A. Additional Method and Implementation Details

This section provides additional implementation details of the AKR-based planning pipeline, covering the AKR inversion and assembly procedure and the collision-sphere fitting process that supports efficient GPU-accelerated collision checking.

A.1. Detailed Implementation for AKR Inversion and Assembly

The AKR inversion and assembly workflow comprises four steps: (a) URDF preprocessing and kinematic inversion, (b) collision sphere generation, (c) object-to-robot attachment, and (d) selective self-collision checking. The process begins by applying a uniform scaling factor to the object URDF to ensure physical consistency—particularly important when working with grasp datasets derived from point cloud or RGB-D data, where object models are often normalized to fit within a standardized bounding volume. Since each link may contain multiple geometric components, these are first merged into a single mesh and scaled accordingly. As mesh scaling alters the spatial relationships in the original kinematic chain $\mathcal{K}_{\text{raw}}$, all joint origins are subsequently recalculated to preserve valid relative transformations, yielding $\mathcal{K}_{\text{scaled}}$.

The tip link ℓ_{tip} corresponding to the grasping point is identified along with its parent joint. To attach the object as an extension of the robot, the kinematic structure is inverted by reassigning ℓ_{tip} as the new base link; the joint hierarchy from the original base ℓ_{base} to the tip is reversed while the rest of the tree is preserved, yielding $\mathcal{K}_{\text{inv}}$.

The attachment transformation between the robot and the object is computed from the grasp pose and the object's FK. Let $T_{\text{tip}}^{\text{base}}$ denote the pose of ℓ_{tip} relative to ℓ_{base} under the joint configuration $\boldsymbol{q}_{\text{init}}$ corresponding to the grasp pose:

$$T_{\text{tip}}^{\text{base}} = \text{FK}_{\mathcal{K}_{\text{scaled}}}(\boldsymbol{q}_{\text{init}}, \ell_{\text{tip}}). \tag{A1}$$

Given the grasp pose $T_{\text{tcp}}^{\text{base}}$ specifying the robot's Tool Center Point (TCP) frame relative to the object's base link, the final attachment transformation is:

$$T_{\text{tip}}^{\text{tcp}} = \left(T_{\text{tcp}}^{\text{base}}\right)^{-1} \cdot T_{\text{tip}}^{\text{base}}. \tag{A2}$$

This transformation is applied as a fixed joint between the robot's TCP and the object's new base link, yielding a unified kinematic model in which the robot and object form a single tree structure.

Finally, the additional self-collision link pairs introduced by the attached object are identified selectively, avoiding full recomputation of the entire self-collision matrix and thereby reducing computational overhead while maintaining sufficient coverage for motion planning. Detailed pseudocode is provided in Algorithm 1.

Algorithm 1: AKR Construction Procedure

Function `akr_construction` (*robot, object, init_state, scaling_factor, sphere_params, grasp_pose, grasp_link, sample_n*) **:**
- `update_object_state`(object, init_state);
- **foreach** *link* ∈ *object.links* **do**
  - merged_mesh ← `merge`(link.geometries);
  - scaled_mesh ← `scale`(merged_mesh, scaling_factor);
  - link.visual ← scaled_mesh;
  - link.collision ← `sphere_fit`(scaled_mesh, sphere_params);
- **end**
- **foreach** *joint* ∈ *object.joints* **do**
  - tf ← `get_tf`(object, joint.child, joint.parent);
  - joint.origin ← `update`(tf, scaling_factor);
- **end**
- scaled_object ← object;
- fk_pose ← `fk`(scaled_object, grasp_link);
- attached_origin ← grasp_pose^{-1} · fk_pose;
- inversed_object ← `inverse`(scaled_object, grasp_link);
- akr ← `attach`(robot, inversed_object, attached_origin);
- added_link_pairs ← `filter`(akr.link_pairs, robot.link_pairs);
- sampled_cfg ← `sample_cfg`(akr, sample_n);
- added_collision_pairs ← `check_collision`(added_link_pairs, sampled_cfg);
- akr.collision_pairs ← `union`(robot.collision_pairs, added_collision_pairs);

A.2. Collision-Sphere Fitting Procedure

As the manipulated object becomes part of the robot representation via AKR, its collision geometry is approximated using a set of spheres, aligning with cuRobo's sphere-based representation and enabling GPU-accelerated parallel collision checking. The fitting procedure proceeds as follows.

1. **Mesh preprocessing:** Multiple geometric components within each link are merged into a single mesh.
2. **Scaling and voxelization:** The merged mesh is uniformly scaled down slightly to ensure conservative collision checking, then voxelized into discrete occupied volumetric regions forming an occupancy grid.
3. **Sphere fitting:** For each occupied voxel region, a sphere is positioned at the voxel centroid with diameter equal to the voxel edge length.
4. **Spatial alignment:** When voxelization introduces trans-

lational offsets, the centroid of the fitted sphere cloud is realigned with that of the original mesh to mitigate discretization drift.

The resulting sphere-based representation ensures computationally efficient collision queries throughout trajectory optimization.

B. Interactive Scene Construction and Data Generation Details

B.1. Generation of Interactive Scenes

`AutoMoMa` leverages two complementary sources of interactive household environments: manually curated scenes and procedurally generated layouts.

The first source consists of 30 high-fidelity scenes derived from AI2-THOR [25] (shown in Fig. A2a), in which static objects such as microwave ovens, dishwashers, and cabinets are manually replaced with functionally equivalent articulated counterparts from the SAPIEN dataset [47]. Replacements are carefully positioned to respect semantic context and physical plausibility, yielding coherent scenes suitable for targeted evaluation.

The second source comprises 300 diverse scenes generated using a custom Infinigen-based pipeline [34]. Articulated object models are converted into static placeholder assets and imported into Infinigen; the generator supports controllable parameters—including object selection, placement optimization, and layout sparsity—to produce manipulation-friendly configurations. Generated layouts are exported in USD format for compatibility with GPU-accelerated planning in cuRobo [42], after which placeholders are replaced with the original articulated objects to restore full kinematic fidelity (Fig. A2b).

Together, these 30 curated and 300 procedural scenes provide a rich and diverse foundation for training and evaluating whole-body mobile manipulation policies across a wide range of household contexts.

B.2. Additional Details of Data Generation Pipeline

Tab. A1 details the configuration of the `AutoMoMa` trajectory generation pipeline. Structured voxel or mesh collision fields are constructed with a safety margin of 0.3 m and a voxelized distance resolution of 2 cm. Both IK and trajectory meshes are sampled at 5 cm pitch to balance precision and scalability across hundreds of thousands of trajectories.

During IK sampling and filtering, up to 10 retries are allowed per attempt, and a hybrid clustering strategy combining k-means and Affinity Propagation (AP) is applied to select diverse candidate configurations. Grasp proposals (20 per object) and open-angle sampling (5–10 degrees) determine the target end-effector poses for trajectory goals, with position and rotation tolerances of 1 cm ensuring consistency across generated sequences.

C. Training Setup and Experimental Details

This section summarizes the complete experimental configuration for downstream policy learning. We evaluate three policy baselines—3D Diffusion Policy (DP3), image-based Diffusion Policy (DP), and Action Chunking with Transformers (ACT)—describing for each the observation and action structure, model architecture, and optimization setup.

C.1. 3D Diffusion Policy (DP3)

Observation, action, and temporal structure DP3 [49] represents the scene with a downsampled point cloud of 4,096 points (XYZ coordinates) together with an 11-dimensional proprioceptive state capturing mobile base and arm joint configurations. A temporal history of three consecutive observation steps is concatenated before being passed to the encoder, providing short-horizon context important for resolving the kinematic coupling between base placement and arm motion. The model predicts future actions over a horizon of eight steps, of which six are executed per re-planning cycle, with an 11-dimensional action vector covering the full whole-body DoF.

Policy architecture A PointNet-based encoder with optional spatial cropping (crop shape $[80 \times 80]$) accepts 3-channel input and produces a 64-dimensional latent with LayerNorm applied both within and at the output; color and normal channels are disabled to ensure simulation-to-real consistency. The latent conditions a U-Net diffusion backbone via Feature-wise Linear Modulation (FiLM) units applied at the downsampling, bottleneck, and upsampling stages, with channel dimensions $[512, 1024, 2048]$, kernel size 5, group normalization with 8 groups, and a 64-dimensional diffusion-step embedding.

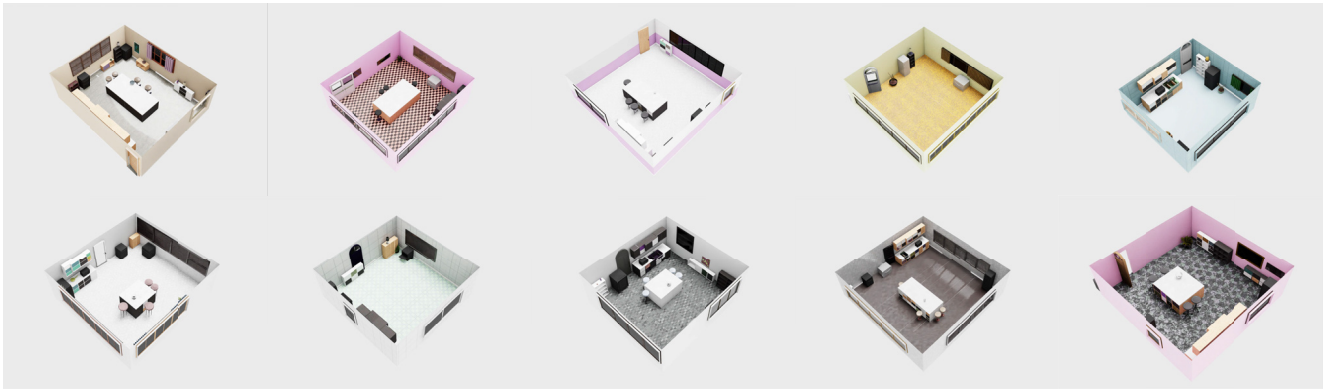
Diffusion scheduler Training uses 100 diffusion steps with a DDIMScheduler, squared-cosine beta schedule, and sample-prediction parameterization. Inference requires only 10 denoising steps, providing a favorable accuracy–speed trade-off for real-time motion generation.

Optimization and training All hyperparameters are listed in Tab. A2. AdamW is used with $\mathrm{lr} = 1 \times 10^{-4}$, betas $(0.95, 0.999)$, weight decay 1×10^{-6}, a cosine-decay schedule with 500 warmup steps, and a batch size of 512. Exponential Moving Average (EMA) with power 0.75 and maximum value 0.9999 stabilizes training. All experiments use 300 epochs with a fixed random seed of 42.

C.2. Image-based Diffusion Policy (DP)

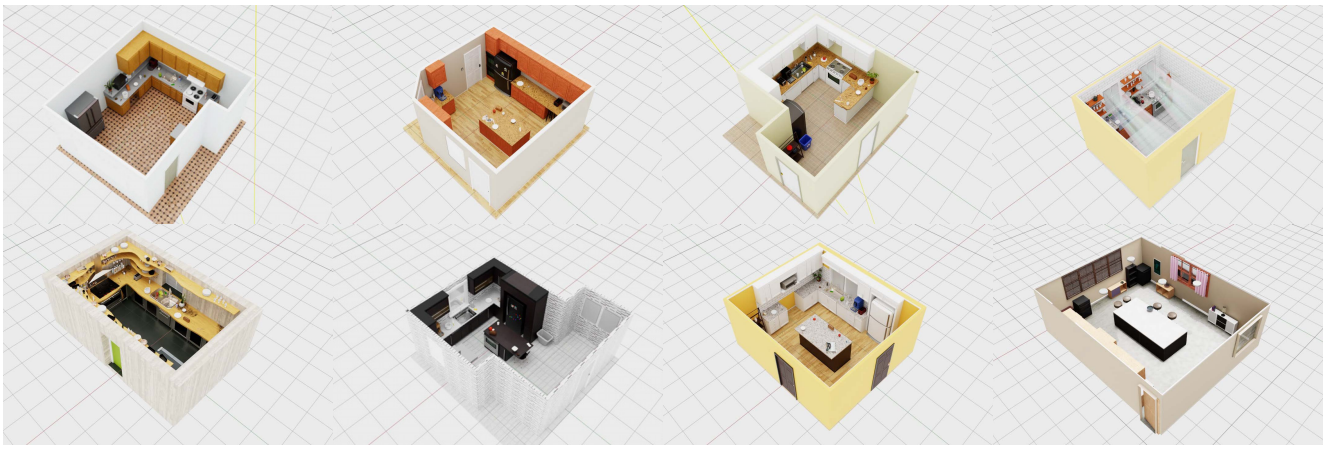
Observation, action, and temporal structure DP [6] replaces the point cloud with raw RGB observations from three camera views (ego top-down, ego wrist, and fixed), retaining the same 11-dimensional proprioceptive state, three-step temporal history, 8-step prediction horizon, 6 executed steps per cycle, and 11-dimensional action vector as DP3.

(a) Training scenes (30 scenes) used for downstream policy training.



(b) Test scenes (10 scenes) used for downstream evaluation.

Figure A1. **Downstream scene split.** 30 training and 10 test scenes are sampled from the 300 procedurally generated scenes for policy training and evaluation.



(a) AI2-THOR (iTHOR) scene with SAPIEN asset replacement.

(b) Procedurally generated scene from Infinigen.

Figure A2. **Two scene sources in `AutoMoMa`.** (a) AI2-THOR scenes with articulated objects substituted from SAPIEN. (b) Procedurally generated layouts from Infinigen.

Policy architecture Each camera stream is independently encoded by a ResNet-18 backbone with group normalization, standard ImageNet normalization, and random crop augmentation; weights are not shared across views and pretrained weights are not used. Concatenated multi-view features condition a U-Net diffusion backbone with channel dimensions $[256, 512, 1024]$, kernel size 5, 8 normalization groups, a 128-dimensional diffusion-step embedding, and conditional prediction scaling.

Diffusion scheduler DP uses a DDPMScheduler with a squared-cosine beta schedule, $\beta_{\text{start}} = 0.0001$, $\beta_{\text{end}} = 0.02$, epsilon-prediction parameterization, and clipped samples. All 100 denoising steps are performed at inference, which is feasible since the image-based variant carries no point cloud preprocessing overhead.

Optimization and training All hyperparameters are listed in Tab. A3. AdamW is used with $\text{lr} = 1 \times 10^{-4}$, betas $(0.95, 0.999)$, weight decay 1×10^{-6}, a cosine-decay schedule with 500 warmup steps, and a batch size of 128. Half-precision training (`fp16`) is enabled to reduce memory consumption. EMA (power 0.75, max 0.9999) is applied identically to the DP3 setup, for 300 epochs.

C.3. Action Chunking with Transformers (ACT)

Observation, action, and temporal structure ACT [13] operates on the same three camera views as DP. Rather than maintaining a temporal observation history, ACT encodes a single observation and predicts a *chunk* of 6 consecutive actions in one forward pass. Temporal aggregation across overlapping chunks is disabled; each chunk is executed open-loop before the next prediction. Mobile base commands are in a relative coordinate frame, and the 11-dimensional action vector covers the full whole-body DoF.

Policy architecture Each image is encoded by a ResNet-18 backbone with sinusoidal position embeddings, feeding visual tokens into a CVAE-based Transformer encoder–decoder. The encoder has 4 layers with 8 attention heads and a 512-dimensional hidden space; the decoder has 7 layers with the same configuration and a 3,200-dimensional feedforward width. Dropout of 0.1 is applied throughout, with pre-norm, dilation, and mask-prediction

disabled. A KL-divergence weight of 1.0 regularizes the CVAE latent.

Optimization and training All hyperparameters are listed in Tab. A4. The backbone and Transformer are jointly optimized with AdamW at $\text{lr} = 1 \times 10^{-5}$ and global weight decay 1×10^{-4}. Training runs for 2,000 epochs on 100 demonstrations per task configuration with episode length 1,000 steps. No learning rate scheduler or EMA is applied; the longer training duration and lower learning rate suffice for stable convergence.

D. Qualitative Results, Failure Analysis, and Real-World Validation

D.1. Representative Success and Failure Cases of Trajectory Generation

Failure due to collision Fig. A3 illustrates a collision failure. Sphere-based geometry approximations, while critical for GPU acceleration, occasionally fail to capture the original shape precisely, causing unintended collisions during planning.

Failure due to constraint violation Fig. A4 depicts a fixed-base constraint violation, where the planned trajectory involves movements inconsistent with the object's environmental attachment. Ensuring strict constraint adherence remains challenging in complex manipulation scenarios.

Successful trajectory Fig. A5 illustrates a successful trajectory in which both collision and motion constraints are correctly respected, validating the pipeline's capacity to generate physically plausible whole-body motions across diverse tasks.

D.2. Representative Inference Success and Failure Cases

Experiments are conducted on 30 training scenes and 10 held-out test scenes; the number of scenes is varied per experimental setting to ensure fair evaluation. The three success cases illustrate that with sufficient scale and diversity, the policy learns robust whole-body motion across varied layouts and object configurations. The failure case shows that errors arise not from misunderstanding articulation structure, but from pose inaccuracies that accumulate over execution.

D.3. Real-World Validation

We validate the planning pipeline on a physical UR5-Ridgeback system comprising two UR5 manipulators mounted on a Clearpath Ridgeback mobile base. On both drawer opening and cabinet door opening tasks, the robot executes the planned trajectories smoothly, accurately reproducing simulation-generated motions without collisions or constraint violations (Fig. A8).

E. Additional Ablation Experiments

E.1. Scaling Analysis up to 100k Trajectories

To systematically analyze the effect of dataset scaling, we evaluate the DP3 policy trained on subsets of 10k, 30k, 50k, and 100k trajectories, spanning five SAPIEN objects across 30 scenes. Fig. 8 details the per-object success rates under both seen and unseen configurations across these varied data scales.

As the trajectory count increases, we observe a consistent upward trend in success rates for both seen and unseen settings across all evaluated objects. At the maximum scale of 100k trajectories, the policy effectively handles multiple object geometries, achieving over 50% success on seen environments for four of the five objects. Notably, the performance on unseen configurations scales closely with the seen performance, demonstrating improved zero-shot generalization as data volume expands.

Performance variance across object categories remains largely correlated with physical constraints. For instance, the consistently lower success rate for object ID 46197 across all data scales corresponds to its unique articulation limits, which restrict the robot's valid manipulation workspace. Overall, this scaling analysis indicates that increasing trajectory density systematically improves the policy's capacity to handle varied kinematic structures and generalize to unseen states.

E.2. Impact of Unique Start States on Training Performance

A central finding is that the number of unique start states has a direct and significant impact on policy generalization: even with a fixed trajectory count, increasing the diversity of initial configurations consistently improves downstream success rates (Fig. A10).

Effect of start-state diversity on the 1,000-trajectory benchmark Evaluating training sets with 50, 100, 500, and 1,000 unique start states on a fixed benchmark of 1,000 trajectories, a clear monotonic improvement emerges despite all configurations achieving a low absolute success rate of around 5%. Broader coverage of initial base and arm configurations enables the diffusion policy to learn more reliable whole-body motion even under challenging test conditions.

Start-state diversity under fixed dataset sizes Using 6,400 trajectories, we compare a variant with 2,713 unique start states (raw distribution) against one with 5,244 unique start states (subsampled from 12,800 generated trajectories). The latter shows a clear improvement, demonstrating that start-state variety matters independently of dataset size. Training on the full 12,800 trajectories yields the highest performance, reflecting the combined benefit of higher data volume and richer start-state coverage.

E.3. Pick Task Performance

We evaluate AutoMoMa on a rigid-object Pick task by training a policy on 1,000 AutoMoMa trajectories and comparing against a baseline trained on an equal amount of MoMaGen [28] data. AutoMoMa achieves a success rate of **51.92%** at the reaching stage, outperforming the MoMaGen baseline at **35%**, suggesting that physically valid whole-body motions are critical for sample-efficient policy learning. Representative inference results are shown in Fig. A11.

E.4. Impact of Grasp-Switching (*vs.* Fixed-Grasp)

We compare trajectories generated with a fixed grasp against those allowing grasp-switching when beneficial (Fig. A12). Grasp-switching enables the robot to change the grasp pose mid-task, increasing the achievable opening angle by avoiding link collisions. Trajectories with grasp-switching consistently attain larger maximum opening angles than fixed-grasp baselines, demonstrating its importance in constrained whole-body manipulation scenarios.

Table A1. Hyperparameters for AutoMoMa data generation.

Hyperparameter	Value
Environment and Collision	
Object offset base (quaternion)	[0, 0, 0, 1, 0, 0, 0]
Expanded dimension	0.3
Disable collision	False
Collision type	Voxel / Mesh
IK mesh pitch	0.05
Trajectory mesh pitch	0.05
Voxel dimensions	[5.0, 5.0, 5.0]
Voxel size	0.02
IK and Trajectory Generation	
Maximum retries	10
k-means clusters	500
AP clusters upper bound	80
AP clusters lower bound	10
AP fallback clusters	30
Position norm tolerance	0.01
Rotation norm tolerance	0.01
Grasp poses per object	20
Open angles	4–10
Cameras and Sensing	
Ego top-down camera	True
Ego wrist camera	True
Fixed camera	True
Camera resolution	[320, 240]
Camera frequency	30 Hz
Focal length (ego top-down)	50
Focal length (ego wrist)	15
Focal length (fixed)	50
Downsample point cloud	True
Downsampled point count	4096

Table A2. Hyperparameters for DP3 [49].

Hyperparameter	Value
Observation and Action	
Number of points in point cloud	4096
Point feature dimension	3
Proprioceptive input dim	11
History length (observation steps)	3
Prediction horizon	8
Number of action steps	6
Action dim	11
Point Cloud Encoder	
Use point crop	True
Crop shape	[80, 80]
PointNet input channels	3
PointNet output dim	64
PointNet uses LayerNorm	True
PointNet final norm	LayerNorm
Use normal channel	False
Use point colors	False
PointNet type	pointnet
Diffusion Backbone	
Condition type	FiLM
Use down/mid/up conditioning	True / True / True
U-Net down dims	[512, 1024, 2048]
U-Net kernel size	5
U-Net number of groups	8
Diffusion step embedding dim	64
Noise scheduler	DDIMScheduler
Number of training diffusion steps	100
Beta schedule	squaredcos_cap_v2
Prediction type	sample
Denoise steps per inference	10
Optimization and Training	
Optimizer	AdamW
Learning rate	1.0×10^{-4}
Adam betas	(0.95, 0.999)
Adam ϵ	1.0×10^{-8}
Weight decay	1.0×10^{-6}
Batch size	512
Number of data loader workers	8
Number of epochs	300
Learning rate scheduler	Cosine decay
Learning rate warmup steps	500
Gradient accumulation steps	1
Use EMA	True
EMA inv_gamma	1.0
EMA power	0.75
EMA min / max value	0.0 / 0.9999
Random seed	42

Table A3. Hyperparameters for DP [6].

Hyperparameter	Value
Observation and Action	
Number of camera views	3
Proprioceptive input dim	11
History length (observation steps)	3
Prediction horizon	8
Number of action steps	6
Action dim	11
Observation as global condition	True
Image Encoder	
RGB backbone	ResNet-18
Pretrained weights	None
Random crop augmentation	True
Use group normalization	True
Share RGB model across views	False
ImageNet normalization	True
Diffusion Backbone	
U-Net down dims	[256, 512, 1024]
U-Net kernel size	5
U-Net number of groups	8
Diffusion step embedding dim	128
Conditional predict scale	True
Noise scheduler	DDPMScheduler
Number of training diffusion steps	100
Beta start / end	0.0001 / 0.02
Beta schedule	squaredcos_cap_v2
Prediction type	epsilon
Clip sample	True
Denoise steps per inference	100
Optimization and Training	
Optimizer	AdamW
Learning rate	1.0×10^{-4}
Adam betas	(0.95, 0.999)
Adam ϵ	1.0×10^{-8}
Weight decay	1.0×10^{-6}
Batch size	128
Number of data loader workers	0
Number of epochs	300
Learning rate scheduler	Cosine decay
Learning rate warmup steps	500
Gradient accumulation steps	1
Mixed precision	fp16
Use EMA	True
EMA inv_gamma	1.0
EMA power	0.75
EMA min / max value	0.0 / 0.9999
Random seed	42

Table A4. Hyperparameters for ACT [13].

Hyperparameter	Value
Observation and Action	
Number of camera views	3
Action dim	11
Action chunk size	6
Temporal aggregation	False
Mobile base mode	relative
Model Architecture	
Image backbone	ResNet-18
Position embedding	sine
Hidden dim	512
Feedforward dim	3200
Number of attention heads	8
Encoder layers	4
Decoder layers	7
Dropout	0.1
Pre-norm	False
Dilation	False
Masks	False
Optimization and Training	
Optimizer	AdamW
Learning rate	1.0×10^{-5}
Backbone learning rate	1.0×10^{-5}
Weight decay	1.0×10^{-4}
KL weight	1.0
Number of epochs	2000
Number of episodes per task	100
Episode length	1000
Random seed	0

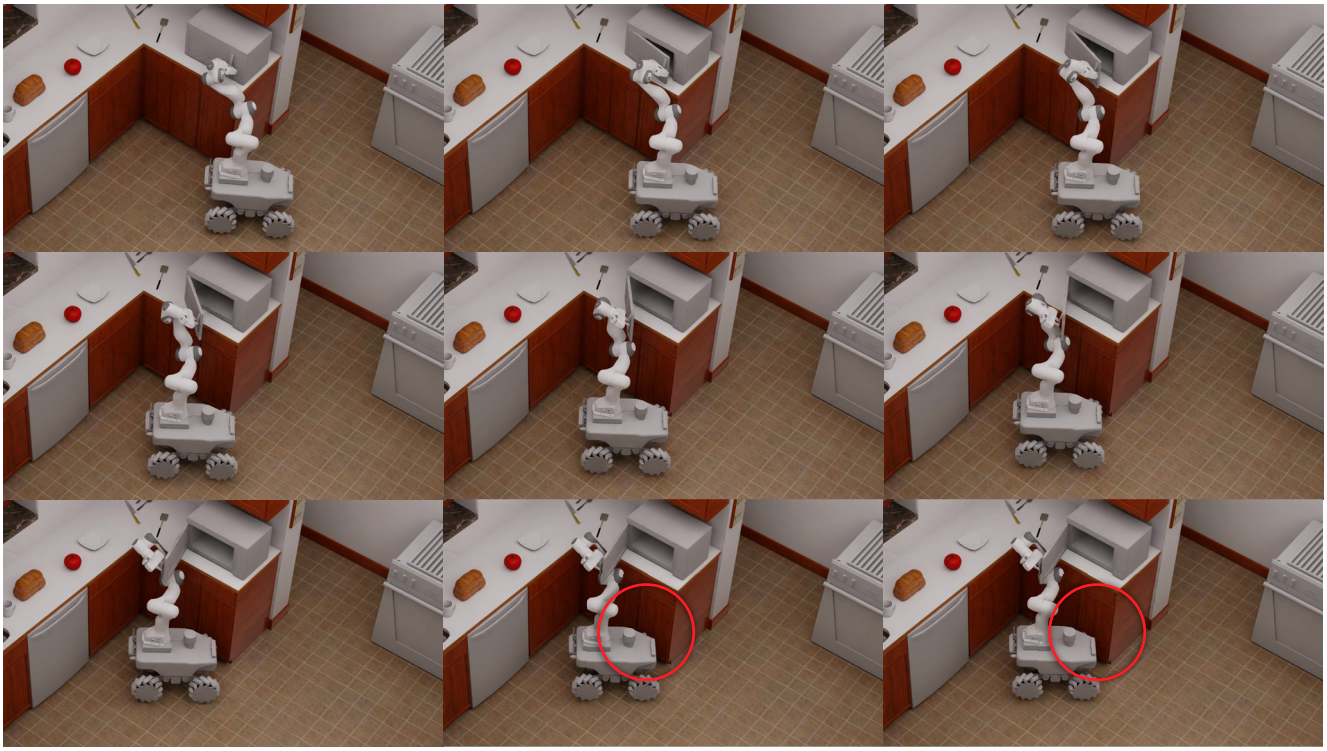


Figure A3. **Trajectory failure caused by collision.** Sphere-based geometry approximations occasionally fail to capture the original shape precisely, resulting in unintended collisions during planning.

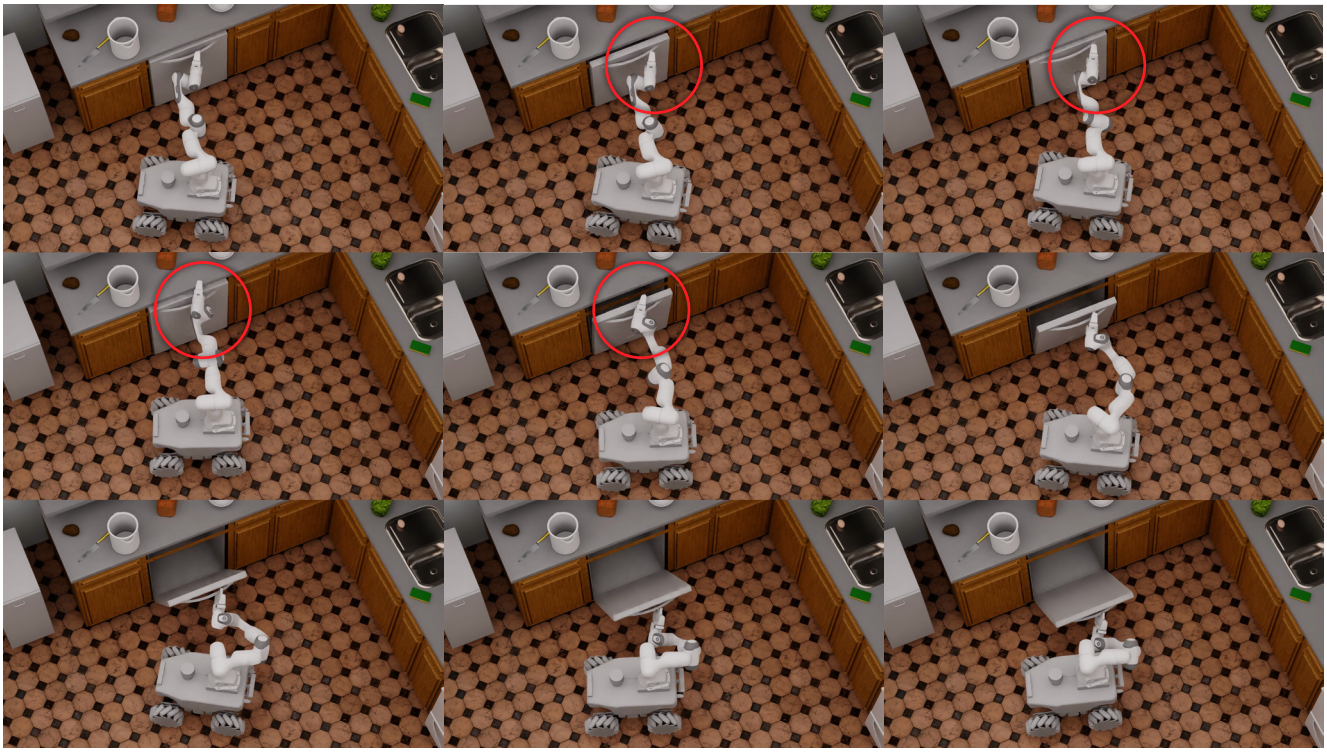


Figure A4. **Trajectory failure caused by fixed-base constraint violation.** The planned trajectory erroneously involves movements inconsistent with the object's fixed-base constraint.

Figure A5. **Representative successful trajectory.** The planned trajectory correctly respects both collision and motion constraints, producing a physically plausible whole-body motion.

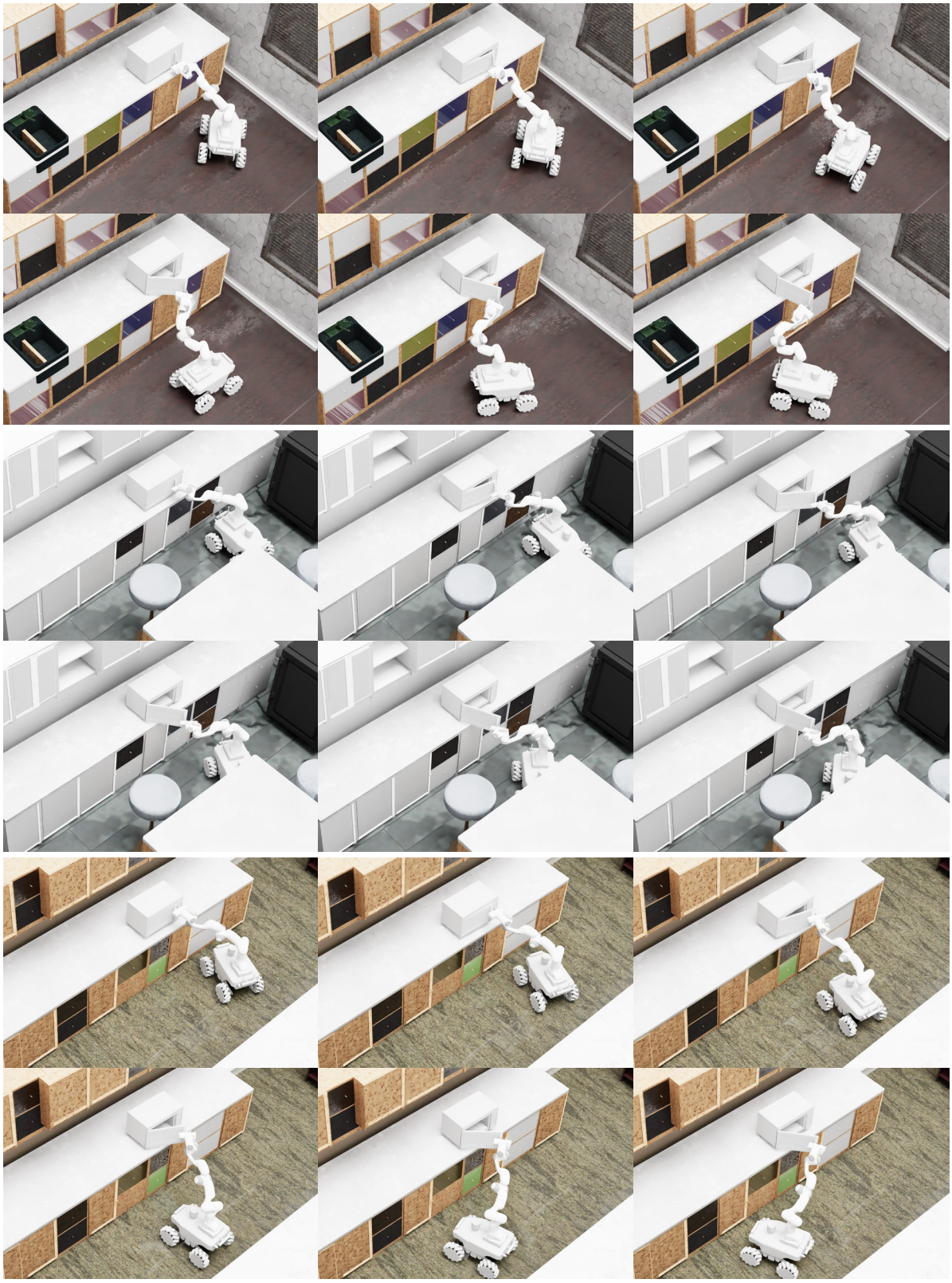


Figure A6. **Representative successful inference trajectories across diverse scenes.** Trained on large-scale `AutoMoMa` data, DP3 [49] produces feasible whole-body trajectories across varied spatial layouts and articulated-object configurations.

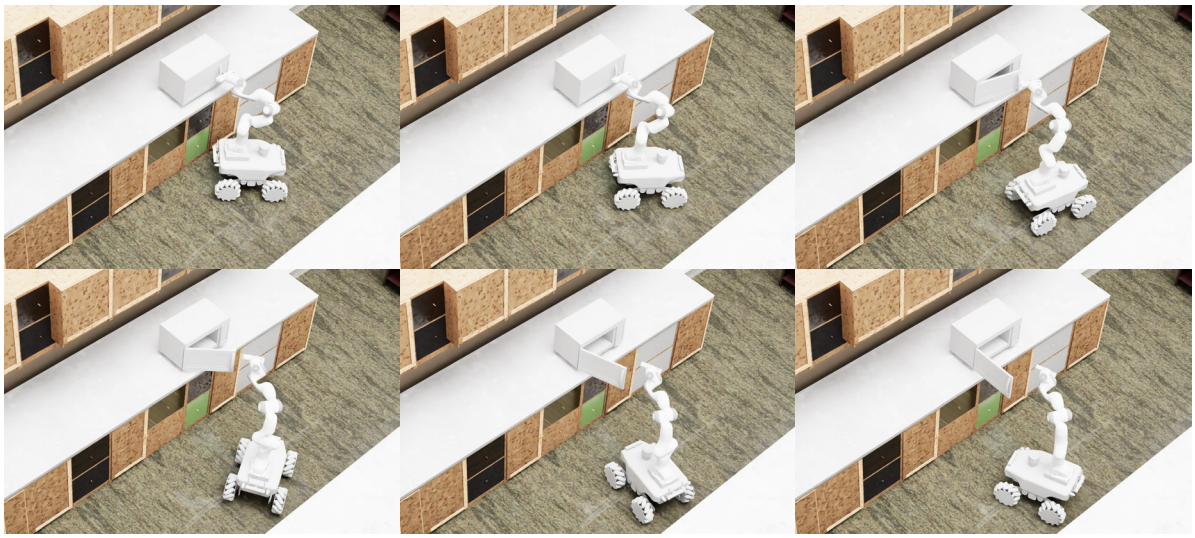
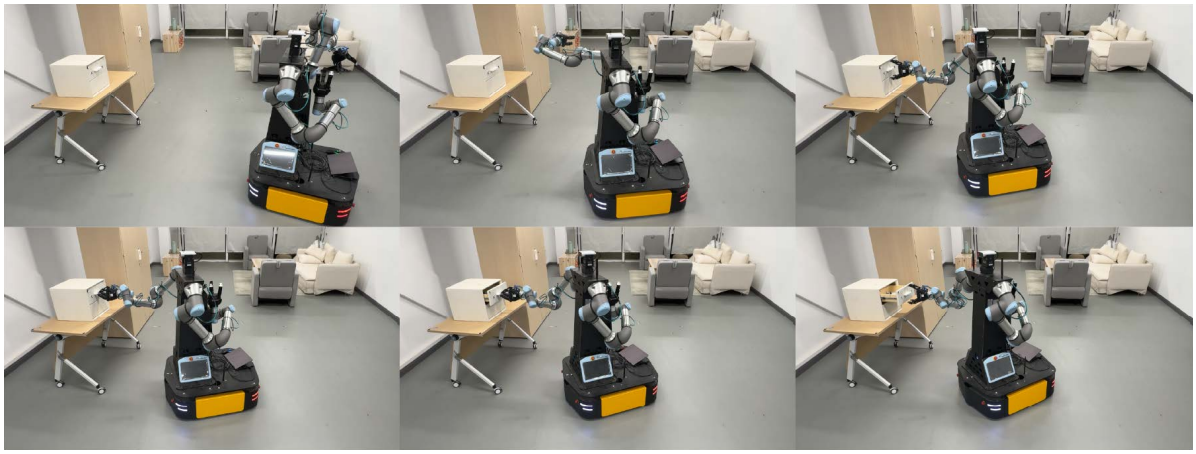
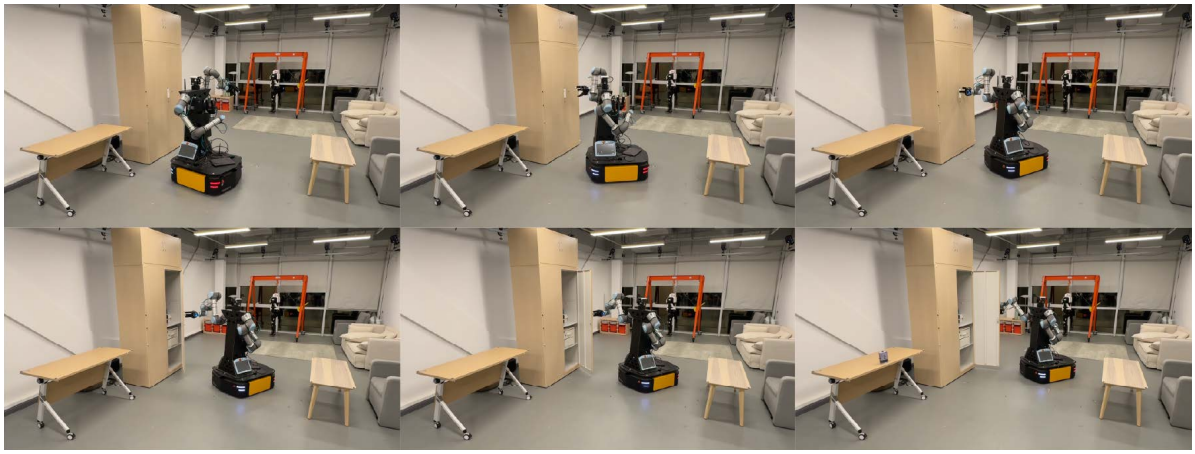


Figure A7. **Representative inference failure.** Small inconsistencies in base or arm pose predictions compound over time, eventually pushing the robot into an infeasible configuration.



(a) Drawer-opening trajectory on the UR5-Ridgeback platform.



(b) Cabinet door opening trajectory on the UR5-Ridgeback platform.

Figure A8. **Real-world validation on a UR5-Ridgeback platform.** Planned trajectories for drawer opening and cabinet door opening are executed smoothly without collision or constraint violation.

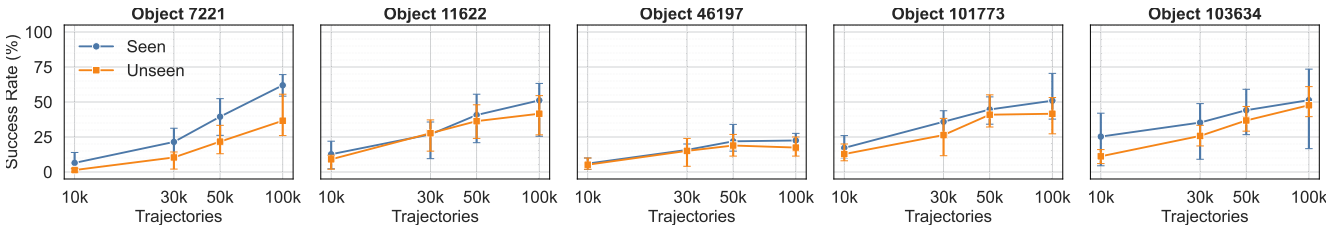


Figure A9. **Scaling analysis at 100k trajectories.** Training DP3 on a 100k-trajectory dataset spanning multiple articulated objects and scenes improves cross-object capability and generalization to unseen environments, compared with smaller-scale training subsets.

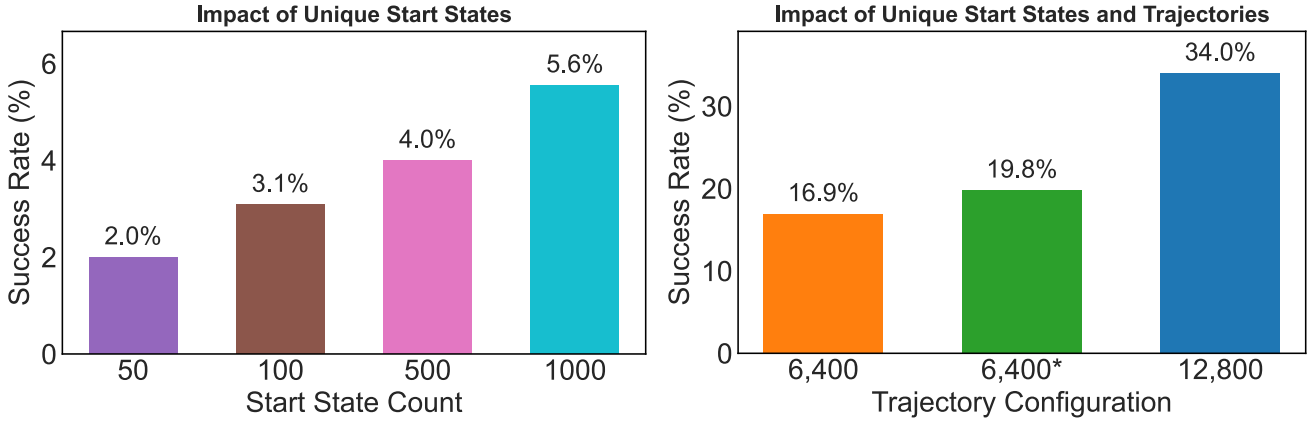
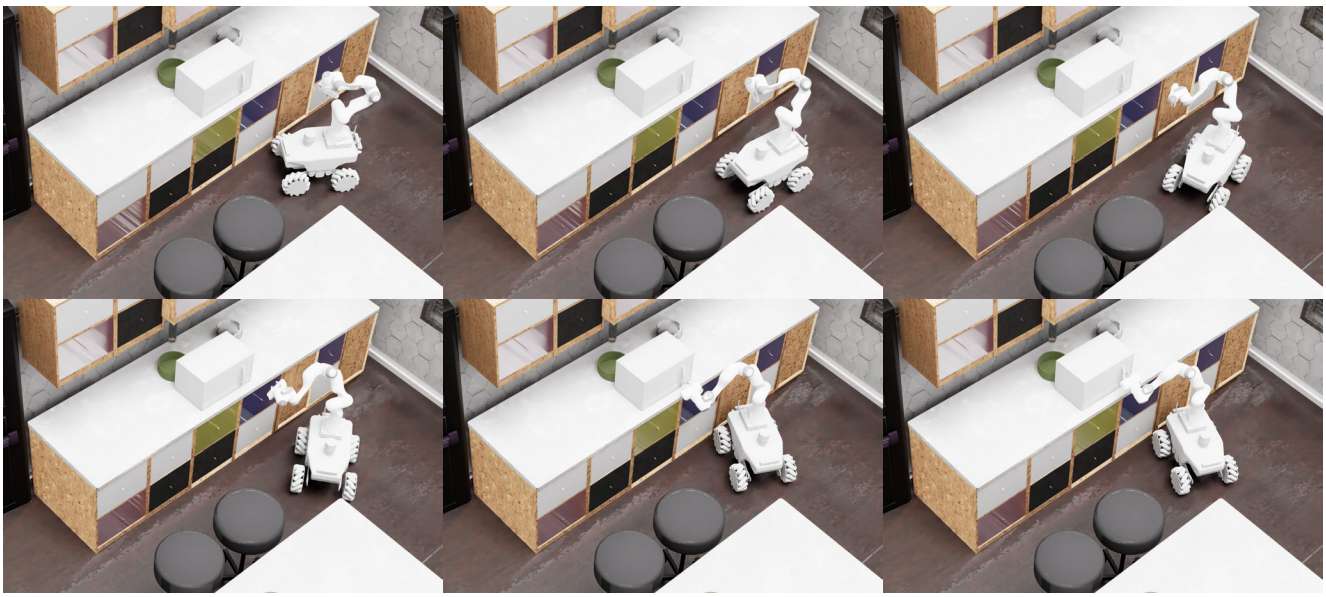
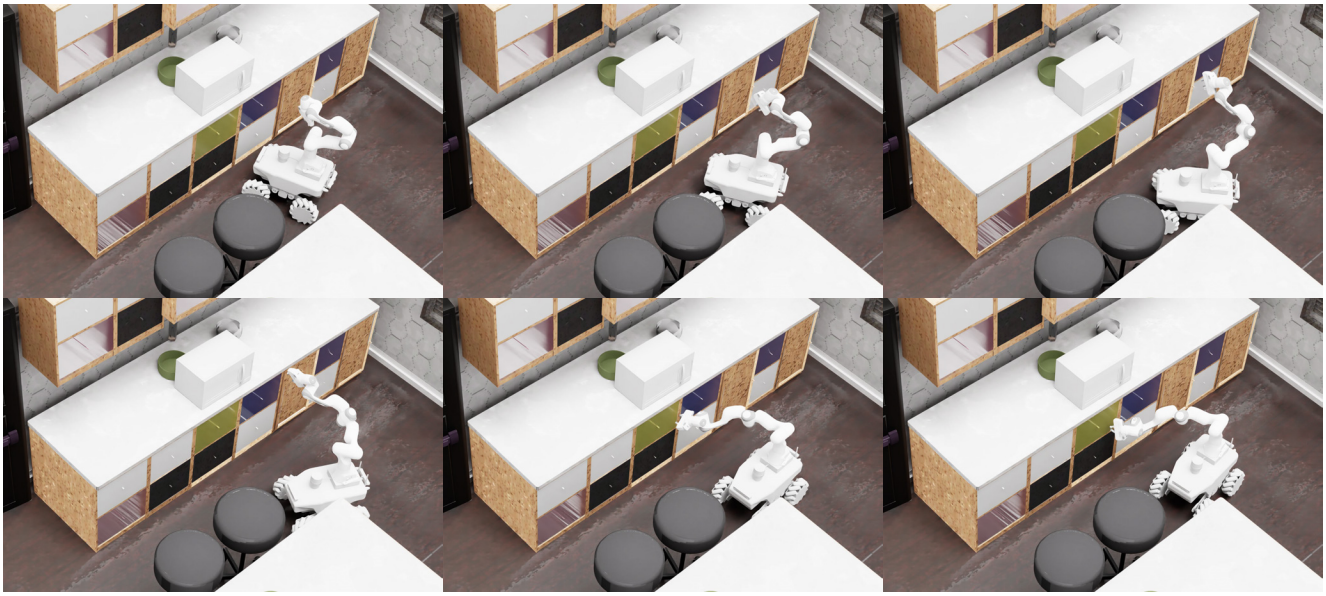


Figure A10. **Ablation on start-state diversity. Top:** Increasing unique start states (50–1,000) monotonically improves success on the 1,000-trajectory benchmark. **Bottom:** For fixed dataset sizes (6,400 trajectories), more diverse start states improve performance; the full 12,800-trajectory dataset achieves the highest success rate.



(a) Successful execution with coordinated base-arm motion.



(b) Failure case where the policy fails to stabilize the grasp.

Figure A11. **Inference examples for the pick task** using a policy trained on 1,000 `AutoMoMa` trajectories.

(a) Fixed-grasp trajectory.

(b) Grasp-switching trajectory.

Figure A12. **Fixed-grasp *vs*. grasp-switching.** Grasp switching enables larger object opening angles by avoiding link collisions.