RoboMorpher: Scaling Robot Morphology Representations for Cross-Embodiment Action Prediction

Anonymous CVPR submission

Abstract

The same joint configuration produces different link poses on different robots. We present RoboMorpher, an action tokenizer conditioned on robot morphology. A survey of 33 robot descriptions informs MorphGen, which normalizes serial-arm kinematics and generates robot descriptions with forward-kinematics supervision. We pretrain RoboMorpher on 40 million morphology–state pairs using joint-aligned residual quantization and joint-angle and link-pose reconstruction. A downstream vision–language–action (VLA) policy predicts these codes and refines the decoded trajectories. We also introduce EmbodiBench, a ten-task benchmark evaluated across 30 generated URDFs. The policy achieves 56.40% success on RoboTwin 2.0 and 23.27% on EmbodiBench, improving over continuous joint-space control by 9.73 and 15.34 percentage points. Removing link-pose supervision lowers EmbodiBench success by 17.94 percentage points. Gains concentrate on lifting, while stacking remains unsolved.

1. Introduction

Generalist robot policies learn from demonstrations collected across platforms [12–14], but their action coordinates remain tied to individual robots. The same joint values can place two arms in different poses when their link lengths or joint arrangements differ. Padding actions to a common dimension does not resolve this ambiguity.

End-effector (EE) control delegates the conversion from Cartesian targets to joint commands to a robot-specific controller. Direct joint control instead specifies the full configuration, including choices left undetermined by an EE target. Learning this interface across robots requires a representation of both the commanded configuration and the kinematics that determine its effect.

We pretrain such a representation using forward kinematics (FK), which supplies link-pose targets without manipulation demonstrations. A survey of 33 URDF/MJCF descriptions provides link-length priors for *MorphGen*. The generator expresses serial chains in a common local-Z convention, samples their geometries, and selects manipulation home poses through collision-aware inverse kinematics (IK). This procedure extends the surveyed inventory with generated descriptions and configuration–pose pairs.

RoboMorpher encodes these pairs into discrete action codes. Morphology conditions the encoder and decoder, and residual quantization retains two codes per joint. Joint-angle reconstruction preserves control coordinates; link-pose reconstruction trains the codes to retain their spatial effects. The tokenizer is then frozen and integrated into the discrete-to-continuous policy architecture of Discrete Forcing [20].

We evaluate on 15 RoboTwin 2.0 tasks across five embodiments [3] and introduce *EmbodiBench*, a ten-task suite across 30 generated URDFs. RoboMorpher improves mean success over continuous joint-space control by 9.73 and 15.34 percentage points, respectively. The full task results and reconstruction metrics identify where the representation helps and where it fails.

Our contributions are:

- A robot-description survey and **MorphGen**, a procedural generator using a common kinematic interface and empirical link priors.
- **RoboMorpher**, a morphology-conditioned action tokenizer pretrained through joint-angle and link-pose reconstruction.
- Integration of the frozen tokenizer into a VLA policy for discrete prediction and continuous refinement.
- **EmbodiBench** and full multi-embodiment evaluations of task performance, representation variants, and reconstruction.

2. Related Work

Learning across embodiments. Open X-Embodiment pools multi-robot demonstrations [14]; Octo adapts to new observation and action spaces [13]; and RDT uses a physically interpretable unified action space [12]. Shared Modular Policies exchange messages between actuator modules [9], while MetaMorph conditions a Transformer on morphology [8]. We pretrain discrete joint-action codes from robot descriptions and kinematics before task-policy learning.

Table 1. **Robot-description survey.** Entry-level statistics; masses and reaches are reported values. The appendix documents description-level discrepancies.

Property	Count / range
Description entries / distinct names	33 / 30
URDF / MJCF entries	24 / 9
Gripper / other tool / no tool listed	11 / 1 / 21
Approx. reach (32 entries)	0.25–2.05 m
Model mass (32 entries)	0.226–33.5 kg

Spatial representations. G3Flow constructs object-centric 3D semantic flow for pose-aware manipulation [4]. Spark-0 aligns visual and tactile pretraining for dexterous manipulation [6]. RoboMorpher uses spatial supervision at the action interface: joint configurations are reconstructed together with the link poses they produce.

Action tokenization and generation. Vector quantization learns discrete latents through reconstruction [19], while FAST compresses action sequences in the frequency domain [16]. Our tokenizer encodes individual configurations conditioned on morphology; temporal fitting is a separate policy operation. For continuous generation, flow matching learns a vector field between source and data distributions [11]. Discrete Forcing uses discrete predictions to guide this refinement [20]. We adopt its two-branch design with learned action codes and MaskGIT-style completion [2]. StarVLA provides the continuous-control baselines [17].

Data generation and benchmarks. RoboGen generates tasks, scenes, and supervision through simulation [21]. RoboTwin 2.0 combines automated demonstrations with domain-randomized bimanual evaluation [3]. RMBench studies memory-dependent manipulation [7], and RoboDojo evaluates generalist policies in simulation and reality [5]. MorphGen generates kinematic pretraining data; EmbodiBench evaluates downstream control on a task–URDF grid, complementing our five-embodiment RoboTwin 2.0 evaluation.

3. Robot Morphology Survey

We survey 33 serial-manipulator descriptions covering 30 names. Gen3, Koch v1.1, and Panda each have URDF and MJCF entries. Statistics count descriptions, with arm degrees of freedom (DoF) taken from the reported arm-DoF field. Table 1 summarizes the inventory, and Fig. 1 plots Gaussian priors fitted to the link-length records. The complete records and description ambiguities appear in Appendix A.

Joint structure. Six- and seven-DoF arms account for 30/33 entries (90.9%), motivating MorphGen's initial sampling range. Joint limits are bounded in 26 entries, mixed in four, and unrestricted in three. Tool and coupling conventions vary across descriptions: the Panda records, for example, differ in whether mimic coupling reduces gripper DoF. We therefore extract an explicit arm chain and handle gripper actuation separately.

Geometric priors. A separate set of 30 link-length records contains 185 reported values across seven positions (Fig. 1(a)). The first five positions have 30 values each, the sixth has 27, and the seventh has eight. We retain zeros and exclude missing cells. Positions 3 and 4 have the largest means, 0.3029 and 0.2270 m. Position 5 has a lower mean (0.1532 m) but the largest sample standard deviation (0.1862 m), with one reported value of 1.016 m. A shared length distribution would discard these positional differences. MorphGen therefore uses seven position-specific priors, then rescales the sampled chain to vary overall size (Sec. 4.2). These priors describe the collected inventory rather than the population of robot hardware; the seventh is supported by fewer records.

4. MorphGen: Scaling Robot Morphology Data

MorphGen generates serial-arm descriptions and configuration–pose pairs for tokenizer pretraining. It normalizes the kinematic interface, samples chain geometries, and selects feasible home poses. Figure 1 shows the survey-derived priors and generated examples.

4.1. A Unified Kinematic Representation

Different local frames and intermediate fixed links can encode the same arm kinematics. We express an ordered serial chain in a common local-Z convention. Let $O_i \in \mathrm{SE}(3)$ be joint i's origin transform after composing intervening fixed transforms, and a_i its rotation axis. We choose C_i such that $C_i e_z = a_i$, using the minimum-angle rotation and a fixed X-axis choice for the antiparallel case. With $C_0 = I$,

$$\widetilde{O}_i = C_{i-1}^{-1} O_i C_i, \qquad \widetilde{T}_i(q) = \prod_{k=1}^{i} \widetilde{O}_k R_z(q_k). \quad (1)$$

Rotations are embedded in homogeneous transforms, R_z is a local-Z rotation, and T_i is the original root-to-link transform. The identity $\widetilde{T}_i(q) = T_i(q)C_i$ preserves joint values, signs, zeros, and physical joint positions. Inverse frame corrections recover link orientations; asset-origin corrections preserve visual and collision placement.

The tokenizer receives $\mathcal{M} = \{(t_i, \phi_i)\}_{i=1}^{J}$: each normalized origin's translation and XYZ roll–pitch–yaw angles,

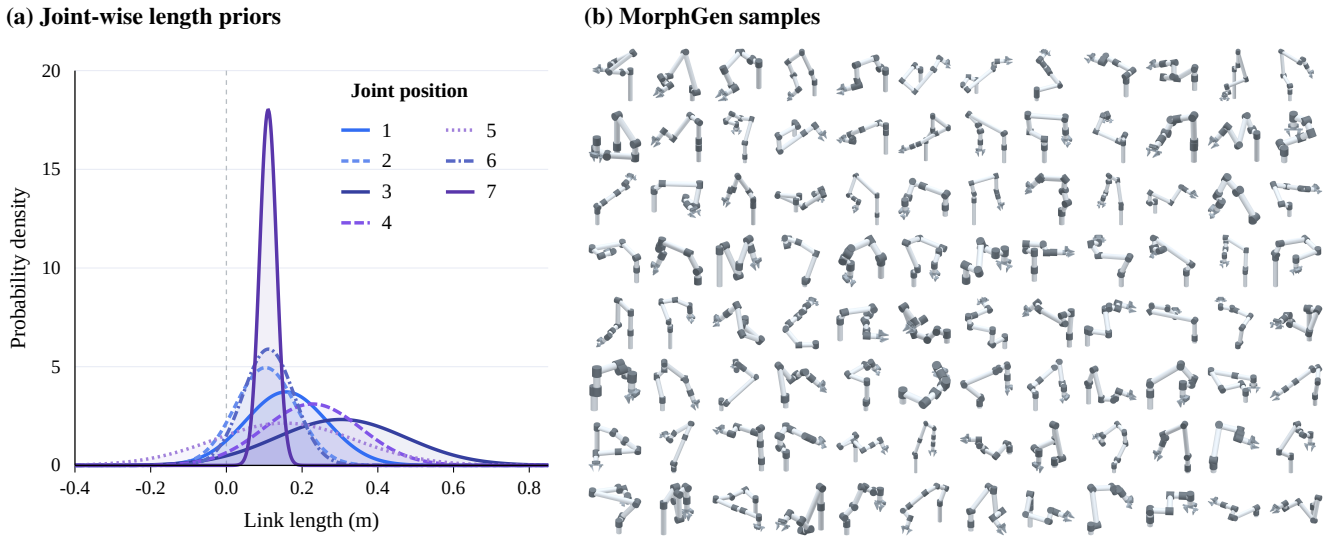


Figure 1. **Survey and morphology generation.** (a) Seven Gaussian priors $\mathcal{N}(\mu_i, \sigma_i^2)$ fitted to 185 reported lengths from 30 description records. Each curve uses the mean and sample standard deviation at one joint position, with zeros retained and missing entries excluded. Curves show the priors before truncation and chain-length rescaling. (b) Ninety-six generated morphologies varying link geometry and joint-frame arrangement. Appendices A and B provide the survey records and generation details.

plus a validity mask. Action angles are supplied separately. This six-value interface removes explicit axis parameters and intermediate fixed paths. It supports independent revolute/continuous chains; selected prismatic or mimic joints and skipped movable joints are rejected. The convention fixes the input frames, without defining a unique canonical form for physical morphology.

4.2. Survey-Guided Procedural Generation

The generator samples six- or seven-joint revolute arms in the local-Z convention. An initial length $\widetilde{\ell}_i$ is drawn from a truncated Gaussian using the mean μ_i and sample standard deviation σ_i of survey column i (Tab. 8). The lower bound is zero and the upper bound is $\max(u, \mu_i + 0.1)$ m, where u is 0.65 m with probability 0.95 and 0.35 m otherwise. The lengths are rescaled to a sampled total chain length:

$$\ell_i = L \frac{\widetilde{\ell}_i}{\sum_k \widetilde{\ell}_k}, \qquad L \sim \begin{cases} \mathcal{U}(0.8, 1.6) & 0.95, \\ \mathcal{U}(1.6, 2.0) & 0.03, \\ \mathcal{U}(0.3, 0.8) & 0.02. \end{cases} \tag{2}$$

Lengths are in meters. L is the sum of link lengths, not a workspace radius. The sampler also varies displacement directions and joint-frame rotations, assigns local-Z axes and $[-2\pi, 2\pi]$ limits, and exports gripper-equipped assets with planner and simulator configurations. Appendix B specifies the sampling distributions and export settings.

4.3. Home Poses and Pretraining Samples

A cuRobo solver [18] searches perturbed tabletop targets with 32 IK seeds and self-collision checking. A successful configuration is absorbed into the joint origins to define the home pose; descriptions with no solution are discarded. This step changes the controller zero, unlike the frame normalization above, and does not translate the joint-limit intervals.

Each training sample pairs a description and configuration with forward-kinematics targets from the original URDF tree, corrected into the normalized frames. Inputs contain $J \times 7$ values (origin translation/rotation and joint angle); targets contain $J \times 10$ values (position, 6D rotation, and angle), padded to seven joints. Pretraining uses 40 million morphology–state pairs.

5. RoboMorpher: Morphology-Aware Action Tokenization

RoboMorpher maps a description $\mathcal{M}$ and configuration $q \in \mathbb{R}^J$ to two code indices per valid joint (Fig. 2). Its decoder reconstructs joint values and link poses conditioned on the same description. A mask m excludes padding. Tokenizer training uses kinematic data only.

5.1. Morphology-Conditioned Joint Encoding

Separate two-layer Transformer branches encode the origin parameters and joint angles, retaining one feature per joint. Morphology features $c_{\mathcal{M},i}$ modulate a six-layer action encoder and four-layer decoder through joint-wise AdaLN-

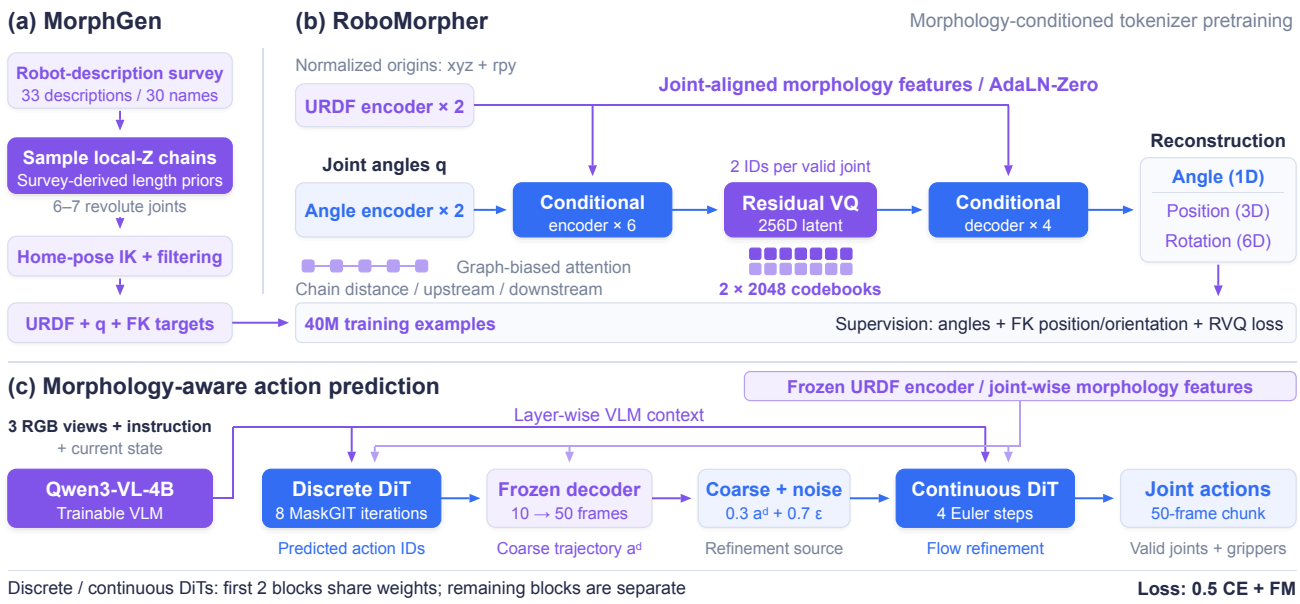


Figure 2. **RoboMorpher pipeline.** (a) MorphGen samples local-Z chains using survey priors and IK filtering. (b) URDF-conditioned tokenization learns two codes per joint with FK supervision. (c) The VLA predicts codes, decodes ten nodes, interpolates 50 frames, and refines them with flow matching. Frozen URDF features condition both DiTs and the decoder; VLM context conditions the DiTs.

Zero [15]. All blocks have width 1024, eight heads, and bidirectional attention.

For head h, query joint i, and key joint j, attention includes serial-chain biases:

$$A^h_{ij} = \frac{Q^{h\top}_i K^h_j}{\sqrt{d}} + b^h_s(|i-j|+1) + b^h_p((i-j)_+) + b^h_c((j-i)_+) + M_{ij}. \tag{3}$$

Here d is the head width and $(x)_+ = \max(x, 0)$. The learned tables encode hop distance and directed upstream/downstream relations. They start at zero; index zero in the directional tables contributes no bias. M masks invalid keys, and padded features are cleared. AdaLN supplies the description-dependent geometry.

5.2. Joint-Aligned Residual Quantization

A linear map projects each conditioned feature to $u_i \in \mathbb{R}^{256}$. Residual vector quantization (RVQ) [10] uses two 2048-entry codebooks shared across joints and descriptions:

$$z^{(1)}_i = \arg\min_k \|u_i - e^{(1)}_k\|^2_2, \tag{4}$$

$$z^{(2)}_i = \arg\min_k \|u_i - e^{(1)}_{z^{(1)}_i} - e^{(2)}_k\|^2_2, \tag{5}$$

$$\widehat{u}_i = e^{(1)}_{z^{(1)}_i} + e^{(2)}_{z^{(2)}_i}. \tag{6}$$

Each joint contributes two indices per frame. The quantized feature is projected back to width 1024 for conditional decoding. Codebooks use exponential moving averages and a straight-through gradient estimator (Appendix C). Decoding depends on $\mathcal{M}$, so a shared code need not correspond to the same joint value on every robot.

5.3. Joint and Geometric Reconstruction

The decoder predicts normalized joint angles and positions, together with each valid joint's child-link 6D rotation [22]. For $N = \sum_i m_i$, the objective is

$$\mathcal{L}_q = \frac{1}{N}\sum_i m_i |\widehat{\bar{q}}_i - \bar{q}_i|, \tag{7}$$

$$\mathcal{L}_p = \frac{1}{3N}\sum_i m_i \|\widehat{\bar{p}}_i - \bar{p}_i\|_1, \tag{8}$$

$$\mathcal{L}_R = \frac{1}{\pi N}\sum_i m_i \, d_{\mathrm{SO}(3)}(\widehat{R}_i, R_i), \tag{9}$$

$$\mathcal{L}_{\mathrm{tok}} = \mathcal{L}_q + \mathcal{L}_p + 0.5\mathcal{L}_R + \mathcal{L}_{\mathrm{RVQ}}. \tag{10}$$

Bars denote normalized values; $d_{\mathrm{SO}(3)}$ is the rotation geodesic in radians. $\mathcal{L}_{\mathrm{RVQ}}$ includes the commitment weight. Rotations are denormalized before Gram–Schmidt conversion. Per-feature min–max statistics are shared across valid joints and robots; raw joint angles are not wrapped.

Joint reconstruction retains control values, while link-pose targets supervise their kinematic consequences. The auxiliary pose head does not enforce exact FK consistency of decoded angles. All tokenizer parameters, including the morphology encoder, are frozen during policy learning.

6. Integrating RoboMorpher into a VLA Policy

The policy predicts 50-frame chunks of absolute joint positions from images I, instruction ℓ, current state s, and morphology $\mathcal{M}$. It combines Qwen3-VL-4B-Instruct [1], the frozen tokenizer, and a two-branch diffusion Transformer (DiT). Each arm has seven padded joint slots and one gripper channel, giving 16 coordinates; masks exclude absent joints and arms.

6.1. From Frame Tokens to Action Chunks

We fit $K = 10$ nodes to each arm's 50-frame trajectory $A \in \mathbb{R}^{50\times J}$. For an endpoint-aligned linear interpolation matrix $U \in \mathbb{R}^{50\times 10}$,

$$B^* = (U^\top U + 10^{-4} I)^{-1} U^\top A. \tag{11}$$

Each fitted node is tokenized independently. Decoded nodes are interpolated to 50 frames at inference. Grippers use linear resampling and 256-bin quantization. An arm with J joints contributes $K(2J+1)$ indices, giving at most 300 for two seven-joint arms. The 256-bin baseline uses the same fitting; FAST [16] uses variable-length autoregressive codes without it.

6.2. Morphology-Conditioned Policy Architecture

LayerNorm and a linear map project each VLM hidden-state level into width-1024, 16-head action blocks. The L blocks align with the VLM's L levels, alternating VLM cross-attention and action self-attention. Following the partially shared design of Discrete Forcing [20], our branches share two initial blocks and use separate remaining blocks and output heads. They share parameters but do not attend to each other. Current state and 32 learned future queries supply additional context. Frozen URDF features modulate the blocks through AdaLN; 256-bin state tokens also enter the VLM instruction.

6.3. Discrete Prediction and Refinement

The discrete branch minimizes cross-entropy on masked valid positions, using a uniformly sampled time and a cosine masking ratio. At inference, eight MaskGIT iterations [2] commit codes by confidence. Frozen decoding and interpolation yield the coarse trajectory a_d.

The continuous branch refines a_d in normalized action space. For target trajectory a^* and context $c = (I, \ell, s, \mathcal{M})$,

$$x_0 = 0.3 a_d + 0.7\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{12}$$

$$x_\gamma = (1-\gamma) x_0 + \gamma a^*, \tag{13}$$

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\|v_\phi(x_\gamma, \gamma, c) - (a^* - x_0)\|_m^2\right]. \tag{14}$$

The squared error averages valid coordinates per example. Both trajectories use per-slot 1st/99th-percentile normalization and clipping to $[-2.2, 2.2]$. Training constructs a_d from ground-truth codes; inference uses predicted codes.

The policy minimizes $0.5\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{FM}}$. Four Euler steps of size $1/4$ advance γ from zero to one, producing the 50-frame action chunk. Appendix D gives optimization and training-time sampling settings.

7. EmbodiBench: Evaluation Protocol

EmbodiBench evaluates manipulation on generated kinematic chains in SAPIEN. Its ten tasks comprise three lifting, three picking, three stacking, and one pushing variant, with single arms or same-/opposite-side dual arms (Fig. 3).

Data and coverage. Each task uses 30 URDFs and 50 demonstrations per task–URDF pair, totaling 15,000 demonstrations. Dual-arm trials use the same morphology for both arms. Policies receive head and wrist RGB views resized to 224×224; absent arm channels are masked.

Evaluation. Separate scene seeds provide five trials per task–URDF pair: 150 per task and 1,500 per method. Planner-infeasible scenes do not consume the quota. Each accepted trial has a fixed instruction and permits 300 executed action commands, counted per frame within a chunk. We report task success rates and their unweighted mean.

Success criteria. Picking requires an object rise above 7 cm. Pushing requires planar placement within 3 cm per coordinate and orientation error below 9°. Stacking requires alignment within $(2.5, 2.5, 1.2)$ cm of the target pose and open grippers. Lifting combines height, grasp/contact, and configurable stability checks (Appendix E).

8. Experiments

We evaluate task success, then use representation variants and reconstruction errors to examine the source and limits of the gains.

8.1. Experimental Setup

Training and evaluation. The tokenizer trains on 40 million morphology–state pairs for 300 epochs, with batch 4096 per process. Policy training freezes the tokenizer and updates the VLM and action head with batch 16 per process. RoboTwin 2.0 uses 3,750 demonstrations (15 tasks, 5 embodiments, 50 demonstrations each) for 30k steps; EmbodiBench uses 15,000 for 60k steps. Full settings appear in Appendix C. RoboTwin 2.0 evaluates 20 trials per task–embodiment pair; EmbodiBench follows Sec. 7. Each benchmark has 1,500 trials per method. Equal task quotas make the macro-average equal to pooled success, up to rounding.

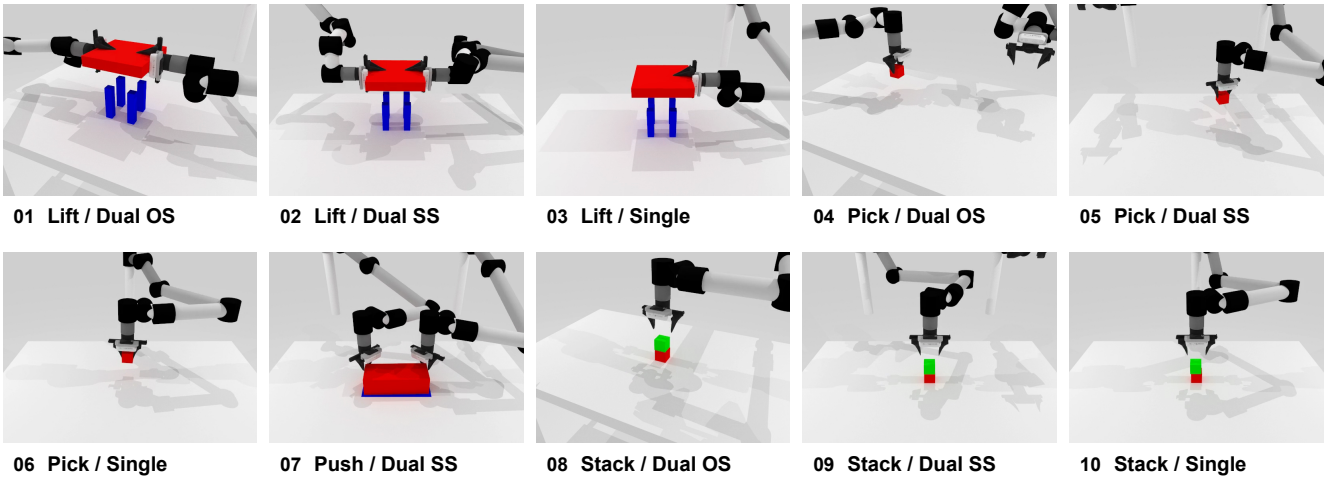


Figure 3. **The ten EmbodiBench tasks.** Successful saved-trajectory keyframes, ordered as in Tab. 3. SS/OS denote same-side/opposite-side dual-arm placement. These replays illustrate tasks; they are separate from learned-policy evaluation.

Comparisons. *StarVLA (joint)* predicts continuous joint actions. *256-bin* uses uniform bins, MaskGIT, and tennode fitting; *FAST* [16] uses autoregressive codes without fitting. *Angle+M* removes link-pose reconstruction. *Angle* also removes morphology conditioning and graph biases and changes commitment weight from 0.1 to 0.25. *StarVLA (EE)* is a separate control-interface reference. Ours is the VLA policy using RoboMorpher.

8.2. Multi-Embodiment Task Performance

RoboMorpher achieves the highest mean success among the evaluated joint-space methods: 56.40% on RoboTwin 2.0 and 23.27% on EmbodiBench (Tabs. 2 and 3), exceeding StarVLA (joint) by 9.73 and 15.34 percentage points (pp).

RoboTwin 2.0: gains across tasks. RoboMorpher improves over continuous joint control on 12 of 15 tasks. The largest gains are on bowl stacking (54% vs. 25%), microphone handover (75% vs. 47%), and hammering (34% vs. 10%). These three tasks contribute 55.5% of the net gain. The median improvement is 6 pp, below the 9.73 pp mean; the three regressions are limited to 3–4 pp on bell clicking, dual-bottle picking, and bottle disposal. Against 256-bin and FAST, the mean gains are 4.80 and 5.67 pp, with improvements on nine and eleven tasks, respectively.

EmbodiBench: lifting drives the improvement. The three lifting tasks average 55.11% success, compared with 14.00% for StarVLA (joint), accounting for about 80% of the overall gain. Picking improves from 7.11% to 12.89% on average, and pushing from 16.00% to 28.67%. The policy improves on six of seven non-stacking tasks, but single-arm picking falls from 9.33% to 4.00%. All three stacking scores remain zero; every evaluated method achieves at most 1.33% on these tasks. RoboTwin 2.0 bowl stacking reaches 54%, but this success does not carry over to the EmbodiBench block-stacking tasks. The objects and success tests differ, so this contrast alone does not identify a morphology effect.

Opposite-side placement also yields higher success than same-side placement: 61.33% versus 52.67% for lifting and 23.33% versus 11.33% for picking. The corresponding gains over continuous joint control are 46.66 versus 38.67 pp for lifting and 17.33 versus 5.33 pp for picking. Thus, the gains vary within a task family as well as across primitives. These comparisons change the arm placement and scene geometry, so they do not isolate a single source of difficulty.

Lifting success does not extend to single-arm picking, although both require raising an object. Their success predicates differ: lifting includes height and grasp/contact checks, while picking measures rise from the initial position (Sec. 7). Task names alone therefore do not explain the discrepancy. Distinguishing perception, grasp, and control failures requires rollout-level annotations.

Tokenization baselines. The baseline ranking changes across benchmarks. On RoboTwin 2.0, 256-bin and FAST exceed continuous joint control by 4.93 and 4.06 pp; on EmbodiBench, they trail it by 2.40 and 2.00 pp. Their mean lifting rates are 9.78% and 10.44%, compared with 55.11% for RoboMorpher. The lifting improvement is thus specific to RoboMorpher among the evaluated discrete-action policies. These comparisons also change decoding and temporal processing; the link-pose ablation below provides a closer test of the representation objective.

Table 2. **Full RoboTwin 2.0 results: success rate (%).** All 15 evaluated tasks across five embodiments; 20 trials per pair. Bold/underline mark best/second-best positive joint-space scores (ties included). EE uses a separate control interface.

	Joint-space baselines			Representation variants		Full model	Reference
Task	Joint	256-bin	FAST	Angle	Angle+M	Ours	EE
grab_roller	90.00%	**97.00%**	95.00%	95.00%	**97.00%**	92.00%	100.00%
shake_bottle	92.00%	95.00%	**96.00%**	88.00%	**96.00%**	94.00%	100.00%
handover_mic	47.00%	67.00%	74.00%	52.00%	74.00%	**75.00%**	93.00%
place_burger_fries	59.00%	65.00%	67.00%	58.00%	53.00%	**74.00%**	52.00%
place_empty_cup	69.00%	**70.00%**	68.00%	69.00%	69.00%	**70.00%**	85.00%
click_bell	88.00%	72.00%	69.00%	84.00%	**100.00%**	84.00%	88.00%
lift_pot	28.00%	**37.00%**	17.00%	5.00%	22.00%	30.00%	82.00%
open_laptop	66.00%	73.00%	76.00%	64.00%	70.00%	**82.00%**	97.00%
beat_block_hammer	10.00%	13.00%	25.00%	11.00%	18.00%	**34.00%**	32.00%
stack_bowls_three	25.00%	59.00%	54.00%	54.00%	**64.00%**	54.00%	68.00%
pick_dual_bottles	32.00%	**35.00%**	32.00%	25.00%	30.00%	29.00%	65.00%
move_can_pot	20.00%	21.00%	17.00%	20.00%	12.00%	**26.00%**	52.00%
place_phone_stand	31.00%	34.00%	42.00%	27.00%	39.00%	**50.00%**	51.00%
put_bottles_dustbin	**21.00%**	10.00%	12.00%	8.00%	18.00%	17.00%	42.00%
place_dual_shoes	22.00%	26.00%	17.00%	20.00%	16.00%	**35.00%**	42.00%
Mean	46.67%	51.60%	50.73%	45.33%	51.87%	**56.40%**	69.93%

Table 3. **Full EmbodiBench results: success rate (%).** All 10 tasks across 30 URDFs; five trials per pair. Bold/underline mark best/second-best positive joint-space scores (ties included). EE uses a separate control interface.

	Joint-space baselines			Representation variants		Full model	Reference
Task	Joint	256-bin	FAST	Angle	Angle+M	Ours	EE
lift_dual_os	14.67%	14.00%	9.33%	10.67%	10.67%	**61.33%**	62.67%
lift_dual_ss	14.00%	4.00%	11.33%	6.67%	8.67%	**52.67%**	58.67%
lift_single	13.33%	11.33%	10.67%	6.00%	9.33%	**51.33%**	78.67%
pick_dual_os	6.00%	2.67%	4.00%	1.33%	3.33%	**23.33%**	14.67%
pick_dual_ss	6.00%	4.00%	2.67%	2.67%	4.67%	**11.33%**	8.00%
pick_single	**9.33%**	4.00%	5.33%	5.33%	1.33%	4.00%	32.67%
push_dual_ss	16.00%	15.33%	15.33%	13.33%	14.67%	**28.67%**	62.67%
stack_dual_os	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.33%
stack_dual_ss	0.00%	0.00%	0.00%	0.00%	**0.67%**	0.00%	0.00%
stack_single	0.00%	0.00%	**0.67%**	0.00%	0.00%	0.00%	0.67%
Mean	7.93%	5.53%	5.93%	4.60%	5.33%	**23.27%**	32.00%

Comparison with EE control. The EE reference remains stronger overall at 69.93% and 32.00%, gaps of 13.53 and 8.73 pp. On EmbodiBench, RoboMorpher exceeds it on both dual-arm picking tasks but trails by 28.67 pp on single-arm picking and 34.00 pp on pushing. The near-matched opposite-side lifting result (61.33% vs. 62.67%) also contrasts with the single-arm gap (51.33% vs. 78.67%). These differences depend on the task and execution interface; they do not isolate tokenizer quality.

8.3. Effect of Representation Design

Link-pose supervision. Removing link-pose reconstruction reduces mean success by 4.53 pp on RoboTwin 2.0 and 17.94 pp on EmbodiBench. On EmbodiBench, lifting drops from 55.11% to 9.56% on average, and all seven non-stacking tasks decline. Angle+M retains morphology input yet reaches only 5.33% overall, below continuous joint control at 7.93%. Link-pose targets are particularly useful in

Table 4. **Tokenizer reconstruction.** Errors across six description groups; P90 is the 90th percentile. Bold/underline mark the lowest/second-lowest value per column (ties included). Metric definitions appear in Appendix C.

	Joint angle (°)			Position (m)			Orientation (°)		
Description	Mean	P90	Max	Mean	P90	Max	Mean	P90	Max
ALOHA	<u>0.4124</u>	0.8951	<u>14.4648</u>	<u>0.004255</u>	<u>0.009446</u>	0.116570	<u>1.3237</u>	<u>3.4343</u>	88.6863
ARX-X5	**0.4107**	**0.7768**	52.6336	0.004899	0.013960	<u>0.112769</u>	**1.0860**	**2.0659**	133.9178
Franka Panda	0.4563	1.0207	42.7852	0.009204	0.022782	0.644741	1.8655	4.4817	178.8929
Piper	0.4448	<u>0.8695</u>	40.3280	**0.003503**	**0.007992**	0.186133	1.6299	4.5327	<u>88.5053</u>
UR5-WSG	0.5429	1.2653	**5.6463**	0.007017	0.016207	**0.048518**	1.3591	3.7680	**11.4713**
Generated URDFs	0.4764	1.0696	80.9388	0.008355	0.018587	0.752892	2.1928	4.6185	179.9428

the generated-chain evaluation, where they supervise how joint values affect spatial configurations.

On RoboTwin 2.0, nine tasks improve. Burger-and-fries and shoe placement gain 21 and 19 pp; bell clicking and bowl stacking lose 16 and 10 pp.

Conditioning and generation choices. Angle+M exceeds Angle by 6.54 pp on RoboTwin 2.0 but only 0.73 pp on EmbodiBench. Supplying morphology without link-pose targets thus yields a much smaller gain on the generated chains. This comparison jointly changes conditioning, graph biases, and commitment weight. A separate refinement ablation is needed to attribute gains to the continuous branch.

8.4. Reconstruction Fidelity

Mean joint errors lie between $0.4107°$ and $0.5429°$ across the six description groups (Tab. 4). Generated URDFs fall within this range at $0.4764°$, with joint P90 $1.0696°$. Their mean position error is 8.355 mm, below Franka Panda's 9.204 mm, whereas their mean orientation error is the largest at $2.1928°$. Joint-angle error alone misses this difference in orientation reconstruction.

Across all six groups, joint P90 errors range from $0.7768°$ to $1.2653°$. For generated URDFs, position and orientation P90 errors are 18.587 mm and $4.6185°$. The maxima for generated URDFs reach $80.9388°$ joint error, 0.752892 m position error, and $179.9428°$ orientation error. Conversely, UR5-WSG has the largest mean joint error but the smallest maximum, $5.6463°$.

Evaluation-set sizes are unavailable, so maxima cannot establish relative tail robustness. The maxima identify severe failures without showing their frequency (Fig. 4). These tests reconstruct supplied configurations; rollouts additionally require the policy to predict suitable codes from observations. Low reconstruction error measures information retained by the tokenizer, not the policy's ability to select an action. Whether flow refinement corrects the outliers requires trajectory-level measurements.

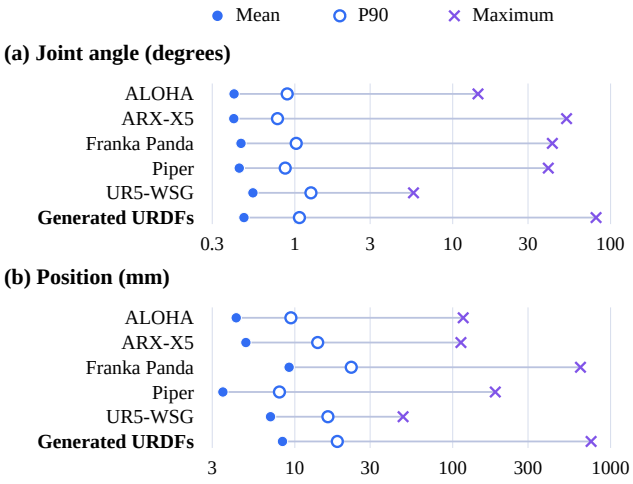


Figure 4. **Reconstruction error profiles.** Mean, P90, and maximum joint-angle and position errors from Tab. 4. Both axes are logarithmic.

8.5. Limitations

The results do not establish held-out-morphology transfer, empirical scaling trends, or variability across training seeds. Stacking and reconstruction outliers remain unresolved. Continuous refinement also uses reconstructed ground-truth codes during training and predicted codes at inference. The survey relies on a small, heterogeneous description inventory; reproducing the evaluation requires the run-specific lifting predicate and manifests (Appendix E.3).

9. Conclusion

RoboMorpher learns joint-action codes from robot descriptions and kinematic supervision supplied by MorphGen. The frozen tokenizer improves joint-space policy success on RoboTwin 2.0 and EmbodiBench. Link-pose reconstruction is most beneficial on lifting in EmbodiBench; reliable stacking and control on held-out morphologies remain open.

References

[1] Shuai Bai et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. 5

[2] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. *arXiv preprint arXiv:2202.04200*, 2022. 2, 5

[3] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. 1, 2

[4] Tianxing Chen, Yao Mu, Zhixuan Liang, Zanxin Chen, Shijia Peng, Qiangyu Chen, Mingkun Xu, Ruizhen Hu, Hongyuan Zhang, Xuelong Li, and Ping Luo. G3Flow: Generative 3D semantic flow for pose-aware and generalizable object manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1735–1744, 2025. 2

[5] Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026. 2

[6] Tianxing Chen, Wenwei Lin, Chuqiao Lyu, Peicheng Xiang, Junyuan Tang, Tian Nian, Zijian Cai, Kehe Ye, Chenze Yu, Wenxuan Zhu, et al. Spark-0: Scaling spatially aligned tactile pretraining for robot dexterous manipulation. *arXiv preprint arXiv:2610.22342*, 2026. 2

[7] Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026. 2

[8] Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. MetaMorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2022. 1

[9] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464, 2020. 1

[10] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022. 4

[11] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. 2

[12] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, 2025. 1

[13] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. 1

[14] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023. 1

[15] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 4

[16] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 2, 5, 6

[17] StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026. 2

[18] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. cuRobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv preprint arXiv:2310.17274*, 2023. 3

[19] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017. 2

[20] Jingbo Wang, Wenxuan Song, Wenhao Yu, Han Zhao, Xi Wang, Jiayi Chen, Donglin Wang, Yan Wang, and Haoang Li. Discrete forcing: Infusing discrete guidance into continuous denoising for few-step action experts. *arXiv preprint arXiv:2609.39526*, 2026. 1, 2, 5

[21] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023. 2

[22] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 4