

ArticFlow: Action-Conditioned Flow Matching for Skeleton-Free Articulated Generation

Anonymous Author(s)

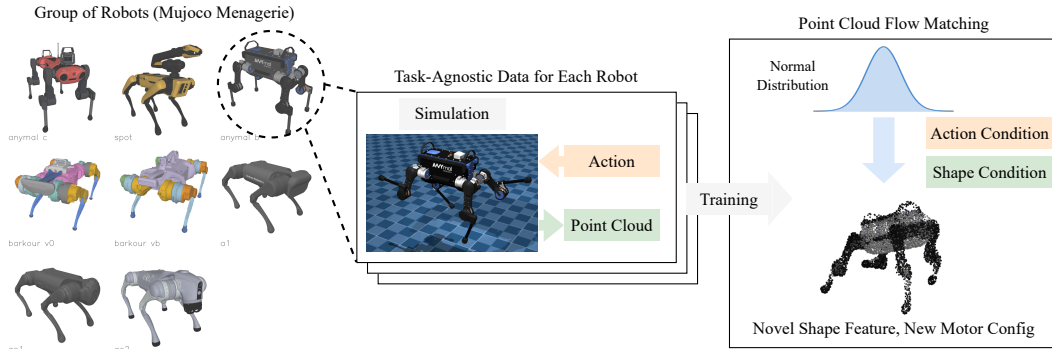
Affiliation

Address

email

Abstract: Generating articulated 3D objects, from everyday tools like pliers and eyeglasses to robotic arms and quadrupeds, requires modeling both geometric diversity and kinematic constraints, a challenge that existing point cloud generators ignore and prior articulated methods address by assuming predefined kinematic skeletons. We propose ArticFlow, a flow matching framework that learns a controllable velocity field from noise to target point sets under explicit action control. ArticFlow couples (i) a latent flow that transports noise to a shape-prior code and (ii) a point flow that transports points conditioned on the action and the shape prior. On PartNet-Mobility and MuJoCo Menagerie, a single model synthesizes novel shapes via latent interpolation and produces kinematically valid deformations under unseen actions, with consistently lower reconstruction error than per-instance baselines. We further verify that generated point clouds yield functional Unified Robot Description Format (URDF) files in a robot simulator, closing the loop from skeleton-free generation to simulation-ready assets.

Overview:



Demos:

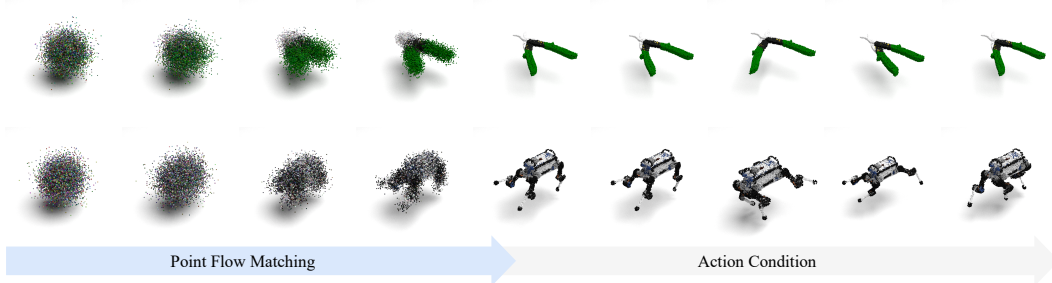


Figure 1: (Top) Robots from *MuJoCo Menagerie* [1, 2] supply task-agnostic (point cloud, joint action) training pairs; ArticFlow learns action-conditioned point cloud generation via flow matching. (Bottom) Point flow matching maps 6D Gaussian noise (position + RGB) to colored shapes (left); action conditioning produces articulated poses of the same shape (right).

1 Introduction

Generative models have advanced rapidly from VAEs [10] and GANs [11] to flow-based methods including continuous normalizing flows (CNFs) [12, 13], diffusion models [14, 15], and flow matching [16], with increasing focus on 3D content [17, 18]. Two paradigms dominate 3D generation: (i) lifting 2D priors to 3D [19, 20, 21], which struggles with geometric fidelity and requires separate reconstruction [22]; (ii) directly producing point clouds [6, 7, 9], meshes, or voxels with set- or volume-aware architectures [23, 24] on curated 3D datasets [25, 26]. Both target static geometry. Many real-world objects are articulated, coupling shape with kinematics, which motivates dedicated generative models for this setting. Most prior work focuses on analysis, such as part segmentation, articulation inference, and motion estimation [27, 28]. The challenge is to ensure 3D consistency, along with kinematic correctness and plausible deformation. Existing generative approaches either condition on a given image and reconstruct its articulation [29, 30], or follow multi-stage pipelines that first generate explicit kinematic structures (e.g., part graphs, skeletons, or bounding boxes) before synthesizing geometry for each component [31]. These methods rely on predefined structural assumptions, limiting their flexibility to represent diverse or free-form articulated geometries.

Against this backdrop, we propose **ArticFlow**, a self-contained solution that learns articulated 3D generation purely from data, without assuming a kinematic skeleton. We represent each object as a deformable point cloud and learn an action-conditioned velocity field via flow matching, capturing kinematic motion from deformation samples. A second flow over a shape-prior latent extends this to category-level generation: one model spans a category of related shapes and supports interpolation across both the action and shape-prior space, generating novel instances and motions while preserving kinematic validity. As shown in Figure 2, evaluation spans five categories: everyday articulated tools from *PartNet-Mobility* [32] and simulated robots from *MuJoCo Menagerie* [1, 2]. The two conditions visually disentangle: holding the shape prior fixed produces kinematically correct deformations of the same appearance, while holding the action fixed varies appearance under a similar pose. Beyond visual novelty, the ability to generate valid articulated morphologies speaks to a long-standing observation in robotics: behavior datasets like RT-X [33, 34, 35] reach 10^5 tasks across only 22 distinct robot embodiments.

Our contributions: (1) We introduce an action conditioned flow matching framework that learns a velocity field in point space to generate deformations under given joint commands. (2) We extend the framework with joint conditioning on action and shape prior, enabling a single model to cover a category of articulated shapes and generate unseen instances with kinematically valid deformations. (3) We evaluate on everyday objects and robots, showing (a) accurate kinematic behavior under novel actions, (b) interpolated samples that exceed dataset-level inter-instance distance (dataset-level novelty), and (c) a skeleton-free generation-to-simulation bridge in which generated point clouds yield functional URDFs actuated in PyBullet.

Method	Model	Shape Gen.	Latent Interp.	Artic. Deform.	Color Supp.
VSM [3]	DeepSDF	✗	✗	✓	✗
FFKSM [4]	NeRF	✗	✗	✓	✗
I-GAN [5]	GAN	✓	✓	✗	✗
PointFlow [6]	CNF	✓	✓	✗	✗
PVD [7]	Diffusion	✓	✗	✗	✗
Point-E [8]	Diffusion	✓	✗	✗	✓
PSF [9]	FM	✓	✗	✗	✗
ArticFlow	FM	✓	✓	✓	✓

Table 1: **Compare with previous methods.** Rows 1–2: neural simulation. Rows 3–7: point cloud generation. FM = Flow Matching.

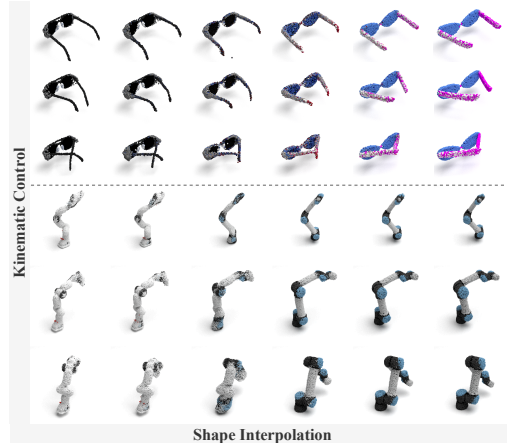


Figure 2: **ArticFlow** models a category with a single point-flow field. Columns interpolate the shape prior (novel instances); rows sweep the action (joint angles). Examples: eye-glasses (*PartNet-Mobility*) and robot arms (*MuJoCo Menagerie*).

67 2 Related Work

68 **Point cloud generation.** Generative modeling of 3D point clouds has progressed through several
 69 families: GAN and VAE adaptations such as l-GAN [5] and Set-VAE [36], continuous normalizing
 70 flows such as PointFlow [6], diffusion-based formulations including PVD [7], PSF [9], DiT-3D [37],
 71 and LION [38], and autoregressive tokenization (PointGPT [39]); conditional variants extend these
 72 with text or image inputs (Point-E [8]). Across families, architectural design has evolved from two-
 73 stage conditional generation, as in l-GAN and PointFlow, toward one-stage formulations such as
 74 PVD and PSF, trading representational complexity for sampling simplicity. All of these target static
 75 geometry, and none model action-conditioned deformation.

76 **Articulated object reconstruction.** Reconstruction methods recover kinematic structure from an
 77 *observed* shape. Early work focuses on part segmentation and motion estimation from images or 3D
 78 scans [40, 41, 42, 43, 44], together with interaction-driven articulation inference such as Ditto [45]
 79 and Structure-from-Action [46]. More recent shape-to-URDF pipelines, including Real2Code [47],
 80 URDFormer [48], Watch-It-Move [49], and AutoURDF [28], translate geometry into executable
 81 robot descriptions, and image-conditioned URDF pipelines (URDF-Anything, Articulate-Anything)
 82 follow the same paradigm from a single view. These methods consume geometry rather than com-
 83 pete with generation; ArticFlow composes with AutoURDF as the simulation-readiness bridge in
 84 Section 4.3.

85 **Articulated object generation.** Generative approaches remain in their early stage. Image-
 86 conditioned pipelines such as SingAPO [29] and FreeArt3D [30] synthesize articulated geometry
 87 from a single view, mesh-conditioned methods deform or animate static meshes [50, 51], and graph-
 88 based frameworks such as CAGE [31] and NAP [52] generate kinematic graphs before geometry.
 89 These approaches rely on visual priors or predefined structures, which limit free-form variation
 90 and smooth cross-instance interpolation. By contrast, ArticFlow is self-contained and skeleton-free,
 91 learning directly from (X, A) pairs without intermediate structural representations.

92 **Neural robot simulation.** A closely related direction learns task-agnostic, differentiable kinematic
 93 models from deformation samples, exemplified by Visual Self-Modeling (VSM) [3] and FFKSM [4].
 94 These methods are fit per robot, whereas a single ArticFlow model spans a category. As task com-
 95 plexity rises for articulated mechanism generation, we revisit the two-stage design with flow match-
 96 ing to disentangle *shape* and *action*: conditioned only on actions, ArticFlow functions as a neural
 97 simulator, predicting kinematically valid deformations of a given instance; conditioned only on a
 98 shape prior, it generates diverse instances while preserving kinematic functionality.

99 3 Method

100 **Overview.** 3D point-cloud generation aims to learn the sampling of permutation-invariant point sets
 101 from the data distribution. Methods based on diffusion and flow matching learn a mapping from a
 102 prior distribution p_0 , typically Gaussian noise $\mathcal{N}(0, I)$, to the data distribution p_{data} . We instantiate
 103 flow matching for action-conditioned articulated 3D shapes.

104 **Skeleton-free setup.** Unlike prior articulated generation methods that consume a kinematic skeleton
 105 (joint axes, link topology, part segmentation), ArticFlow learns only from (X, A) pairs of point
 106 clouds and joint-angle vectors. No kinematic graph is provided.

107 Given a dataset of K pairs $\{(X^i, A^i)\}_{i=1}^K$ with $X^i \in \mathbb{R}^{N \times 3}$ and $A^i \in \mathbb{R}^J$, we encode the action as
 108 $Z_a = E_a(A)$ and learn a point flow $u_\theta(X_t, t \mid Z_a)$ that transports a Gaussian prior $p_0 = \mathcal{N}(0, I_{N \times 3})$
 109 to the articulated shape distribution. Training uses straight-line interpolation $X_t = (1-t)X_0 + tX_1$
 110 with conditional target velocity $u_t = X_1 - X_0$; the corresponding marginal velocity is denoted u^* .
 111 Section 3.2 extends this to a two-stage formulation by introducing a shape-prior latent Z_x and a
 112 second flow v_ψ , allowing a single model to cover a category.

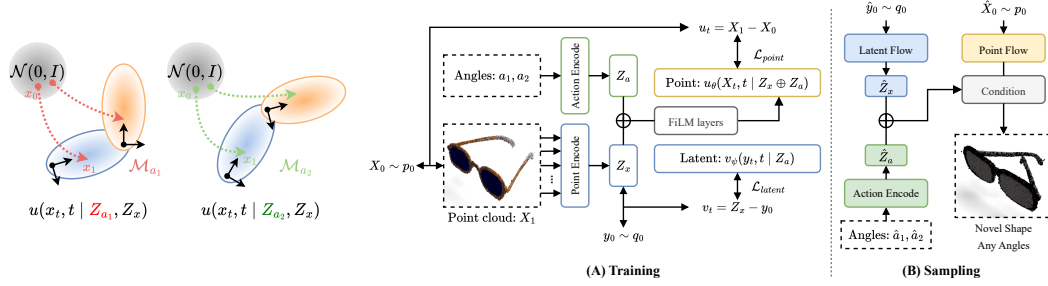


Figure 3: **Left: Action-conditioned velocity fields** (Section 3.1). With the shape prior Z_x fixed, the action embedding modulates $v(x, t \mid Z_a, Z_x)$, transporting Gaussian noise to action-specific manifolds $\mathcal{M}_a$. **Right: Two-stage flow matching.** (A) **Training** (Section 3.2): PointNet [23] extracts the shape latent Z_x and Fourier layers + MLP extract the action latent Z_a ; conditions are injected via Feature-wise Linear Modulation (FiLM) [53] layers, and the two flows are trained jointly with straight-line interpolation losses $\mathcal{L}_{\text{point}}$ and $\mathcal{L}_{\text{latent}}$. (B) **Sampling** (Section 3.3): sequential integration of the latent ODE ($\dot{y}_t = v_\psi$) followed by the point ODE ($\dot{X}_t = u_\theta$) produces the articulated point cloud $\hat{X}$.

3.1 Action conditioned flow

Flow matching [16] learns the velocity field of a continuous ODE [54] that transports samples from a prior p_0 to data p_{data} by regressing target velocities along straight-line interpolations. Since a given X_t may correspond to multiple (X_0, X_1) pairs, we learn the *marginal velocity* conditioned on action:

$$u^*(X, t \mid Z_a) = \mathbb{E}[u_t \mid X_t = X, t, Z_a]. \quad (1)$$

The flow matching loss regresses the network prediction against u_t :

$$\mathcal{L}_{\text{point}} = \mathbb{E}_{t, X_0, X_1} \left[\|u_\theta(X_t, t \mid Z_a) - u_t\|_2^2 \right], \quad (2)$$

with $t \sim \text{Beta}(2, 1)$, which biases toward late steps for stability. After training, samples are obtained by integrating

$$\dot{X}_t = u_\theta(X_t, t \mid Z_a), \quad \dot{X}_0 \sim p_0, \quad (3)$$

to yield $\hat{X}$ under the action code Z_a ; in practice we use Euler updates with step size h , or Heun’s method for higher accuracy.

3.2 Action and shape-prior conditioned flow

To cover a category with a single model, we introduce a shape-prior latent $Z_x = E_x(X) \in \mathbb{R}^D$ that summarizes each instance, and add a second flow $v_\psi(y_t, t \mid Z_a)$ mapping Gaussian noise $q_0 = \mathcal{N}(0, I_D)$ to the shape-prior space. The latent interpolation is $y_t = (1-t)y_0 + tZ_x$ with conditional target velocity $v_t = Z_x - y_0$, and the corresponding marginal velocity is v^* . The point flow is extended to $u_\theta(X_t, t \mid Z_a, Z_x)$. As shown in Figure 3 (A), we jointly optimize both flows with marginal velocities

$$u^*(X, t \mid Z_a, Z_x) = \mathbb{E}[u_t \mid X_t = X, t, Z_a, Z_x] \quad (4)$$

$$v^*(y, t \mid Z_a) = \mathbb{E}[v_t \mid y_t = y, t, Z_a]. \quad (5)$$

For training, we sample $X_0 \sim p_0$, $X_1 \sim p_{\text{data}}(\cdot \mid Z_a, Z_x)$, $y_0 \sim q_0$, and $t \sim \text{Beta}(2, 1)$ (biased toward late steps for stability), then optimize the squared error against u_t and v_t (Algorithm 1).

We concatenate Z_x, Z_a with the time-step embedding and inject them into the velocity networks via FiLM layers, which modulate intermediate activations through affine scale-and-shift transforms.

This factorization disentangles shape and action: v_ψ compresses geometric variability into a compact shape-prior distribution, while u_θ , conditioned on a sampled shape code, focuses solely on action-driven deformation. Sampling proceeds in this order, yielding independent control over instance identity and pose.

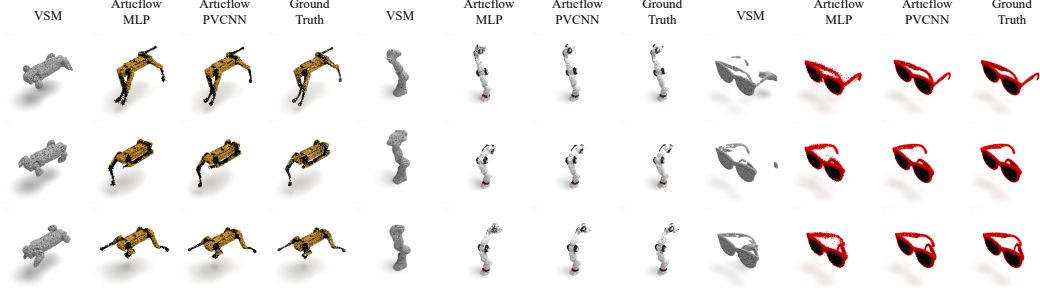


Figure 4: **Action-conditioned flow: qualitative results.** Deformed point clouds for fixed objects under varying joint actions: VSM, ArticFlow-MLP, ArticFlow-PVCNN, and ground truth.

3.3 Sampling and Interpolation

At sampling time (Figure 3 (B)), we encode an input action as $\hat{Z}_a = E_a(\hat{A})$, draw $y_0 \sim q_0$ and integrate $\dot{y}_t = v_\psi(y_t, t \mid \hat{Z}_a)$ from $t=0$ to 1 to obtain a shape code $\hat{Z}_x = y_1$, then draw $X_0 \sim p_0$ and integrate $\dot{X}_t = u_\theta(X_t, t \mid \hat{Z}_x, \hat{Z}_a)$ to obtain a novel articulated shape $\hat{X} = X_1$. Both ODEs are solved with Heun’s method (S_{lat} and S_{pt} steps).

For interpolation between two sampled codes $\hat{Z}_x^{(0)}, \hat{Z}_x^{(1)}$, we use spherical linear interpolation (slerp) on the unit hypersphere:

$$\text{slerp}(\hat{Z}_x^{(0)}, \hat{Z}_x^{(1)}; \alpha) = \frac{\sin((1-\alpha)\phi)}{\sin \phi} \hat{Z}_x^{(0)} + \frac{\sin(\alpha\phi)}{\sin \phi} \hat{Z}_x^{(1)},$$

where ϕ is the geodesic angle between the two latents.

Because PartNet-Mobility provides far fewer instances per category than static datasets like ShapeNet [25], the shape latent may still entangle kinematic information; we ablate the shape-flow design in Section 4.

4 Experiments

We evaluate ArticFlow along two axes: per-object *action-conditioned simulation* (does it follow commanded kinematics?) and category-level *shape-and-action generation* (can one model span a category and synthesize novel instances?). We then examine whether the generated shapes are dataset-level novel and can be lifted into simulation-ready assets.

Datasets. We validate on five categories: pliers, scissors, and eyeglasses from PartNet-Mobility, and robot arms and quadrupeds from MuJoCo Menagerie. Each PartNet-Mobility object provides 50 kinematic samples and each robot 1,000 action configurations, split 5:1 for train/test. PartNet point clouds use 20k points for flow training; robots use 4,096 points per sample (they supply far more samples per object), and every prediction is resampled to 20k points at evaluation for equal-cardinality metrics. As degrees of freedom (DoF) vary across instances within a category, we zero-pad action vectors to the per-category maximum; for robots, whose joint-rotation directions differ along the kinematic tree, we normalize rotations to the Denavit–Hartenberg (DH) convention before training.

Evaluation protocol. We assess whether ArticFlow produces kinematically correct shapes and accurately follows commanded actions. As ground-truth deformations for unseen objects are unavailable, we measure kinematic accuracy on latent codes decoded from dataset objects under unseen actions, and also study a single-object setting (one model per robot, action-only conditioning). Accuracy is reported as L2-Chamfer Distance (CD) and Earth Mover’s Distance (EMD) to the ground-truth clouds: CD captures overall shape similarity and is cheap to compute, while EMD is a transport-based measure more sensitive to local misalignment.

Simulation	CD ↓					EMD ↓				
	Arms	Quadrupeds	Pliers	Scissors	Eyeglasses	Arms	Quadrupeds	Pliers	Scissors	Eyeglasses
VSM [3]	12.4763	14.5676	0.0527	1.7208	10.7652	57.7256	66.9668	8.7149	45.9159	76.1755
ArticFlow(MLP)	0.0599	0.1853	0.1232	0.0780	0.1522	20.9800	15.9860	14.9094	14.6320	29.1480
ArticFlow(PVCNN)	0.6228	0.1675	0.1240	0.0481	0.1400	11.1780	5.8883	9.9892	8.5441	4.4339

Table 2: **Action-conditioned simulation: VSM baseline and point-flow backbone ablation.** CD and EMD values $\times 10^3$.

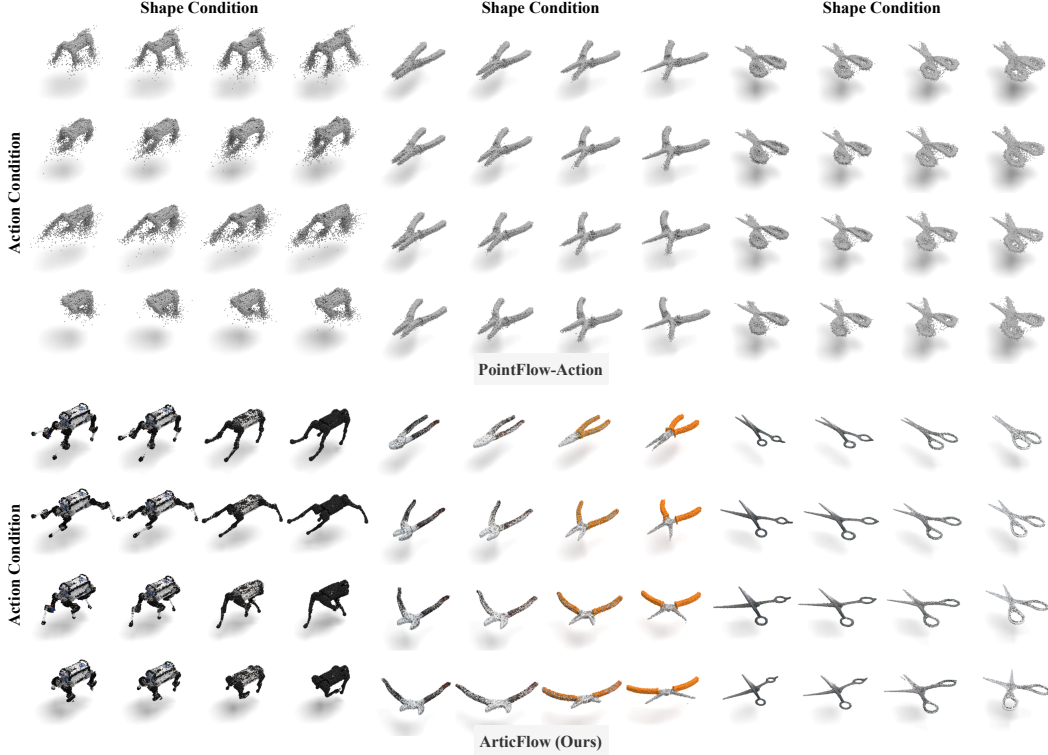


Figure 5: **Action and shape-prior conditioned flow, qualitative results.** ArticFlow vs. action-conditioned PointFlow baseline. Columns: shape prior (different instances). Rows: action condition (joint angles).

171 **Baselines.** We select baselines by task alignment. For action-conditioned simulation (Table 2) we
172 compare with VSM, a neural simulator for robot self-modeling from deformation samples and the
173 closest prior work; since VSM requires surface normals, we build a dedicated dataset from the same
174 point clouds, reconstructing watertight meshes and computing normals, so both methods train on
175 the same source. For shape-and-action generation (Table 3) we adapt PointFlow into an action-
176 conditioned generator by injecting the action through FiLM layers while keeping its CNF training
177 framework, as it shares our two-stage latent-point architecture. We do not compare with image-
178 or graph-conditioned methods (e.g., SingAPO, CAGE), which require inputs (images or kinematic
179 graphs) that ArticFlow does not assume.

180 4.1 Qualitative Results

181 For the action-conditioned setting (Fig. 4), with a fixed object under varying joint actions, ArticFlow
182 follows the target kinematics more closely than VSM, especially in thin structures such as eyeglass
183 legs and robot limbs; the Point-Voxel CNN (PVCNN) [55] backbone yields sharper, more complete
184 surfaces than the per-point MLP. For joint shape-and-action generation (Fig. 5), ArticFlow pro-
185 duces higher-quality surfaces and more kinematically consistent deformations across instances than

Generation	CD ↓					EMD ↓				
	Arms	Quadrupeds	Pliers	Scissors	Eyeglasses	Arms	Quadrupeds	Pliers	Scissors	Eyeglasses
PointFlow-Action	0.2824	1.4416	0.3591	0.4829	1.8308	62.8441	79.5037	73.3096	76.0229	182.6917
ArticFlow(Uncond-latent)	0.6061	0.2502	0.0281	0.0308	0.0207	16.8010	8.2525	4.4291	6.0803	4.3361
ArticFlow(Cond-latent)	0.6776	0.2651	0.0281	0.0295	0.0366	18.2628	8.7380	4.5394	6.2370	4.2984
ArticFlow(Adv-latent)	0.0616	0.2194	0.0306	0.0361	0.0311	9.8705	7.9411	4.6393	6.4390	3.6163

Table 3: **Action and shape-prior generation: PointFlow baseline and shape-latent design ablation.** CD and EMD values $\times 10^3$.

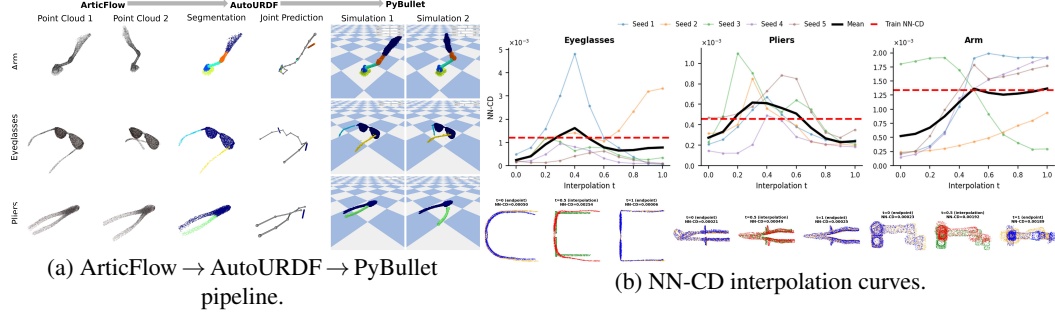


Figure 6: **Generation-to-simulation bridge and dataset-level novelty.** (a) ArticFlow point clouds (cols. 1–2) → AutoURDF [28] part segmentation (col. 3) and joint prediction (col. 4) → URDF actuated in PyBullet (cols. 5–6), for arm, eyeglasses, and pliers. (b) NN-CD between slerp-interpolated samples and the training set (5 seeds). The red dashed line is the average inter-instance distance within the training set (train–train NN-CD).

the action-conditioned PointFlow baseline, and natively generates color via 6D vectors (position + RGB), which VSM and PointFlow do not.

4.2 Quantitative Results

We report two settings, each ablating a different design choice. Table 2 (single-object, action-conditioned simulation) varies the *point-flow backbone* (per-point MLP vs. PVCNN); the shape-latent flow does not apply, as each model covers one object. Table 3 (category-level generation) varies the *shape-latent design* with the backbone fixed to PVCNN. Our primary configuration is Cond-latent + PVCNN (Table 1).

Action-conditioned simulation. ArticFlow outperforms VSM overall in CD and EMD (Table 2), with VSM competitive only on the low-DoF pliers, indicating more accurate kinematic deformation for appearance-fixed objects. The PVCNN backbone improves over the per-point MLP overall, most clearly in EMD, as its point-convolution module better captures local geometry and reduces transport cost to the ground truth.

Action and shape-prior generation. We evaluate three shape-latent designs against the action-conditioned PointFlow baseline (Table 3): **Cond-latent** (primary) injects the action into the shape-latent flow, per Section 3.2; **Uncond-latent** drops it, leaving disentanglement to the encoder; **Adv-latent** adds a gradient-reversal layer [56] to the unconditional shape flow to strip action information. All three outperform PointFlow by a wide margin and perform comparably overall. Adv-latent is strongest on the categories with the largest within-category shape variation, leading both metrics on arms and quadrupeds and EMD on eyeglasses, which suggests that discouraging kinematic information in the shape latent helps as within-category variation grows.

4.3 Generation Diversity and Simulation Readiness

Beyond accuracy against held-out actions, we examine whether ArticFlow generates *novel* shapes and whether those shapes can be lifted into *simulation-ready* assets.

Dataset-level novelty (Fig. 6b). We measure the nearest-neighbor Chamfer distance (NN-CD) between shapes sampled along a slerp interpolation $\hat{Z}_x^{(0)} \rightarrow \hat{Z}_x^{(1)}$ and the training set. As a threshold for what counts as “dataset-level novel,” we compute the train–train NN-CD, the average distance between training instances within a category. Mean NN-CD rises above the train–train baseline near $t=0.5$ across all three categories and returns toward the baseline at the endpoints. This pattern indicates that midpoint interpolations exceed dataset-level inter-instance distance (they lie outside the neighborhood of any single training instance), while endpoints recover training-level shape similarity. The arm category exhibits higher variance, reflecting the smaller pool of training morphologies.

Generation-to-simulation bridge (Fig. 6a). To verify that the generated geometry is usable downstream, we feed ArticFlow point-cloud sequences into AutoURDF, which recovers part segmentation, joint axes, and meshed links, and we actuate the resulting URDF in PyBullet. The pipeline succeeds end-to-end on arm, eyeglasses, and pliers categories, producing functional simulation assets from skeleton-free generation alone. This closes the loop from generated point clouds to controllable simulation, complementing the kinematic accuracy reported in Tables 2 and 3. As an additional check on geometric quality, generated point clouds also surface-reconstruct into watertight, color-preserving meshes (Appendix B).

5 Limitations

We outline the main limitations and the directions they open below.

Joint correspondence and topology. Within-category DoF variation is handled by zero-padding the action vector, so joint correspondence across instances follows serial order rather than anatomical role (e.g., index 5 may be a knee on one quadruped and a hip on another), the price of being skeleton-free. A fixed-skeleton alternative recovers near-perfect kinematics on the topologies it observes but cannot interpolate between morphologies with different DoF counts, and the same rigidity blocks extension toward soft or hybrid bodies. ArticFlow instead trades exact intra-topology precision for the ability to span a category of related-but-non-identical morphologies (Figure 6b). Relatedly, although the model accepts actions beyond the training range, extrapolated poses can become unrealistic at extreme joint angles or at distal joints when DoF counts differ across instances. Generalizing further, across DoF counts within a category and across rigid-to-soft boundaries, remains open.

Cross-category scope and physical plausibility. Generation is limited to categories seen in training: the model does not transfer zero-shot to unseen classes (e.g., quadrupeds to humanoids), and joint directions and ranges must be pre-aligned within a category (DH normalization for robots), which limits direct learning from unstructured real-world data. Our evaluation also targets *geometric* usability (kinematic accuracy and dataset-level novelty) rather than physical fidelity: the bridge of Section 4.3 produces actuated URDFs but does not yet validate collision-free configurations, mass distributions, or contact behavior. Closing this gap, and testing compatibility with related converters such as Real2Code and URDFormer, is necessary for downstream uses like morphological co-design or procedural generation for reinforcement learning.

Sampling efficiency. Inference requires sequential ODE integration for both flows, which is slower than single-pass generators; one-step or few-step flow variants [57, 58] are a promising direction.

6 Conclusion

We have presented ArticFlow, a two-stage flow matching framework for controllable articulated 3D generation. By disentangling shape priors and action-conditioned deformations into separate latent and point flows, ArticFlow trains one model to span diverse shapes within a category and generalize to novel joint configurations without requiring predefined kinematic structures. Experiments on PartNet-Mobility and MuJoCo Menagerie demonstrate consistent improvements over existing baselines in simulation accuracy and generation quality, dataset-level novelty of interpolated samples, and an end-to-end path from skeleton-free generation to functional URDFs actuated in PyBullet.

References

- [1] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.
- [2] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [3] B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson. Fully body visual self-modeling of robot morphologies. *Science Robotics*, 7(68):eabn1944, 2022.
- [4] Y. Hu, J. Lin, and H. Lipson. Teaching robots to build simulations of themselves. *Nature Machine Intelligence*, pages 1–11, 2025.
- [5] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.
- [6] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019.
- [7] L. Zhou, Y. Du, and J. Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021.
- [8] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.
- [9] L. Wu, D. Wang, C. Gong, X. Liu, Y. Xiong, R. Ranjan, R. Krishnamoorthi, V. Chandra, and Q. Liu. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9445–9454, 2023.
- [10] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [12] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.
- [13] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.
- [14] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [15] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [16] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- [17] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

- [18] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.
- [19] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023.
- [20] R. Shi, H. Chen, Z. Zhang, M. Liu, C. Xu, X. Wei, L. Chen, C. Zeng, and H. Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.
- [21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.
- [22] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.
- [25] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- [26] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.
- [27] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020.
- [28] J. Lin, L. Zhang, K. Lee, J. Ning, J. Goldfeder, and H. Lipson. Autourdf: Unsupervised robot modeling from point cloud frames using cluster registration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27628–27637, 2025.
- [29] J. Liu, D. Iliash, A. X. Chang, M. Savva, and A. Mahdavi-Amiri. Singapo: Single image controlled generation of articulated parts in objects. *arXiv preprint arXiv:2410.16499*, 2024.
- [30] C. Chen, I. Liu, X. Wei, H. Su, and M. Liu. Freeart3d: Training-free articulated object generation using 3d diffusion. *arXiv preprint arXiv:2510.25765*, 2025.
- [31] J. Liu, H. I. I. Tam, A. Mahdavi-Amiri, and M. Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024.
- [32] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

- [33] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [34] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.
- [35] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [36] J. Kim, J. Yoo, J. Lee, and S. Hong. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15059–15068, 2021.
- [37] S. Mo, E. Xie, R. Chu, L. Hong, M. Niessner, and Z. Li. Dit-3d: Exploring plain diffusion transformers for 3d shape generation. *Advances in neural information processing systems*, 36: 67960–67971, 2023.
- [38] A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, K. Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022.
- [39] G. Chen, M. Wang, Y. Yang, K. Yu, L. Yuan, and Y. Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *Advances in Neural Information Processing Systems*, 36: 29667–29679, 2023.
- [40] J. Liu, A. Mahdavi-Amiri, and M. Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 352–363, 2023.
- [41] J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.
- [42] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019.
- [43] S. Liu, S. Gupta, and S. Wang. Building rearticulable models for arbitrary 3d objects from 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21138–21147, 2023.
- [44] J. Huang, H. Wang, T. Birdal, M. Sung, F. Arrigoni, S.-M. Hu, and L. J. Guibas. Multi-body sync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7108–7118, 2021.
- [45] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.
- [46] N. Nie, S. Y. Gadre, K. Ehsani, and S. Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022.
- [47] Z. Mandi, Y. Weng, D. Bauer, and S. Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

- 389 [48] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta.
390 Urdformer: A pipeline for constructing articulated simulation environments from real-world
391 images. *arXiv preprint arXiv:2405.11656*, 2024.
- 392 [49] A. Noguchi, U. Iqbal, J. Tremblay, T. Harada, and O. Gallo. Watch it move: Unsupervised
393 discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF*
394 *Conference on Computer Vision and Pattern Recognition*, pages 3677–3687, 2022.
- 395 [50] D. Gao, Y. Siddiqui, L. Li, and A. Dai. Meshart: Generating articulated meshes with structure-
396 guided transformers. In *Proceedings of the Computer Vision and Pattern Recognition Confer-*
397 *ence*, 2025.
- 398 [51] X. Qiu, J. Yang, Y. Wang, Z. Chen, Y. Wang, T.-H. Wang, Z. Xian, and C. Gan. Articulate
399 anymesh: Open-vocabulary 3d articulated objects modeling. *arXiv preprint arXiv:2502.02590*,
400 2025.
- 401 [52] J. Lei, C. Deng, B. Shen, L. Guibas, and K. Daniilidis. Nap: Neural 3d articulation prior. *arXiv*
402 *preprint arXiv:2305.16315*, 2023.
- 403 [53] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with
404 a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*,
405 volume 32, 2018.
- 406 [54] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential
407 equations. *Advances in neural information processing systems*, 31, 2018.
- 408 [55] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-voxel cnn for efficient 3d deep learning. *Advances*
409 *in neural information processing systems*, 32, 2019.
- 410 [56] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and
411 V. Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning*
412 *research*, 17(59):1–35, 2016.
- 413 [57] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative
414 modeling. *arXiv preprint arXiv:2505.13447*, 2025.
- 415 [58] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. *arXiv*
416 *preprint arXiv:2410.12557*, 2024.

417 A Algorithms

418 Algorithm 1 details training: at each step we jointly fit the point flow u_θ and the latent flow v_ψ
 419 by flow matching on straight-line interpolants, drawing the interpolation time from a late-biased
 420 Beta(2, 1) for stability and combining the two losses with weight λ . Algorithm 2 details sampling:
 421 we first integrate the latent flow to draw a shape-prior code $\hat{Z}_x$, then integrate the point flow under
 422 $(\hat{Z}_x, \hat{Z}_a)$ to obtain the articulated point cloud, both with the second-order Heun integrator.

Algorithm 1 ArticFlow Training (Flow Matching). Each step jointly regresses two straight-line flow-matching objectives: the *point flow* u_θ (Gaussian prior $\rightarrow$ articulated point set X_1 , conditioned on shape and action) and the *latent flow* v_ψ (Gaussian prior $\rightarrow$ shape-prior code Z_x , conditioned on action), combined with weight λ .

Require: Dataset $\{(X^i, A^i)\}_{i=1}^K$; priors $p_0 = \mathcal{N}(0, I_{N \times 3})$, $q_0 = \mathcal{N}(0, I_D)$; encoders E_x, E_a ; velocity fields u_θ, v_ψ ; weight λ ; optimizer Opt.

- 1: **for** step = 1 to T **do**
- 2: Sample minibatch (X_1, A) ▷ point cloud + joint action
- 3: $Z_x \leftarrow E_x(X_1)$, $Z_a \leftarrow E_a(A)$ ▷ encode shape and action
- // Point flow: noise $\rightarrow$ articulated points*
- 4: Sample $t_x \sim \text{Beta}(2, 1)$, $X_0 \sim p_0$ ▷ late-biased time, prior
- 5: $X_t \leftarrow (1 - t_x) X_0 + t_x X_1$ ▷ straight-line interpolant
- 6: $u_t \leftarrow X_1 - X_0$ ▷ target velocity
- 7: $u \leftarrow u_\theta(X_t, t_x \mid Z_x, Z_a)$
- 8: $\mathcal{L}_{\text{point}} \leftarrow \text{MSE}(u, u_t)$
- // Latent flow: noise $\rightarrow$ shape-prior code*
- 9: Sample $t_z \sim \text{Beta}(2, 1)$, $y_0 \sim q_0$
- 10: $y_t \leftarrow (1 - t_z) y_0 + t_z Z_x$ ▷ straight-line interpolant
- 11: $v_t \leftarrow Z_x - y_0$ ▷ target velocity
- 12: $v \leftarrow v_\psi(y_t, t_z \mid Z_a)$
- 13: $\mathcal{L}_{\text{latent}} \leftarrow \text{MSE}(v, v_t)$
- 14: $\mathcal{L} \leftarrow \mathcal{L}_{\text{point}} + \lambda \mathcal{L}_{\text{latent}}$ ▷ joint objective
- 15: Opt.step($\nabla \mathcal{L}$) ▷ update $E_x, E_a, u_\theta, v_\psi$
- 16: **end for**

Algorithm 2 ArticFlow Sampling (Heun RK2). For a target action $\hat{A}$, the latent flow v_ψ first generates a shape-prior code $\hat{Z}_x$ from noise; the point flow u_θ is then integrated from noise under $(\hat{Z}_x, \hat{Z}_a)$ to yield the articulated point cloud $\hat{X}$. Both ODEs use the second-order Heun (improved-Euler) integrator.

Require: Trained fields v_ψ, u_θ ; action encoder E_a ; target action $\hat{A}$
 Priors $q_0 = \mathcal{N}(0, I_D)$, $p_0 = \mathcal{N}(0, I_{N \times 3})$; step counts $S_{\text{lat}}, S_{\text{pt}}$

- 1: **procedure** HEUN(field, x_0 , cond, S) ▷ integrate $\dot{x}$ = field over $[0, 1]$
- 2: $h \leftarrow 1/S$; $x \leftarrow x_0$; $t \leftarrow 0$
- 3: **for** $s = 1$ to S **do**
- 4: $k_1 \leftarrow \text{field}(x, t \mid \text{cond})$ ▷ slope at current state
- 5: $k_2 \leftarrow \text{field}(x + h k_1, t + h \mid \text{cond})$ ▷ slope after an Euler step
- 6: $x \leftarrow x + \frac{h}{2}(k_1 + k_2)$; $t \leftarrow t + h$ ▷ average the two slopes
- 7: **end for**
- 8: **return** x
- 9: **end procedure**
- 10: $\hat{Z}_a \leftarrow E_a(\hat{A})$ ▷ encode target action
- 11: $\hat{Z}_x \leftarrow \text{HEUN}(v_\psi, y_0 \sim q_0, \hat{Z}_a, S_{\text{lat}})$ ▷ sample shape-prior code
- 12: $\hat{X} \leftarrow \text{HEUN}(u_\theta, X_0 \sim p_0, (\hat{Z}_x, \hat{Z}_a), S_{\text{pt}})$ ▷ sample articulated points
- 13: **return** $\hat{X}$

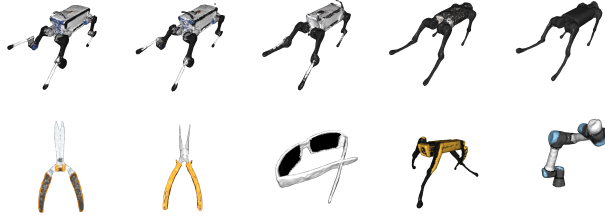


Figure 7: **Point cloud to mesh.** Interpolated point clouds from ArticFlow and their conversion to watertight meshes. Top row: interpolation between two legged robots. Bottom row: random samples from other categories. Color preserved from point cloud to mesh.

B Mesh Conversion

Figure 7 illustrates an alternative downstream route: interpolated and sampled ArticFlow point clouds are converted to watertight, color-preserving meshes via standard surface reconstruction, complementing the URDF bridge of Section 4.3.

C Experiment Analysis and Backbone Models

For the action-conditioned experiments, we adopt the baseline method from Visual Self-Modeling (VSM) and train it using ground-truth surface normals. We initially experimented with merging part meshes into a single watertight body mesh, but this produced unstable results on the robot datasets, suggesting that mesh smoothness has a strong impact on VSM training. Instead, we use ray-traced directions to consistently align the surface normals of each individual mesh, which yields stable baseline performance while still highlighting the advantages of our method in accurate, kinematically controlled shape generation (Figure 8a).

For the dual-conditioned experiments, we adapt PointFlow by adding per-layer FiLM conditioning modules that inject the action into the Point-CNF model, conceptually mirroring our unconditional latent-generation design but using CNF instead of flow matching. Varying the action condition produces only minor changes in kinematic deformation, the generated surfaces are noisier than ours, and training costs more on the same datasets (Figure 8b).

For the backbone, following prior work (PointFlow, PVD, PSF), we compare two architectures. The **MLP** backbone processes each point independently with residual FiLM MLP blocks of shared weights: each point is concatenated with a global embedding and passed through the network to predict per-point velocities, yielding an efficient $O(N)$ architecture that scales to large point clouds. The **PVCNN** backbone adds a multi-scale point-voxel context network (PVConv pyramid) that extracts local geometric features per point, fused with the global context through a learned time-gated blending that transitions from global-only to local+global conditioning to stabilize early-time integration from pure noise. Both backbones optionally support classifier-free guidance by randomly dropping the action and shape conditions during training.

To further verify that the learned flow respects the underlying kinematics, we visualize joint-wise action sweeps for both sampled and interpolated robots (Figure 9).

We attach the full codebase (dataset pipeline, training code, checkpoints) for reproducibility at the anonymous repository <https://anonymous.4open.science/r/articflow-submission-6CC8>.

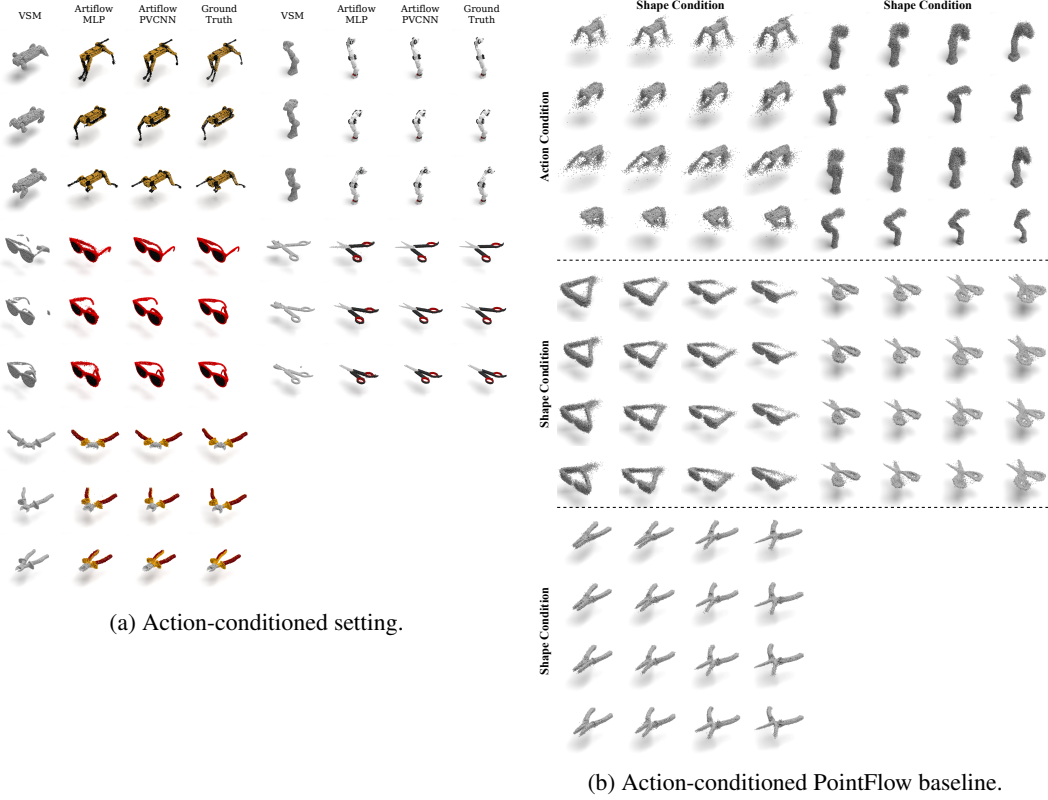


Figure 8: **Extended qualitative results.** **(a) Action-conditioned simulation**, extending Figure 4 to all five categories (quadrupeds, arms, eyeglasses, scissors, pliers). For a fixed object under varying joint actions, columns show VSM, ArticFlow (MLP), ArticFlow (PVCNN), and ground truth. VSM (gray) blurs geometry and collapses thin parts, whereas both ArticFlow variants recover the commanded articulation and match the ground truth; the PVCNN backbone yields sharper, more complete, color-preserving surfaces than the per-point MLP. **(b) Action-conditioned PointFlow baseline**, extending Figure 5 to all five categories: within each block, columns vary the shape condition and rows vary the action. The CNF baseline responds only weakly to the action and produces noisier, color-free surfaces, highlighting the advantage of our flow-matching formulation.

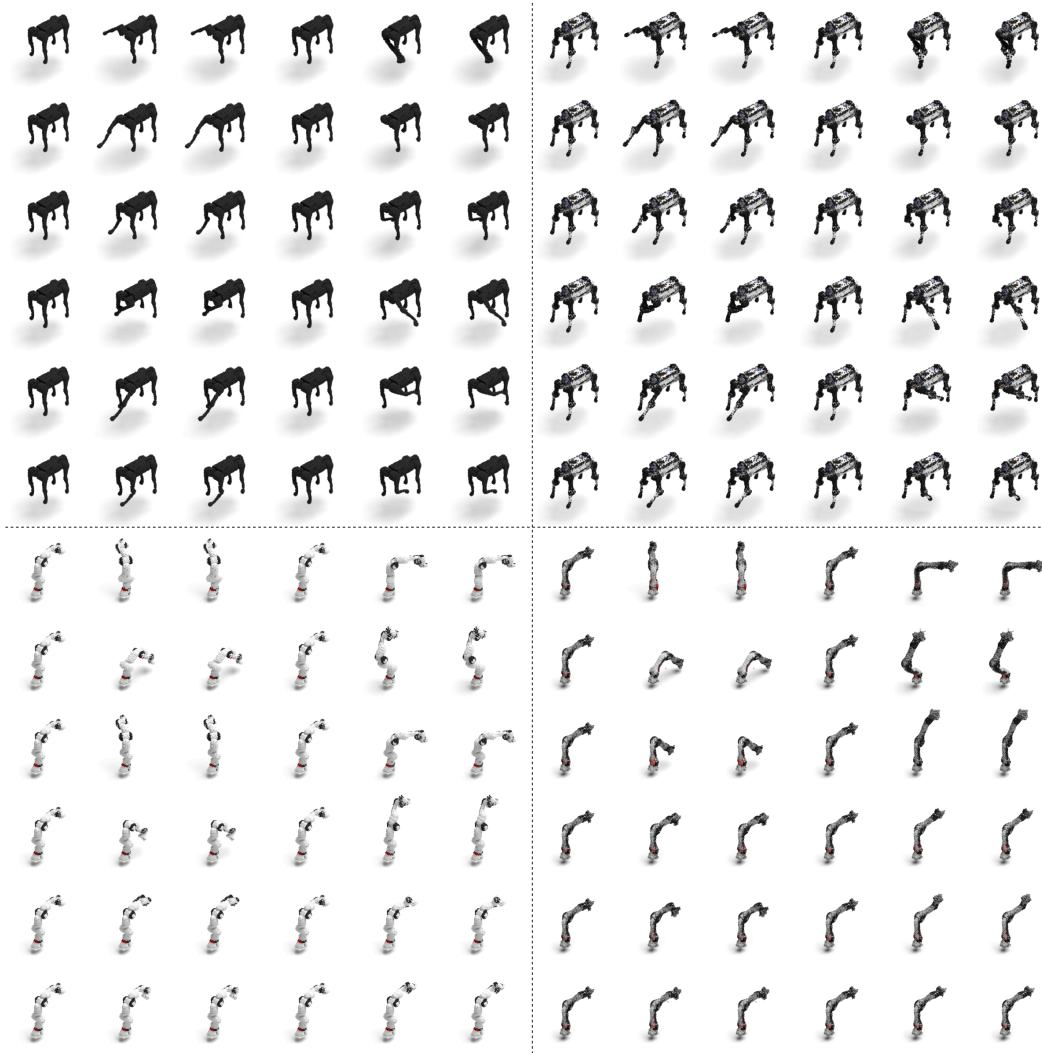


Figure 9: **Kinematic control validation.** We sweep joints one at a time to confirm that the learned point flow respects the underlying kinematics. In each 6×6 block, each row varies one of the first six joints across its range (left to right). Left column: a quadruped (top) and a robot arm (bottom) *sampled* from the dataset; right column: two *novel* shapes obtained by interpolating the shape-prior latent. For both real and interpolated morphologies, each swept joint produces a smooth, localized deformation consistent with the commanded action, indicating that action control transfers to unseen interpolated instances.