O_0^k = O_0$
    $\{\epsilon^n\}_{n=1}^{N_{cand}} \sim \mathcal{U}(-r, r)^d$
    $\tilde{A}_{1:H}^k \leftarrow \boldsymbol{\mu}_k + \epsilon^n \odot \boldsymbol{\Sigma}_k$ ▷ Sample N_{cand} action trajectories
    for $t = 1, \ldots, H$ **do**
      $\delta\mathbf{z}_t^k = T(\mathbf{z}_t^k, \tilde{A}_t^k)$ ▷ *Delta-dynamics* function
      $\alpha_t^k, \delta\mu_t^k, \delta\boldsymbol{\Sigma}_t^k = Dec_p(\delta\mathbf{z}_t^k)$ ▷ Obtain parameters for *delta* pose distributions
      $\{y_{(s,n)}\}_{\substack{s=1,\ldots,N_{mc} \\ n=1,\ldots,N_{cand}}} \sim \mathcal{U}(0, 1)$
      $c_{(s,n)} = \min\left\{c \in \{1, \ldots, C\} \;\middle|\; \sum_{l=1}^{c} \alpha_{t,l}^k \geq y_{(s,n)}\right\}$ ▷ MDN component samples
      $\delta\mathbf{p}_{t,(s,n)}^k \sim \mathcal{N}\left(\delta\mu_{t,c_{(s,n)}}^k, \delta\boldsymbol{\Sigma}_{t,c_{(s,n)}}^k\right)$ ▷ Sample *delta* pose
      $O_{t+1,(s,n)}^k = \omega(\delta\mathbf{p}_{t,(s,n)}^k) O_t^k$ ▷ Point clouds transformations
      $\mathbf{p}_{t+1,(s,n)}^k = \omega(\delta\mathbf{p}_{t,(s,n)}^k) \mathbf{p}_t^k$ ▷ Pose transformations
      $\mathbf{z}_{t+1}^k = Enc_{pc}(O_{t+1,(s,n)}^k)$
    end for
    for $i = 1, \ldots, N_{cand}$ **do**
      $cost_i = \frac{1}{N_{mc}} \sum_{j=1}^{N_{mc}} \sum_{t=1}^{H_k} \mathcal{L}_{cost}(\mathbf{p}_{t,(i,j)}^k, \mathbf{p_G})$
    end for
    $\mu_k, \Sigma_k \leftarrow \text{MeanStd}\left(\left\{\tilde{A}_{1:H}^k \mid i \in \text{TopK}_{\min}(cost_i),\ |TopK| = N_{\text{elite}}\right\}\right)$
    ▷ Update μ and Σ using N_{elite} lowest-cost actions
  end for
end function

B. Additional qualitative results

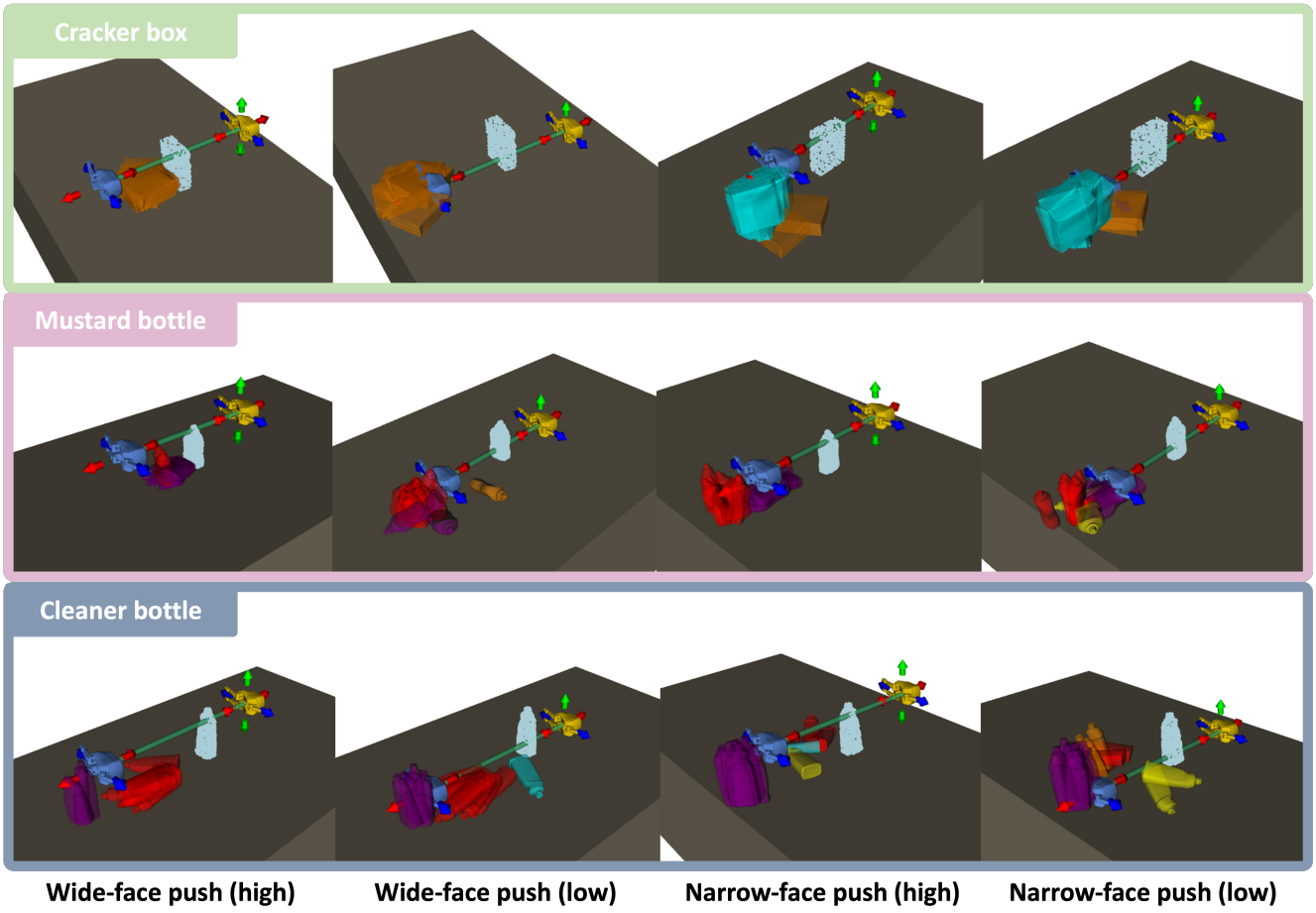

Figure 3. Our multimodal MDN (with 5 components) captures the multimodal skill effects with different object shapes. Especially here with a cracker box that has unequal side lengths, it tends to topple down easily when it pushes on the wide face (thin support along the push axis), whereas it rarely topples down when it pushes on the narrow face (wide support).