

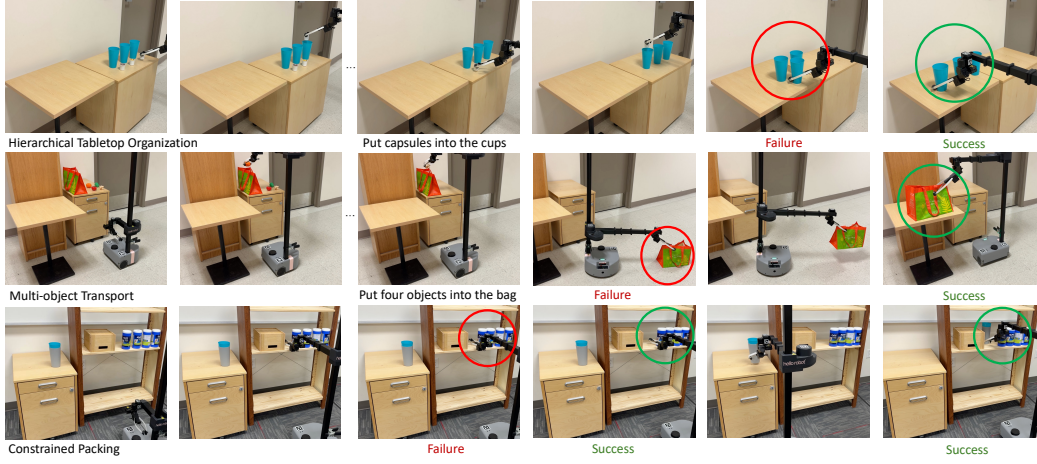
2 **A Appendix**

3 **Overview**

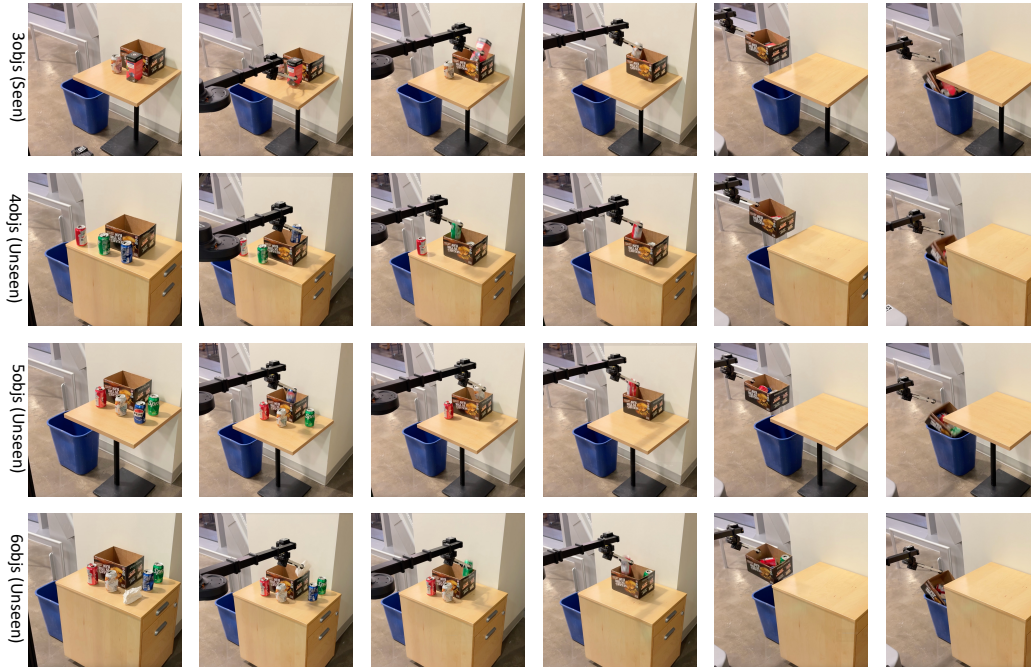
4 The appendix provides additional details, experiments, and results. Please refer to the supplemental  
5 video for real-world robot executions.

6	<a href="#">A.1 Qualitative Analysis . . . . .</a>	A2
7	<a href="#">A.2 Detailed Experimental Tasks . . . . .</a>	A3
8	<a href="#">A.3 Efficiency Experiments . . . . .</a>	A3
9	<a href="#">A.4 Key Findings . . . . .</a>	A3
10	<a href="#">A.5 Ablation Study . . . . .</a>	A4
11	<a href="#">A.6 Detailed Simulation Results . . . . .</a>	A4
12	<a href="#">A.7 Detailed Sim2Real Gap . . . . .</a>	A5
13	<a href="#">A.8 Extra Related Work . . . . .</a>	A5
14	<a href="#">A.9 Relations Definition . . . . .</a>	A5
15	<a href="#">A.10 Skills Definition . . . . .</a>	A5
16	<a href="#">A.11 Details of Skill Effect Models . . . . .</a>	A6
17	<a href="#">A.12 Real-to-sim details . . . . .</a>	A8
18	<a href="#">A.13 Stein Update Details . . . . .</a>	A8
19	<a href="#">A.14 Detailed Generalization Experiments . . . . .</a>	A9
20	<a href="#">A.15 Hardware Information . . . . .</a>	A9

## 21 A.1 Qualitative Analysis



**Figure 4:** Rollouts of real-world evaluations and corresponding failure cases. A detailed explanation of this figure is provided in Sec. A.1.



**Figure 5:** Real-world generalization visualizations. We show how Fail2Progress generalizes to different numbers of objects (3-6), different object shapes, and different tables.

22 We present qualitative results in Fig. 4. **Hierarchical Tabletop Organization** task (First row): The  
 23 robot is tasked with organizing the cups and capsules on another table while keeping them in a row. It  
 24 first places several capsules into their corresponding cups. In the failure case, the robot fails to recognize  
 25 the correlation between cups and capsules, resulting in the wrong organization. After learning from this  
 26 failure, Fail2Progress successfully completes this task by understanding that the capsules will move  
 27 with their corresponding cups. **Multi-object Transport** task (Second row): The robot is tasked with  
 28 packing groceries and placing them on the table. It places all four groceries inside a bag. In the failure  
 29 case, the robot places the bag on the ground instead of the table, failing the task. After fine-tuning  
 30 the model with a targeted dataset, Fail2Progress moves the bag to the table. **Constrained Packing**

task (Third row): The robot is tasked with organizing a shelf by placing a stack of cups on a constrained shelf. In the failure case, the robot fails to make all the wipes in contact to clear enough space. After learning from the failure, Fail2Progress first pushes the wipes aside in contact to create sufficient space for the cups, then places them on the shelf.

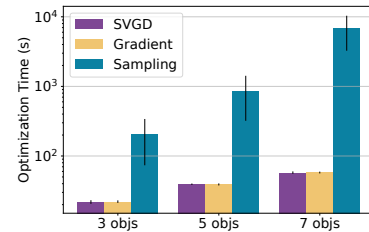
Furthermore, we demonstrate how our approach generalizes to different numbers and shapes of objects, as well as different tables, in Fig. 5. Specifically, the model is fine-tuned only on failure cases with 3 objects but is able to generalize to scenarios involving 3-6 diverse objects on two tables.

## A.2 Detailed Experimental Tasks

**Multi-object Transport** tasks the robot to transport multiple objects within a container using a single skill (e.g., carrying multiple fruits in a grocery bag). To succeed, the robot has to understand that all objects inside the container move together when the container is moved. **Hierarchical Tabletop Organization** tasks the robot to organize a table by arranging objects into a hierarchical structure (e.g., multiple objects in different cups). Success requires the robot to understand the relationships between these objects and how its skills impact future relations based on the hierarchical structure. **Constrained Packing** tasks the robot to organize objects in a constrained environment (e.g., a bookshelf). Success involves using a non-prehensile push skill to create space and then packing the remaining objects onto the shelf. In this paper, we present quantitative results for the **Multi-object Transport** and **Hierarchical Tabletop Organization** tasks, and qualitative results for the **Constrained Packing** task.

## A.3 Efficiency Experiments

We compare Fail2Progress with the two best-performing baselines, Gradient and Sampling, to assess optimization efficiency using the best-performing architecture (Points2Plans). Error bars in the figure represent standard deviations across five different random seeds. As shown in Fig. 6, Fail2Progress is significantly more efficient than Sampling. This superior efficiency is attributed to the parallel computation capabilities of SVI on GPUs. Gradient achieves comparable efficiency to Fail2Progress, but it still performs significantly worse in terms of fine-tuned model performance shown in Fig. 3 and Table 1.



**Figure 6:** Efficiency experiments show that Fail2Progress is comparable to Gradient and more efficient than Sampling. The optimization time is presented on a logarithmic scale.

## A.4 Key Findings

**Importance of Learning from Failures:** Learning from failures is essential because an initial training dataset for a skill effect model cannot capture all possible transitions in the real world. When the robot encounters novel scenarios outside the training data distributions, failures become inevitable. By learning from these failures, the robot can improve its performance more reliably and efficiently.

**Limitations of Replanning:** While Replanning can recover from certain failures, its effectiveness is inherently limited. It relies on the learned dynamics model, and any inaccuracies in the model can significantly degrade its performance.

**Effectiveness of Fail2Progress with SVI:** Through large scale comparisons, we demonstrate that Fail2Progress consistently outperforms the Sampling and Gradient baselines. This superior performance is attributed to SVI’s ability to approximate high-dimensional posterior distributions using multiple particles. In contrast, Sampling does not leverage gradient information, making it highly inefficient. It also suffers from poor exploration with high-dimensional input. While Gradient uses gradient information, it suffers from mode collapse, leading to poor performance when the posterior distribution is multi-modal.

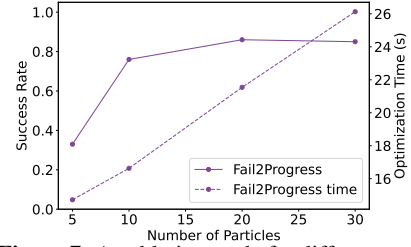
**Efficiency of Fail2Progress with SVI:** Generating additional simulation datasets from observed failures requires solving a high-dimensional and multi-modal posterior distribution inference problem. Fail2Progress, utilizing SVI, performs this task efficiently and achieves superior performance compared to Sampling. This efficiency is due to SVI’s ability to approximate complex posterior distributions in parallel, leveraging GPU computational power.

Our experiments further reveal that as few as 20 targeted simulation data points are sufficient to fine-tune the skill effect model. This efficiency stems from the pre-trained model’s ability to capture general representations that are transferable to related tasks. As a result, when the robot fails at a specific task, it can be efficiently fine-tuned on the new task to recover from failures.

**Generalization of Fail2Progress:** Through both simulation and real-world experiments, we demonstrate that our approach generalizes to varying numbers and shapes of objects, different environments (e.g., tables), and viewpoints beyond those in the fine-tuning dataset. This demonstrates that our framework does not overfit to specific scenarios but instead captures object interaction in a generalizable manner. By combining the ability to continuously learn from failures and generalize to unseen scenarios, we believe our framework can adapt to diverse and complex real-world household environments, assisting in daily tasks.

## A.5 Ablation Study

To determine the best fine-tuning dataset size, measured by the number of particles, we conduct an ablation study presented in Fig. 7. The study is performed on the **Hierarchical Tabletop Organization** task with three objects, using Points2Plans as the underlying architecture. Through the ablation study, we find that increasing the number of particles generally improves the execution success rate of Fail2Progress but also increases the optimization time. Using 20 particles achieves the best balance between performance and efficiency. Therefore, we use 20 particles for our experiments.



**Figure 7:** An ablation study for different particles.

## A.6 Detailed Simulation Results

#Objs	Base	Original	Small	Large	Replanning	Sampling	Gradient	Fail2Progress
3	Points2Plans [1]	16%	18%	21%	28%	64%	52%	<b>90%</b>
3	Stow-GNN [2]	13%	15%	19%	24%	61%	51%	<b>83%</b>
3	Binary-Pred [3]	11%	13%	15%	23%	51%	47%	<b>78%</b>
5	Points2Plans [1]	10%	13%	15%	26%	54%	45%	<b>87%</b>
5	Stow-GNN [2]	9%	10%	14%	21%	52%	43%	<b>81%</b>
5	Binary-Pred [3]	8%	8%	13%	19%	43%	37%	<b>73%</b>
7	Points2Plans [1]	8%	9%	12%	18%	42%	39%	<b>82%</b>
7	Stow-GNN [2]	8%	7%	11%	17%	41%	37%	<b>76%</b>
7	Binary-Pred [3]	5%	7%	9%	14%	30%	29%	<b>64%</b>
Average	Points2Plans [1]	11%	13%	16%	24%	53%	45%	<b>86%</b>
Average	Stow-GNN [2]	10%	11%	15%	21%	51%	44%	<b>80%</b>
Average	Binary-Pred [3]	8%	9%	12%	19%	41%	38%	<b>72%</b>

**Table 3:** Simulation experiments for the **Hierarchical Tabletop Organization** task across different numbers of objects. The comparisons demonstrate that Fail2Progress outperforms baselines by a large margin.

We provide the detailed simulation results in Table 3. Through the comparison, we find that Fail2Progress outperforms all baselines with different numbers of objects using different base architectures.



## A.7 Detailed Sim2Real Gap

We provide a detailed analysis of the Sim2Real gap for our work. The Sim2Real gap can be caused by the following reasons: (1): Perception gap: This includes differences in rendering and visualization between simulation and the real world (e.g., real-world perception is less accurate and noisier). In simulation, ground-truth object segmentation are readily available, whereas in the real world, obtaining accurate information is challenging, especially for partial views and cluttered scenes. (2): Controller mismatch: This arises due to differences in robot control between simulation and the real world. Real robots would have latency, compliance, and joint limit constraints, which are often not fully modeled in simulation. (3): Object geometry gap: Real-world objects vary in material, shape, and appearance, and often differ from their simulation twin. This discrepancy is particularly significant for deformable objects. (4): Physical modeling: The physical modeling in simulation can be inaccurate, which contributes to unrealistic physical interactions.

## A.8 Extra Related Work

Solving long-horizon manipulation tasks remains a significant challenge in the community. Traditionally, task and motion planning [4] addresses this problem by separating high-level symbolic reasoning from low-level geometric reasoning. However, TAMP methods typically rely on explicit 3D object models [5, 6, 7, 4, 8] and symbolic operators with predefined effects [9, 6, 7, 4, 8, 10]. Alternatively, recent works propose to sequence learned skills to handle geometrically complex tasks, but these approaches are also limited to hand-crafted states [11, 12, 13, 14]. A more recent study [1] introduces a method to learn the effects of skills directly from partial-view point clouds, enabling robots to reason about real-world scenarios involving hard-to-define object interactions. However, none of the existing skill effect models [11, 12, 13, 14, 1] reason about or learn from failures after deployment. Our work is the first to leverage the failure cases to improve skill effect models, thereby minimizing future failures.

## A.9 Relations Definition

We use both unary relations and binary relations in this paper.

We consider the following unary relations: (1) **Movability**: indicates whether a specific segment is movable (e.g., a table is not movable). (2) **Drawer identification**: specifies whether a segment is a drawer. (3): **Drawer state**: determines whether a drawer is open or closed.

We define the following binary relations: (1) **Spatial relations**: includes six spatial relationships- **left**, **right**, **front**, **behind**, **above**, **below**- defined following [3]. (2) **In-contact**: identifies whether two objects are in contact. Ground-truth labels for this relation are obtained directly from the IsaacGym [15] simulator. (3) **Boundary**: Specifies whether an object is on the boundary of a supporting surface (e.g., a table or shelf). We define  $\text{boundary}(A, B)$  as true if object A is above object B, the distance between A and the nearest boundary of B is less than a threshold  $\epsilon_{\text{boundary}}$ , and the dimensions of B exceed  $\epsilon_{\text{bottom}}$ . In this paper,  $\epsilon_{\text{boundary}}$  is set to 0.1m, and  $\epsilon_{\text{bottom}}$  is set to 0.2m. (4) **Inside**: indicates whether an object is inside another (e.g., a container). We define  $\text{Inside}(A, B)$  as true if the bounding box of A is completely contained within the bounding box of B.

## A.10 Skills Definition

We use the following skills in this paper.

**Pick-and-place**: This skill enables the robot to grasp an object and place it in a specific pose. If the grasp pose or placement pose is outside the robot’s current reachable space, the robot will first move its base to make it reachable before executing the arm motions. The continuous parameter encodes the difference between the placement pose and the grasp pose.

153 **Push:** This skill allows the robot to push multiple objects. The robot first moves to a pre-push pose and  
 154 then moves its end-effector along the push direction for a specific distance. The continuous parameter  
 155 encodes both the push direction and push distance.

156 **Open/Close Drawer:** This skill enables the robot to open or close a drawer. The corresponding  
 157 continuous parameters encode the distance and direction of the motion.

158 Notably, if failures occur with a newly introduced skill, a new skill effect model can be trained to handle  
 159 that skill effectively. Due to the composability of the skill effect model, the planning can incorporate  
 160 all the skills.

## 161 A.11 Details of Skill Effect Models

### 162 A.11.1 Introduction

163 Given an observation,  $O_t$ , at time  $t$ , represented as segmented point clouds, the skill effect model en-  
 164 codes  $O_t$  to a latent state  $X_t$  using  $Enc$ . Using a decoder,  $Dec$ , the latent state can be decoded to either  
 165 geometric states like object poses or symbolic states such as inter-object relations,  $\mathcal{R}$ . Furthermore, the  
 166 latent state,  $X_t$ , could also be propagated by a skill  $\phi_t(a_t)$  with a dynamics model  $Dyn$  to predict the la-  
 167 tent state  $X_{t+1}$  at the next time step. The predicted latent state  $X_{t+1}$  could also be decoded to predicted  
 168 object poses or relations. To simplify, in this paper, we use  $\gamma(\cdot)$  to represent the skill effect model com-  
 169 posing the different components  $Enc$ ,  $Dec$ , and  $Dyn$  as  $\gamma(O_t, \phi_t, a_t) = Dec(Dyn(Enc(O_t), \phi_t(a_t)))$ ,  
 170 that outputs the probabilities of different relations in  $\mathcal{R}$ , with  $\Gamma(O_0, \phi_{1:H}, a_{1:H}) = \gamma_H \circ \gamma_{H-1} \circ \dots \circ \gamma_1$ ,  
 171 representing a composition of skill effect for a skill sequence  $\phi_{1:H}(a_{1:H})$ .

### 172 A.11.2 Implementation Details

173 The input of a skill effect model (e.g., Points2Plans) is a segmented point cloud at timestep  $t$ , denoted  
 174 as  $O_t = \{O_t^0, \dots, O_t^n\}$ , where  $n$  represents the number of segments.

175 **Encoder:** We utilize PointConv [16] as the  $Enc$ . The employed PointConv architecture consists of  
 176 three set abstraction layers, each processing input point data and corresponding positional data to  
 177 produce sampled positional data and feature data as output. Both the input and output positional data  
 178 have three channels. The first abstract layer samples 128 points, with 8 neighbors per point determined  
 179 using a bandwidth of 0.1. It employs an MLP with 6 input channels (3 for positions and 3 for features),  
 180 32 output channels, and a kernel size of 1. The second layer reduces the sample size to 16 points with  
 181 16 neighbors per point and uses a bandwidth of 0.2. This layer’s MLP takes 35 input channels (3 for  
 182 positions and 32 for features), and outputs 64 channels with a kernel size of 1. The third layer is a  
 183 ”group all” layer that generates 128-dimensional features per segment, using a bandwidth of 0.4. Its  
 184 MLP has 67 input channels (3 for positions and 64 for features) and 128 output channels, with a kernel  
 185 size of 1.

186 Specifically, the encoder ( $Enc$ ) processes each segment to generate a corresponding point cloud feature,  
 187 represented as  $P_t^i = Enc(O_t^i)$ , where each point cloud feature has 128 dimensions. Additionally, we  
 188 use positional encoding in PyTorch [17] to assign a unique identifier to each object, represented as  $ID_i$ ,  
 189 which also has 128 dimensions. For each object, we concatenate the point cloud feature and positional  
 190 encoding to form  $X_t^i = P_t^i \oplus ID_i$ , resulting in a feature vector with 256 dimensions. Consequently, the  
 191 latent state is represented in an object-centric form as  $X_t = \{X_t^0, \dots, X_t^n\}$ , where each object’s latent  
 192 state contains 256 features.

193 **Dynamics:** The dynamics model ( $Dyn$ ) takes as input the latent state  $X_t$  along with the corresponding  
 194 skill and continuous parameter  $\phi_1(a_1)$ . Since  $\phi_1$  encodes discrete parameter identifying the object  
 195 to manipulate, we use positional encoding to represent the manipulated object ID as  $ID_i$ , which  
 196 has 128 features. For the continuous parameter  $a_1$ , we use a simple MLP  $MLP_{para}$  to encode a  
 197 latent continuous parameter with 128 features. The  $MLP_{para}$  consists of two layers, each with 128  
 198 neurons, using ReLU as the activation function. As a result, each skill is represented as a latent state  
 199  $AL_1 = MLP_{para}(a_1) \oplus ID_i$ , where  $AL_1$  has 256 features.

200 Once the latent skill  $AL_1$  is obtained, we use a transformer-based dynamics model,  $Dyn$ . The  
 201 transformer comprises 2 sub-encoder layers, 2 attention heads in the multi-head attention mechanism,  
 202 and a dimensionality of 256 for the input and output. Given the latent state  $X_t$  and the corresponding  
 203 latent skill  $AL_1$ ,  $Dyn$  outputs the change in each latent state, represented as  $\delta X_t$ . The predicted new  
 204 latent state is then computed as  $X_{t+1} = X_t + \delta X_t$ . For long-horizon planning, the dynamics model can  
 205 be applied recurrently as  $X_{t+H} = Dyn(X_t, AL_{1:H})$ .

206 **Decoder:** The decoder ( $Dec$ ) consists of three distinct modules: a position decoder  $Dec_p$ , a unary  
 207 relation decoder  $Dec_u$ , and a binary relation decoder  $Dec_b$ . All decoders take the latent state ( $X_t$ ) as  
 208 input.

209 **Position Decoder ( $Dec_p$ ):** The position decoder processes the predicted changes in each latent  
 210 state ( $\delta X_t$ ) and outputs the predicted changes in object positions ( $\delta p_t$ ).  $Dec_p$  is a three-layer network,  
 211 with each layer containing 64 neurons and ReLU as the activation function.

212 **Unary relation decoder ( $Dec_u$ ):** The unary relation decoder takes the absolute latent state ( $X_t$ ) as  
 213 input and outputs unary object relations.  $Dec_u$  consists of two layers, each with 64 neurons, and uses  
 214 Softmax as the activation function because unary relations are binary variables.

215 **Binary relation decoder ( $Dec_b$ ):** The binary relation decoder takes pairwise latent states ( $(X_t^i, X_t^j)$ ) as  
 216 input and outputs pairwise object relations, defined as  $Dec_b(X_t^i, X_t^j)$ .  $Dec_b$  is a three-layer network,  
 217 with each layer containing 64 neurons. Like  $Dec_u$ ,  $Dec_b$  uses Softmax as the activation function since  
 218 binary relations are also binary variables.

219 **Training Details:** We collect ground-truth data from the simulation at the current step, including point  
 220 cloud observations ( $O_t$ ), relations ( $\mathcal{R}_t$ ), and position ( $p_t$ ). Ground-truth data at the next step is collected  
 221 after executing a robot skill, which includes point cloud observations ( $O_{t+1}$ ), relations ( $\mathcal{R}_{t+1}$ ), and  
 222 position ( $p_{t+1}$ ). To train the skill effect model using the simulation dataset ( $\mathcal{D}$ ), we employ several loss  
 223 functions:

224 **Current step detection loss:** Using the current step point cloud observation ( $O_t$ ), the model ( $\Gamma$ ) predicts  
 225 current step relations ( $\hat{\mathcal{R}}_t$ ). The current step detection loss is calculated as  $L_{detection} = CE(\hat{\mathcal{R}}_t, \mathcal{R}_t)$ ,  
 226 where  $CE$  denotes the cross-entropy loss.

227 **Latent space regularization loss:** The mode encodes the observations ( $O_t, O_{t+1}$ ) into the current step  
 228 latent state ( $X_t$ ) and the next step latent state ( $X_{t+1}$ ). Using a skill,  $\Gamma$  predicts the next time step  
 229 latent state ( $X'_{t+1}$ ), where  $X'_{t+1}$  is derived from  $O_t$  and the skill while  $X_{t+1}$  derives from  $O_{t+1}$ . The  
 230 regularization loss, calculated as the L2 norm, is  $L_{regularization} = \|X_{t+1} - X'_{t+1}\|_2^2$ .

231 **Position loss:** Based on  $O_t$  and the applied skill, the model predicts the change in object positions ( $\delta p_t$ ).  
 232 The position loss compares the predicted position changes with the ground-truth position changes:  
 233  $L_{pos} = b \cdot \sqrt{a \cdot \|\delta p_t - (p_{t+1} - p_t)\|}$ . Here,  $a = 12$  and  $b = 5$  are used to balance other loss terms, as  
 234 defined in [1].

235 **Prediction loss:** To minimize the difference between predicted relations ( $R'_{t+1} = Dec_b(X'_{t+1})$ ) and  
 236 ground-truth relations ( $R_{t+1}$ ) at the next time step, we compute the prediction loss as:  $L_{prediction} =$   
 237  $CE(R'_{t+1}, R_{t+1})$ .

238 The total loss is the sum of all four terms:  $L = L_{detection} + L_{regularization} + L_{pos} + L_{prediction}$ . We  
 239 train and fine-tune the skill effect model using the Adam optimizer with a learning rate of  $1 \times 10^{-4}$ . In  
 240 this paper, we use 10 epochs for the pre-training and 200 epochs for the fine-tuning.

241 **Planning Details:** To achieve the goal relations ( $\mathcal{G}$ ), we employ a shooting-based approach to sample  
 242 the continuous parameters ( $a_{1:H}$ ) given the initial observation ( $O_1$ ) and the plan skeleton ( $\phi_{1:H}$ ). To  
 243 maximize the likelihood of achieving the goal relations, we sample a set of continuous parameters  
 244  $\{a_{1:H}^j\}_{j=1}^{K_a}$  from the robot’s workspace. Each continuous parameter sequence  $a_{1:H}^j$  is rolled out, and  
 245 we select the sequence that maximizes the probability of satisfying  $\mathcal{G}$ .

## 246 A.12 Real-to-sim details

247 For the real-to-sim process, SVI generates both simulation states and robot skills. The simulation states  
248 specify the pose of each object.

249 To create a simulation scene, we assume a set of object shape priors, including cuboids, open boxes,  
250 shelves, drawers, and tables. Based on the semantics of each segment in the observation, our method  
251 selects the appropriate object shape prior. The bounding box of each segment determines the dimension  
252 of the corresponding object in the simulation. By combining these dimensions with the object poses,  
253 we can construct the simulation scene.

254 For the robot skills, we directly execute the parameterized skills within the simulation, starting from  
255 the initial scene. During the process, we could record point clouds and object relations both before and  
256 after manipulation. Combined with the executed robot skills, we can generate a fine-tuning dataset to  
257 refine the skill effect model.

258 Note that we select this bounding-box approximation for the real-to-sim approach due to efficiency  
259 considerations. However, Fail2Progress can compliment other real-to-sim approaches [18, 19, 20].

## 260 A.13 Stein Update Details

### 261 A.13.1 Generating State Samples

262 First, we aim to solve for the simulation state set  $S^+$ . Here we want to find samples  $q(S) = \{s_i^+\}_{i=1}^M$   
263 that approximate the posterior distribution  $P(r^F | O^+ = \xi(S)O^F)P(S)$ , where  $P(S)$  is a uniform prior  
264 over all feasible simulation states. The posterior distribution ensures that the transformed point clouds  
265 match the relations in the failure case  $r^F$ . This defines the following variational inference problem:

$$\underset{q(S)}{\operatorname{argmin}} D_{KL} \left( q(S) \parallel \Gamma(r^F | O^+ = \xi(S)O^F)P(S) \right) \quad (2)$$

266 For each state particle  $s_i^+$ , the Stein update term is:

$$\Phi(s_i^+) = \frac{1}{M} \sum_{j=1}^M [k(s_j^+, s_i^+) \nabla_{s_j^+} \ln P(\mathcal{R}^F | \xi(s_j^+)O^F) + \nabla_{s_j^+} k(s_j^+, s_i^+)] \quad (3)$$

267 where  $k(s_j^+, s_i^+)$  is a kernel function that defines the similarity between different particles. The first  
268 term in Eq. 3 represents an attractive force that pushes the particles to move in a direction based on  
269 the gradient while the second term is a repulsive term that prevents the particles from collapsing. This  
270 update can generate object states that match the failure case while ensuring diversity over object states.

### 271 A.13.2 Generating Action Samples

272 Given our state samples generated by using Stein variational inference to approximate the distribution  
273 in Eq. 1b, we can now turn our attention to solving for the action set  $A^+$ . To formulate this problem we  
274 make use of the generalized Bayesian inference framework outlined above. Here we define the loss  
275 function,  $\mathcal{L}$ , to be the entropy loss defined in Eq. 1a and let  $\beta = 1$ . Note that the variational distribution  
276  $q(A) = \{(s_i^+, a_i^+)\}_{i=1}^M$ , however we keep the values of  $s_i^+$  fixed and search only over actions. This  
277 defines the following variational inference problem:

$$\underset{q(A)}{\operatorname{argmin}} \mathbb{E}_{s^+, a^+ \in q(A)} \left[ \prod_{r \in \mathcal{R}^F} -H(\Gamma(r | \xi(s^+)O^F, \phi^F, a^+, D)) \right] + D_{KL} \left( A^+ \parallel P(A) \right) \quad (4)$$

278 where  $P(A)$  is uniform prior over actions.

279 The Stein update term for the action particles  $a_i^+$  is:

$$\Phi(a_i^+) = \frac{1}{M} \sum_{j=1}^M [k(a_j^+, a_i^+) \cdot \nabla_{a_j^+} \ln H(\Gamma(\mathcal{R}^F | \xi(s_j^+)O^F, \phi^F, a_j^+, D)) + \nabla_{a_j^+} k(a_j^+, a_i^+)] \quad (5)$$

### A.13.3 Implementation Details of SVI

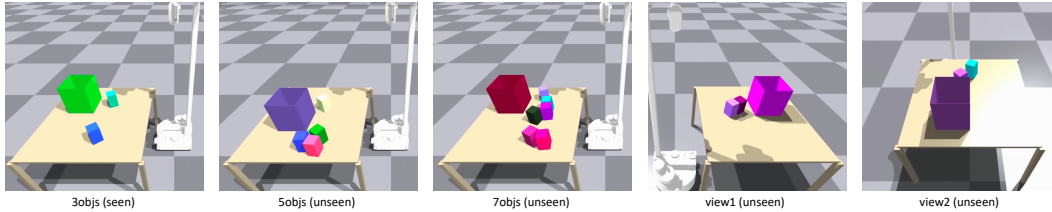
We use RBF kernels for SVI and follow previous works [21, 22] by applying the median heuristics to determine the kernel bandwidth. Additionally, the step size is optimized using the Adam optimizer [23].

### A.14 Detailed Generalization Experiments

**Generalization Evaluation:** We assess the generalization capability of Fail2Progress compared to the Gradient and Sampling baselines in the **Multi-object Transport** task. First, we evaluate generalization to an unseen number of objects, as shown in Table 4. The model is fine-tuned only on scenarios with 3 objects and tested on unseen scenarios with 5 and 7 objects. While all approaches experience some performance degradation, Fail2Progress maintains strong performance, even in scenarios with 7 objects. In contrast, Gradient and Sampling perform poorly, particularly in the 7-object scenarios. Next, we assess generalization to unseen viewpoints, also shown in Table 4. Fail2Progress demonstrates robust performance across two unseen viewpoints and consistently outperforms Gradient and Sampling baselines. For both evaluations, we perform 100 trials per approach for each evaluation metric. Visualizations of these generalization scenarios are provided in Fig. 8.

Generalization Scenarios	3objs ↑	5objs ↑	7objs ↑	view1 ↑	view2 ↑
Fail2Progress	87%	81%	71%	83%	85%
Gradient	51%	40%	18%	42%	44%
Sampling	62%	45%	23%	51%	47%

**Table 4:** Generalization results for the **Multi-object Transport** task. We show the generalization capability of Fail2Progress with respect to different numbers of objects and different viewpoints (5objs, 7objs, view1, and view2 are unseen in the training dataset). Evaluations on unseen objects and unseen viewpoints show that Fail2Progress performs well and outperforms the best-performing baselines (Sampling and Gradient).



**Figure 8:** Visualizations of simulation generalization scenarios. Fail2Progress, fine-tuned on a dataset with 3 objects, successfully generalizes to scenes with 5 and 7 objects. Additionally, Fail2Progress demonstrates generalization to two unseen viewpoints.

### A.15 Hardware Information

All the skill effect models are trained and fine-tuned on a standard workstation with an NVIDIA GeForce RTX 3090 Ti GPU. All the real-world experiments are conducted with a Stretch-re2 from Hello-Robot.

### References

- [1] Y. Huang, C. Agia, J. Wu, T. Hermans, and J. Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics. *arXiv preprint arXiv:2408.14769*, 2024.
- [2] H. Chen, Y. Niu, K. Hong, S. Liu, Y. Wang, Y. Li, and K. R. Driggs-Campbell. Predicting object interactions with behavior primitives: An application in stowing tasks. In *7th Annual Conference on Robot Learning*, 2023. URL <https://openreview.net/forum?id=VH6WIPF4Sj>.
- [3] C. Paxton, C. Xie, T. Hermans, and D. Fox. Predicting Stable Configurations for Semantic Placement of Novel Objects. In *Conference on Robot Learning (CoRL)*, 11 2021. URL <https://arxiv.org/abs/2108.12062>.



- [4] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021. URL <https://arxiv.org/abs/2010.01083>.
- [5] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer. Search-Based Task Planning with Learned Skill Effect Models for Lifelong Robotic Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. URL <https://arxiv.org/abs/2109.08771>.
- [6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020. URL <https://arxiv.org/abs/1802.08705>.
- [7] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Sample-based methods for factored task and motion planning. In *Robotics: Science and Systems*, 2017. URL <https://dspace.mit.edu/bitstream/handle/1721.1/137701/garrett-rss17.pdf?sequence=2&isAllowed=y>.
- [8] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez. Learning to guide task and motion planning using score-space representation. *The International Journal of Robotics Research*, 38(7):793–812, 2019. URL <https://arxiv.org/abs/1807.09962>.
- [9] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1940–1946. IEEE, 2022.
- [10] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Proceedings of Robotics: Science and Systems*, 2020. URL <https://arxiv.org/abs/2006.05398>.
- [11] C. Agia, T. Migimatsu, J. Wu, and J. Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.
- [12] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.
- [13] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.
- [14] S. Cheng and D. Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 2023.
- [15] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Advances in Neural Information Processing Systems*, 2021. URL <https://sites.google.com/view/isaacgym-nvidia>.
- [16] W. Wu, Z. Qi, and L. Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019. URL <https://arxiv.org/abs/1811.07246>.
- [17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

- 359 [18] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta.  
360 Urdformer: A pipeline for constructing articulated simulation environments from real-world  
361 images. *arXiv preprint arXiv:2405.11656*, 2024.
- 362 [19] Z. Mandi, Y. Weng, D. Bauer, and S. Song. Real2code: Reconstruct articulated objects via code  
363 generation. *arXiv preprint arXiv:2406.08474*, 2024.
- 364 [20] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality  
365 through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint*  
366 *arXiv:2403.03949*, 2024.
- 367 [21] J. Pavlasek, S. R. Lewis, B. Sundaralingam, F. Ramos, and T. Hermans. Ready, set, plan! planning  
368 to goal sets using generalized bayesian inference. In *7th Annual Conference on Robot Learning*,  
369 2023. URL <https://openreview.net/forum?id=5JMGq83yf1N>.
- 370 [22] D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic.  
371 *arXiv preprint arXiv:1707.07269*, 2017.
- 372 [23] D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*,  
373 2014.