VELA: SCALING VISION-LANGUAGE-ACTION MODELS WITH COMPACT ACTION CURVE PARAMETRIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most vision-language-action models represent future motion as fixed-rate action chunks, tying temporal resolution and prediction horizon to a fixed output budget. This pointwise representation wastes capacity on highly correlated neighboring actions, leaves temporal continuity and smoothness to be learned implicitly, and forces a tradeoff between long-horizon coverage and the local precision required for contact-rich manipulation. To address these limitations, we introduce Vela, a vision-language-action foundation model that represents future robot behavior as continuous trajectories. Vela combines a compact spline-based action representation with motion-dependent temporal support and a shared action interface for heterogeneous embodiments, allowing a fixed output budget to adapt its temporal resolution across motions. We pretrain Vela on large-scale multi-embodiment robot data and evaluate it on LIBERO-X, EBench, and two real-world long-horizon tasks, egg-cake cooking and potato shredding, obtaining promising results across simulation and physical manipulation. These results highlight the potential of continuous action representations as a foundation for future embodied foundation models.

1 INTRODUCTION

Vision–Language–Action (VLA) models are rapidly scaling to larger datasets, broader task distributions, and more diverse robot embodiments (Intelligence et al., 2026; Yuan et al., 2026a; Ma et al., 2026). Yet most VLAs still represent future motion as a fixed-length sequence of actions sampled on a predetermined temporal grid. This pointwise interface assigns separate prediction capacity to highly correlated neighboring actions and couples the output budget to both prediction horizon and temporal resolution: representing a longer future requires more action points, while a fixed budget forces coarser sampling. Conversely, spreading the same budget over a long horizon can remove the local precision needed for contact-rich manipulation. Because temporal continuity is left to be learned implicitly, execution is also tied to a predefined sampling grid. These limitations reflect a mismatch between the representation used by current VLAs and the object they control: robot actions are not isolated points, but samples from continuous motions.

A trajectory-level representation provides a more natural action space. Instead of predicting every temporal sample, a policy can predict a compact set of parameters that defines a continuous trajectory and query that trajectory at the desired temporal locations. This representation compresses correlated actions, exposes temporal continuity, and decouples trajectory geometry from the time span over which it is executed. The same fixed parameter budget can therefore describe a long, smooth motion or a short, finely resolved one, while supporting different control frequencies. Splines provide a practical realization of this idea: a small set of control points specifies a continuous curve that can be evaluated on arbitrary temporal grids.

Trajectory-level representations have a long history in robotics, including dynamical and probabilistic movement primitives (Ijspeert et al., 2013; Paraschos et al., 2013). Recent learning-based policies have explored dynamical-system parameterizations (Bahl et al., 2020), differentiable trajectory optimization (Wan et al., 2024), and geometric curve representations (Li et al., 2026). Spline-based policies have further studied compact action tokenization, asynchronous execution, and temporal

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rescaling (Zhou et al., 2026; Yang et al., 2026; Han et al., 2026). These studies show the value of structured motion representations, but they mainly focus on task-specific policies, individual robot platforms, or downstream action tokenization and execution. Whether trajectory-level action modeling can serve as the output space of foundation-scale VLA pretraining across large and heterogeneous robot datasets remains largely unexplored.

To address this gap, we build a VLA that learns future motion in trajectory space and make this representation trainable at foundation scale. A shared cross-embodiment action interface and scalable supervision procedure allow the same model to absorb heterogeneous robot corpora. We further make the temporal span of each prediction motion-dependent: with a fixed number of spline control points, each demonstrated motion is assigned a sample-specific horizon by selecting the longest span that adequately represents it under a noise-aware criterion. Smooth motions can therefore use longer spans, whereas motions with rapid local variation can use shorter spans without changing the output budget. We train this framework on large multi-embodiment robot data and evaluate its transfer to downstream manipulation tasks.

Our main contributions are:

- We show how trajectory-space action modeling can be scaled to VLA pretraining through a shared cross-embodiment action interface and a training procedure designed for large heterogeneous robot corpora.
- We introduce motion-dependent horizon adaptation, selecting for each demonstrated motion the longest temporal span that remains adequately represented by a fixed spline budget under a noise-aware criterion.
- We evaluate the framework with controlled comparisons, downstream manipulation benchmarks, and ablations, showing improvements from trajectory-space modeling, large-scale pretraining, and horizon adaptation, particularly under contact-rich, long-horizon, and distribution-shifted tasks.

2 Related Work

Vision-Language-Action Foundation Models. Vision-language-action (VLA) foundation models scale robot learning across heterogeneous tasks, data, and embodiments by coupling pretrained vision-language representations (Beyer et al., 2024; Bai et al., 2025) with generalist action policies (Team et al., 2024; Black et al., 2024; Bjorck et al., 2025). Recent systems further improve cross-task and cross-embodiment generalization through heterogeneous co-training, embodiment alignment, and predictive supervision (Intelligence et al., 2025; 2026; Yuan et al., 2026a; Zheng et al., 2026; Cai et al., 2026a; Ma et al., 2026). Yet the low-level motion interface of modern chunk-based VLAs remains largely organized around fixed-resolution sequences of sampled future actions. Recent work begins to relax this temporal rigidity: MoH (Jing et al., 2025) models multiple prediction horizons, while AutoHorizon (Wang et al., 2026b) and AAC (Liang et al., 2026) execution lengths using attention patterns and action entropy. These methods still operate on sampled action sequences. In contrast, Vela makes temporal support part of the action representation itself, using a fixed set of control points to parameterize continuous trajectories with motion-adaptive future spans.

Spline-Based Robot Policies. Structured representations of robot motion have long been studied through dynamical and probabilistic movement primitives (Ijspeert et al., 2013; Paraschos et al., 2013). Recent work incorporates splines into learned policies for efficient action tokenization (Zhou et al., 2026), asynchronous execution Yang et al. (2026), and structured action generation across policy backbones Tian et al. (2026). B-spline Policy Han et al. (2026) focuses on accelerating visuomotor policies through adaptive spline fitting and temporal rescaling. These works primarily study splines as policy-level representations, tokenizers, or execution interfaces. To our knowledge, Vela is the first VLA foundation model established in trajectory space through large-scale pretraining on multi-embodiment robot data. Our framework is designed to support this scaling: a unified action interface accommodates heterogeneous embodiments, while a noise-calibrated horizon procedure constructs spline targets on the fly. Together, Vela constructs consistent trajectory targets from heterogeneous datasets, investigates spline trajectories as the output space of a foundation model.

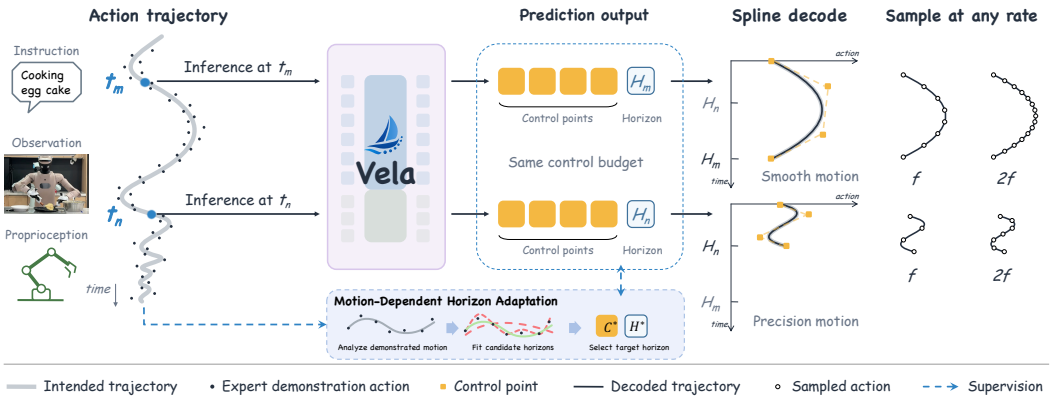


Figure 1: Overview of Vela. At inference, Vela jointly predicts a fixed set of spline control points and a motion-dependent horizon, defining a continuous action trajectory that can be sampled at arbitrary control rates. During training, *motion-dependent horizon adaptation* constructs supervision targets that adequately represents the demonstrated motion. The same control point budget thus covers longer spans for smooth motion and shorter spans when finer temporal resolution is needed.

3 Methodology

We develop Vela by treating demonstrated actions as samples of a continuous trajectory and learning to predict a compact representation of that trajectory. Building on this formulation, we describe the policy, construct motion-dependent horizons, and scale training to heterogeneous robot data.

3.1 Preliminaries: What should a VLA predict?

Action prediction. Conventional VLA policies represent future motion as an action chunk conditioned on the current observation and instruction. At timestep t, let $z_t = (\mathbf{v_t}, \mathbf{s_t}, \ell)$ denote the multimodal policy context, consisting of visual observations $\mathbf{v_t}$, proprioceptive state $\mathbf{s_t}$, and a language instruction ℓ. A conventional Vision-Language-Action (VLA) policy parameterizes a conditional distribution over a future action chunk,

$$\mathbf{A}_{t,H} \sim \pi_\theta(\cdot \mid z_t) \qquad \mathbf{A_{t,H}} = [\mathbf{a_t}, \mathbf{a_{t+1}}, \ldots, \mathbf{a_{t+H-1}}]^\top \in \mathbb{R}^{H \times d_a}, \tag{1}$$

where H is the prediction chunk horizon and d_a is the action dimension. Regardless of the generative objective, this representation describes motion through its values on a discrete temporal grid.

From actions to trajectories. We instead view a demonstrated action chunk as a sampled realization of an underlying continuous motion. For a given future window, let $\gamma_t : [0,1] \to \mathbb{R}^{d_a}$ denote its *latent* intended trajectory, and write the recorded actions as

$$\mathbf{a}_{t+i} = \boldsymbol{\gamma}_t(u_i) + \boldsymbol{\epsilon}_{t,i}, \qquad u_i = \frac{i}{H-1}, \quad i = 0, \ldots, H-1, \tag{2}$$

where $\boldsymbol{\epsilon}_{t,i}$ captures deviations introduced during demonstration and execution. This view separates the motion to be represented from a particular noisy recording of it. This modeling view motivates a structured approximation to the underlying motion rather than adopting its sampled realization as the action representation.

To obtain a finite-dimensional representation (De Boor, 1968; De Boor & De Boor, 1978), we approximate the path $\boldsymbol{\gamma}_t$ using a B-spline curve,

$$\hat{\gamma}_t(u) = \sum_{n=0}^{N-1} b_n(u)\, \mathbf{c}_{t,n}, \qquad u \in [0,1], \tag{3}$$

where $\mathbf{b}_n$ are cubic B-spline basis functions with uniform knots, and $\mathbf{c}_{t,n} \in \mathbb{R}^{d_a}$ are control points. The resulting curve is determined by $\mathbf{C}_t = [\mathbf{c}_{t,0}, \ldots, \mathbf{c}_{t,N-1}]^\top \in \mathbb{R}^{N \times d_a}$. The continuous motion coordinates are C^2-continuous at simple interior knots within each predicted segment and can be queried at arbitrary temporal locations. The name Vela, inspired by sailing, reflects this view of future motion as a continuous course described by a compact set of parameters.

3.2 VELA MODEL ARCHITECTURE

Policy backbone. Vela follows the $\pi_{0.5}$ (Intelligence et al., 2025) architecture, consisting of a multimodal vision-language backbone and a flow-matching action expert. The backbone encodes the policy context as $\mathbf{g} = G_\phi(z)$, and the action expert F_θ defines the conditional flow

$$\frac{\mathrm{d}\mathbf{y}^\tau}{\mathrm{d}\tau} = F_\theta(\mathbf{y}^\tau, \tau; \mathbf{g}), \qquad \mathbf{y}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \tau \in [0, 1], \tag{4}$$

whose terminal state specifies the predicted motion. Here, τ denotes generative flow time.

Trajectory prediction. Rather than representing future motion by samples on a fixed temporal grid, Vela parameterizes each future motion by two complementary quantities: its geometry and its temporal span. The geometry is parameterized by a set of N control points $\mathbf{C}$, which specify the curve, while a scalar horizon H specifies its extent in action steps. We model them jointly as $\mathbf{y} = (\mathbf{C}, H)$ and obtain their prediction from $\mathbf{y}^1$. In implementation, H occupies an additional scalar output channel alongside the control-point coordinates. Thus, the number of predicted control points remains fixed, while the span of future motion they describe can vary with the context.

Action decoding. The horizon maps the normalized curve to the future action timeline: $\mathbf{a}(s) = \hat{\gamma}(s/(H-1))$, $s \in [0, H-1]$. Evaluating at integer offsets gives

$$\hat{\mathbf{A}}_H = \mathbf{B}_H \mathbf{C}, \qquad [\mathbf{B}_H]_{i,n} = b_n\left(\frac{i}{H-1}\right), \tag{5}$$

where $\mathbf{B}_H \in \mathbb{R}^{H \times N}$. Other query grids change the sampling density without changing the predicted representation. The remaining question is how to choose a suitable horizon for each demonstrated motion under the fixed control-point budget.

3.3 MOTION-DEPENDENT HORIZON ADAPTATION

With N fixed, different temporal spans induce different allocations of the same representational capacity. Extending the horizon H distributes the same curve parameters over more action steps. This is well suited to smooth and directionally consistent behaviors such as reaching and transport. Conversely, contact, alignment, and rapid corrective motion can require the parameters to be concentrated over a shorter interval in order to preserve local temporal variation. Existing spline-based policies typically predict with a globally fixed temporal span (Zhou et al., 2026; Tian et al., 2026; Yang et al., 2026), thereby imposing a single compromise between temporal coherence and local expressivity on all motions.

We instead formulate this trade-off as a horizon-selection problem and seek the longest temporal span that remains sufficient for the demonstrated motion. Let $\mathcal{H}$ denote the candidate horizons, each satisfying $H > N$:

$$H^\star = \max\{H \in \mathcal{H} : \psi(H) \le \kappa\}, \tag{6}$$

where $\psi(H)$ measures whether the demonstrated motion is adequately represented at span H and κ is a shared tolerance. The central problem is therefore to construct $\psi(H)$ such that insufficient temporal resolution can be distinguished from variation already inherent in the demonstrations.

Noise-aware horizon selection. Here, noise refers to variation introduced by teleoperation, sensing, and execution rather than the underlying motion. We estimate a per-source, per-coordinate reference scale from normalized demonstrations and use it to distinguish unresolved motion structure from variation that is already present in the demonstrations. We assess each candidate horizon by fitting its demonstrated action sequence with the spline basis from equation 5:

$$\mathbf{C}_H = \mathbf{B}_H^\dagger \mathbf{A}_H, \tag{7}$$

where $\dagger$ denotes the Moore–Penrose pseudoinverse. As suggested by equation 2, the fitting residual $r(H) = \mathbf{A}_H - \mathbf{B}_H \mathbf{C}_H$ contains both motion not captured by the spline and demonstration variation. Its raw magnitude alone therefore does not indicate whether the temporal span is too long.

To obtain a reference scale, consider a continuous action coordinate j and assume that its demonstration perturbations $E[\epsilon^{(j)}] = 0$ and $\mathrm{Cov}[\epsilon^{(j)}] = \sigma_j^2 \mathbf{I}$. For a fixed underlying trajectory, let $\boldsymbol{\gamma}_j$ denote its samples over the candidate window. The expected residual decomposes as

$$\mathbb{E}[\|\mathbf{r}_j(H)\|_2^2] = \|(\mathbf{I} - \mathbf{B}_H \mathbf{B}_H^\dagger)\boldsymbol{\gamma}_j\|_2^2 + (H - N)\sigma_j^2, \tag{8}$$

which consists of the first term unrepresented motion and the second term demonstration variation. Thus even when the underlying motion is fully representable, the residual is nonzero and scales with both the noise variance and the residual degrees of freedom $H - N$. A raw-error threshold would mix these effects with the approximation error that horizon selection is intended to control.

We therefore normalize the residual by its noise reference:

$$\psi^2(H) = \frac{1}{|\mathcal{J}|(H-N)} \sum_{j \in \mathcal{J}} \frac{\|\mathbf{r}_j(H)\|_2^2}{\hat{\sigma}_j^2}, \tag{9}$$

where $\mathcal{J}$ contains the continuous action dimensions and $\widehat{\sigma}_j$ is estimated from temporal second differences of normalized demonstrations, separately for each data source. Under the reference noise model and accurate scale estimates, the expectation of $\psi^2(H)$ is approximately one for representable motion and increases with unresolved structure. This calibration reduces the tendency to shorten a horizon merely to fit demonstration fluctuations.

Gripper commands are evaluated separately because their sparse transitions are not appropriately assessed by division by a near-zero noise scale. Including such channels in the noise-aware statistic equation 9 would therefore be both numerically ill-conditioned and semantically inappropriate. Instead, for each such valid channel $j \notin \mathcal{J}$, we reject candidate spans whose normalized projection error $\|\mathbf{r}_j(H)\|_2/\sqrt{H}$ exceeds a small tolerance η.

On-the-fly target construction. The selected fit yields the supervision target $\mathbf{y}^\star = (\mathbf{C}^\star, H^\star)$, with $\mathbf{C}^\star = \mathbf{C}_{H^\star}$. Candidate fits and residuals are independently evaluated using batched matrix operations for each sampled transition. This avoids a separate trajectory annotation pass or iterative knot optimization and supports shuffled training over large heterogeneous robot corpora. At deployment, no fitting or horizon search is performed: the policy predicts both quantities from the current context.

3.4 Trajectory-Space Pretraining and Objectives

We extend the target construction above to large-scale trajectory-space pretraining through a shared action representation and flow-matching objective. The action interface accommodates different embodiments, while source-specific normalization accounts for differences in motion statistics.

Shared action space. To scale training across heterogeneous robot datasets, we map all data sources to a shared $d_a = 26$ dimensional action-path space. Each arm contributes 10 dimensions describing end-effector motion, including 3-D translation, 6-D rotation, and a 1-D gripper command. The remaining 6 dimensions describe the head, waist, and base motion with two coordinates each. Per-dimension validity masks exclude unavailable coordinates from target selection and supervision.

Each source uses its own per-coordinate affine quantile normalization and estimates its noise scales $\hat{\sigma}_j$ in the same normalized coordinates. This allows the shared tolerance κ to act on a dimensionless, noise-normalized statistic rather than on source-dependent raw errors. Because spline fitting acts along time and independently across action coordinates, the same fixed bases and horizon adapation procedure apply throughout the training mixture.

Training Objective. Given a target $\mathbf{y}^\star = (\mathbf{C}^\star, H^\star)$, Gaussian noise $\mathbf{y}^0$, and flow time τ, the interpolated input is $\mathbf{y}^\tau = (1-\tau)\mathbf{y}^0 + \tau \mathbf{y}^\star$. We write the control-point and horizon components of the flow prediction error as

$$(\mathbf{E}_C, \mathbf{E}_H) = F_\theta(\mathbf{y}^\tau, \tau; \mathbf{g}) - (\mathbf{y}^\star - \mathbf{y}^0). \tag{10}$$

For the trajectory geometry, a direct choice is to apply the standard flow-matching objective to the control point coordinates (CP-FM): $\mathcal{L}_{\mathrm{CP}} = \mathbb{E}[\|\mathbf{E}_C\|_F^2/(Nd_a)]$. This objective weights prediction errors isotropically in the control point space, without accounting for their different effect on the decoded trajectory depends on the spline basis. We instead use decoded-trajectory flow matching (DT-FM), which evaluates the same error after applying the linear decoder:

$$\mathcal{L}_{\mathrm{DT}} = \mathbb{E}[\frac{\|\mathbf{B}_{H^\star}\mathbf{E}_C\|_F^2}{H^\star d_a}]. \tag{11}$$

Equivalently, DT-FM replaces the isotropic weighting of CP-FM with the decoder-induced quadratic weighting $\mathbf{B}_H^\top \mathbf{B}_H / H$, thereby emphasizing control-point errors according to their effect on the

decoded trajectory. DT-FM is our default geometry objective. Its relation to CP-FM and the effect of this weighting are examined in Appendix A.2 and Sec. 4.4.

The temporal-support component H is trained with the standard flow-matching objective, denoted by $\mathcal{L}_H = \mathbb{E}[\|E_H\|^2]$. Our final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{DT}} + \lambda_H \mathcal{L}_H. \tag{12}$$

4 Experiments

4.1 Experimental Setup

Implementation Details. Vela follows the $\pi_{0.5}$ (Intelligence et al., 2025) architecture, comprising a 3B PaliGemma (Beyer et al., 2024) vision-language backbone and a 300M flow-matching action expert, without introducing additional trainable parameters. Our study examines trajectory-space action modeling at scale without relying on a larger network, a stronger VLM, or additional input information. Starting from pretrained PaliGemma weights, we pretrain Vela directly in trajectory space on approximately 20,000 hours of public robot data from 60 datasets spanning 11 embodiment classes, together with approximately 20,000 hours of private robot data. For comparison, we also introduce $\pi_{0.5}$-Vela, which is initialized from the same $\pi_{0.5}$ base checkpoint as the point-wise baseline. It applies our trajectory representation and supervision during benchmark specific post-training, without undergoing our trajectory-space pretraining stage.

Each data source is mapped to the shared 26-D action-path interface described in Sec. 3.4 with source-specific normalization and per-dimension validity masks. We use cubic B-splines with a fixed budget of $N = 12$ control points and candidate horizons $\mathcal{H} = \{16, 17, \ldots, 40\}$. At inference time, the conditional flow is integrated using 10 Euler steps. Additional training and implementation details are provided in Appendix. A.2.1.

Benchmarks and Evaluation. We evaluate Vela on two complementary simulation benchmarks, LIBERO-X and EBench, as well as through real-world experiments on a wheeled dual-arm robot with 7-DoF arms and three cameras mounted on the head and wrists. The real-world study considers two long-horizon tasks that require bimanual coordination, tool use, and contact-rich manipulation. We report egg-cake cooking task and potato-shredding task in Sec. 4.3 and Appendix A.4, respectively. **LIBERO-X** (Wang et al., 2026a) is a MuJoCo-based tabletop manipulation benchmark evaluating generalization across five levels of progressively stronger spatial, visual, and instruction perturbations. We report success rate (SR) at each level and overall. **EBench** (Gao et al., 2026) is an Isaac Sim benchmark for generalist dual-arm mobile manipulation, comprising 26 tasks across three complementary regimes. Its evaluation covers background, object, instruction, and mixed distribution shifts. We report SR and the official task-progress Score for each task family and overall.

For each simulation benchmark, we follow its official $\pi_{0.5}$ fine-tuning recipe and evaluation protocol. Further training and execution details can be found in A.2.2.

4.2 Simulation Evaluation

Simulation Evaluation on LIBERO-X. LIBERO-X evaluates robust manipulation under progressively stronger distribution shifts. As shown in Tab 1, Vela achieves the highest average performance among the compared methods. Its advantage over $\pi_{0.5}$ is more pronounced at levels 3 to 5 than at levels 1 and 2, indicating that the gains extend to settings with a more strong distribution shift.

Introducing trajectory-space modeling during benchmark post-training also improves performance. Starting from the same base checkpoint and using the same finetuning recipe as $\pi_{0.5}$, $\pi_{0.5}$-Vela improves SR at every level and reaches 42.0% on average, compared with 39.3% for the point-wise baseline. The results support trajectory-space action modeling both for adapting an existing VLA and for pretraining a robot policy over heterogeneous robot data.

Simulation Evaluation on EBench. EBench evaluates generalist manipulation under diverse control demands, spanning *dexterous-and-precise tabletop interaction* involving sub-centimeter insertion, alignment, and bimanual coordination, *mobile pick-and-place* (PnP), and *long-horizon mobile* tasks composed of extended multi-stage sequences. As shown in Tab. 2, Vela achieves the highest

Table 1: Evaluation on LIBERO-X (Wang et al., 2026a) across five progressively challenging levels. We report success rate (%), with green values indicating absolute gains over baseline $\pi_{0.5}$. $\pi_{0.5}$-Vela denotes variant initialized from $\pi_{0.5}$ and post-trained in trajectory space.

Model	Level 1	Level 2	Level 3	Level 4	Level 5	Avg.
OpenVLA-OFT (Kim et al., 2025)	29.0	17.6	8.8	6.4	4.2	13.2
π_0 (Black et al., 2024)	29.4	21.9	11.0	7.6	5.1	15.0
X-VLA (Zheng et al., 2026)	30.1	22.6	10.3	6.0	4.1	14.6
OpenWAM-α (Wang et al., 2026c)	35.9	28.2	21.6	13.0	10.0	21.7
GR00T N1.5 (Bjorck et al., 2025)	43.3	32.9	18.7	13.3	9.7	23.6
τ_0-VLA (Cai et al., 2026c)	46.5	34.2	20.5	13.0	10.5	24.9
Xiaomi-R0 (Cai et al., 2026b)	**71.4**	49.6	33.7	22.1	17.2	38.8
$\pi_{0.5}$ (Intelligence et al., 2025)	65.2	53.2	36.0	24.1	18.0	39.3
$\pi_{0.5}$-Vela	66.1	53.5	40.7	27.3	22.5	42.0
Vela	67.2 (+2.0)	**55.9** (+2.7)	**44.9** (+8.9)	**31.5** (+7.4)	**27.1** (+9.1)	**45.3** (+6.0)

Table 2: Evaluation on EBench (Gao et al., 2026) across its three manipulation task families. We report success rate (SR, %) and the official task-progress Score (Score, %) with green values indicating the absolute gains over baseline $\pi_{0.5}$.

Model	Table Top		Simple PnP		Long Horizon		Overall	
	SR	Score	SR	Score	SR	Score	SR	Score
X-VLA (Zheng et al., 2026)	8.6	24	50.0	54	6.2	25	23.7	36
InternVLA-A1 (Cai et al., 2026a)	4.3	11	43.0	47	17.9	46	23.9	36
FastWAM (Yuan et al., 2026b)	2.9	13	49.5	53	16.9	38	25.6	37
Cosmos3-Edge (Agarwal et al., 2026)	18.6	32	41.5	44	24.1	49	29.3	42
GigaBrain-0.7 (Team et al., 2026)	37.9	59	42.0	45	20.0	37	33.3	46
π_0 (Black et al., 2024)	32.9	49	41.5	45	25.7	49	33.7	47
InternVLA-A1.5 (Ma et al., 2026)	12.1	22	50.0	53	33.7	59	34.2	46
Qwen-RobotManip (Yuan et al., 2026a)	**50.0**	**70**	56.5	60	29.9	55	45.6	60
$\pi_{0.5}$ (Intelligence et al., 2025)	29.3	45	56.5	61	34.1	55	41.4	54
$\pi_{0.5}$-Vela	34.3	54	59.0	63	28.5	55	42.0	58
Vela	44.3 (+15.0)	64 (+19)	**63.0** (+6.5)	**67** (+6)	**39.1** (+5.0)	**65** (+10)	**49.7** (+8.3)	**66** (+12)

overall SR and task-progress Score among the compared methods, and leads both metrics on PnP and Long-Horizon tasks. Relative to $\pi_{0.5}$, its largest improvement occurs on dexterous-and-precise tabletop manipulation which demands on fine-scale motion representation.

Trajectory space finetuning variant $\pi_{0.5}$-Vela also improves overall performance to 42.0% SR and a Score of 58, with family-wise improvements on both Table Top and PnP tasks. Additionally, Vela, pretrained directly in trajectory space, outperforms both $\pi_{0.5}$ and $\pi_{0.5}$-Vela on all metrics. These results demonstrate the breadth of the pretrained trajectory-space policy across precision-intensive interaction and extended execution, rather than an advantage confined to a single manipulation regime. Further result details are provided in Appendix A.3.

4.3 Real-World Evaluation

We further evaluate Vela on a real-world multi-stage *cooking egg cake* task using the wheeled dual-arm robot. The workspace contains a cooking pan, a batter ladle, ingredient containers, a spatula, and a serving plate. As illustrated in Fig. 2, the task comprises four ordered stages:

1. **Add batter and return the ladle:** fill the ladle with batter, pour the batter into the pan, and return the ladle.
2. **Add ingredients and return the ladle:** pick up the ingredients, add them to the egg cake, and return the ladle.
3. **Grasp the pan with the left hand:** move the left arm to the pan and grasp it firmly.
4. **Remove, plate, and finish:** pick up the spatula with the right hand, remove the egg cake from the pan, place it on the plate, and complete the finishing actions.

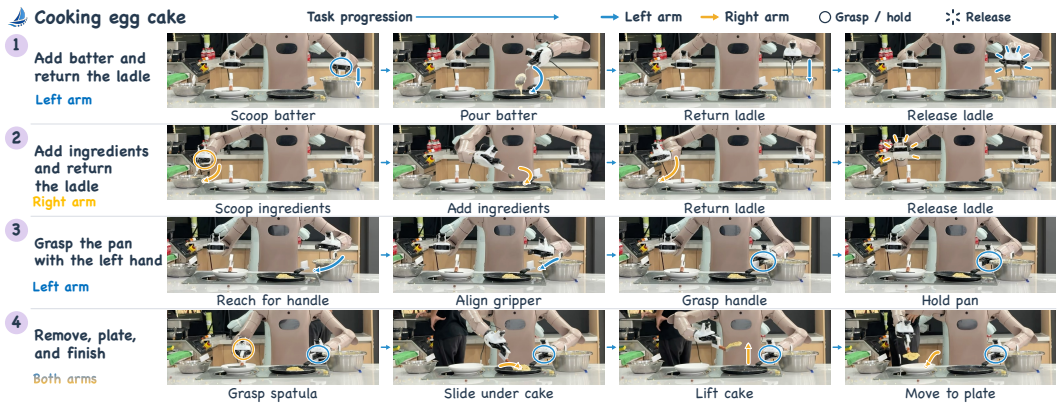


Figure 2: Vela executing *cooking egg cake*, which spans four subtasks (top to bottom).

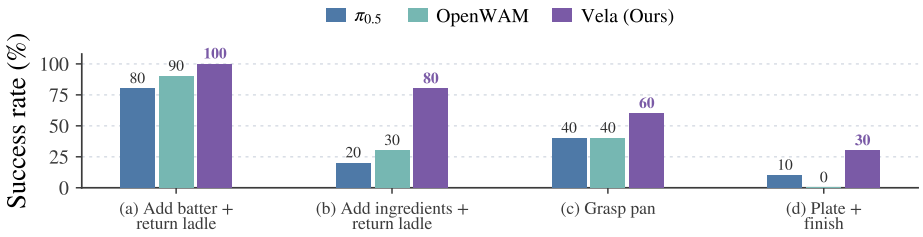


Figure 3: Real-world results on the *cooking egg cake* task across four subtasks.

Each subtask is evaluated over 10 trials. Figure 3 compares Vela with $\pi_{0.5}$ and OpenWAM. Vela achieves the highest success rate on all four subtasks, with a mean subtask success rate of 67.5%, compared with 37.5% for $\pi_{0.5}$ and 40.0% for OpenWAM. The largest margin occurs during ingredient transfer, while the advantage persists for pan grasping and final plating. These results provide physical-robot evidence that Vela's trajectory-space policy remains effective beyond simulation, particularly during long-horizon sequencing, bimanual coordination, tool use, and contact-rich manipulation. Additional real-world results on potato shredding are provided in Appendix A.4. More visualizations of the real-world experiments are provided in the supplementary video.

4.4 Ablation Studies

We ablate the core design of Vela. We examine whether trajectory-space learning offers benefits beyond post-hoc spline fitting, how motion-dependent horizons affect the use of a fixed control-point budget, and whether the choice of geometry supervision is critical. These ablations distinguish the role of the learned representation from its temporal allocation and loss design.

① **Can Post-hoc Trajectory Fitting Recover the Gains?** Since trajectory fitting can provide a continuous path for action resampling, one may ask whether the benefit of Vela can simply come

Table 3: Ablation of post-hoc trajectory fitting (Post-hoc Traj. Fit.) on LIBERO-X. Post-hoc Traj. Fit. spline fits and resamples predicted point-wise action chunk, whereas Vela predicts trajectory parameters directly. Max. Steps denotes the maximum number of actions executed before replanning.

Configuration			Success Rate (%)					
Method	**Post-hoc Traj. Fit.**	**Max. Steps**	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Avg.**
$\pi_{0.5}$	✗	10	65.2	53.2	36.0	24.1	18.0	39.3
	✓	10	61.7	50.5	37.0	24.7	20.4	38.9
	✓	25	62.6	52.2	38.8	24.7	21.9	40.0
$\pi_{0.5}$-Vela	N/A	25	66.1	53.5	40.7	27.3	22.5	42.0
Vela	N/A	25	**67.2**	**55.9**	**44.9**	**31.5**	**27.1**	**45.3**

Table 4: Effect of adaptive temporal support on LIBERO-X. We compare Vela with globally fixed temporal support (Fixed H.) against the full model with adaptive temporal support (Adap. H.).

Method	Level 1	Level 2	Level 3	Level 4	Level 5	Avg.
$\pi_{0.5}$	65.2	53.2	36.0	24.1	18.0	39.3
Vela w/ Fixed H.	66.4	54.3	42.8	28.1	24.4	43.2
Vela w/ Adap. H.	**67.2**	**55.9**	**44.9**	**31.5**	**27.1**	**45.3**

Table 5: Ablation of trajectory geometry objectives on LIBERO-X.

Objectives		Success Rate (%)					
CP-FM	DT-FM	Level 1	Level 2	Level 3	Level 4	Level 5	Avg.
✓		65.2	54.7	43.1	31.3	25.8	44.0
✓	✓	65.7	**56.0**	44.0	31.3	**27.6**	44.9
	✓	**67.2**	55.9	**44.9**	**31.5**	27.1	**45.3**

from a smoother trajectory. To test this, We compare two ways of introducing trajectory structure from the same $\pi_{0.5}$ base checkpoint. The post-hoc baseline retains action chunk prediction during post-training, then fits and resamples a spline at inference. In contrast, $\pi_{0.5}$-Vela learns to predict trajectory parameters during post-training. Alongside the official 10-step configuration, we retrain a $\pi_{0.5}$ baseline to support execution of up to 25 actions per prediction, the largest per-observation execution steps supported by the official evaluator. As shown in Tab. 3, post-hoc fitting alone does not recover the gains of Vela's modeling under all execution setting. This advantage is observed without our trajectory-space pretraining stage, supporting the value of learning trajectory parameters itself rather than imposing spline structure only after point-wise action prediction.

② **Does Motion-Dependent Horizon Adaptation Matter?** We next isolate the contribution of motion-dependent horizon adaptation from Vela's modeling. We compare a globally fixed horizon with motion-dependent adaptive horizon while retaining the same spline representation and control-point budget. As shown in Tab. 4, Vela's modeling provides gains even under a fixed temporal support, and horizon adaptation further raises performance at every level. This result supports allowing the same trajectory capacity to cover different temporal spans, rather than imposing a uniform temporal allocation on every predicted motion.

③ **Control Point vs. Decoded Trajectory Supervision.** CP-FM and DT-FM supervise the same control-point flow residual with different metrics: CP-FM uses Euclidean distance in parameter space, whereas DT-FM measures the residual after spline decoding. We compare CP-FM, an equal-weight combination of CP-FM and DT-FM, and DT-FM alone. All three objectives yield effective policies, with DT-FM achieving the highest average SR in Tab. 5. The modest differences suggest that effective trajectory-space learning is *not* contingent on a particular loss composition, decoder-aware weighting provides a useful refinement rather than a prerequisite.

5 Conclusion

We introduced Vela, a trajectory-space vision-language-action foundation model designed to address the limitations of fixed-rate action chunks. It predicts cubic B-spline control points together with a motion-dependent temporal span, using a shared action interface and scalable target construction to represent heterogeneous robot motion as continuous trajectories. Our experiments show that native trajectory-space learning improves action modeling, while post-hoc spline fitting is insufficient. Motion-dependent temporal support lets a fixed representation adapt its resolution to different motions, and decoder-aware supervision and large-scale heterogeneous pretraining further improve transfer across precision, mobile, and long-horizon tasks. Experiments on two real-world manipulation tasks further show that trajectory-space action modeling transfers beyond simulation to multi-stage bimanual manipulation involving tool use and sustained physical contact. Overall, these results support continuous trajectories as a promising action representation for future embodied foundation models.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for assisting with drafting and polishing the presentation of this paper, including improving grammar, clarity, and readability. We have not used generative AI tools for designing the experiments, cleaning and reformating dataset, and generating synthetic data sets, generating experimental results, developing theoretical or mathematical claims are not applicable to this work. We have reviewed all AI-assisted work. We revised for technical correctness, consistency with the underlying research, and appropriate attribution to prior work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

This work studies action representations and training methods for robotic manipulation. It does not involve human-subject studies or the collection of sensitive personal information. The simulation experiments are conducted on established robotics benchmarks, and real-robot experiments are performed in controlled research environments. The methods presented here are intended for research use, and deployment on physical systems should therefore incorporate appropriate hardware and operational safety mechanisms. We follow the applicable licenses and usage conditions of the datasets, models, and benchmarks used in this work.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

We provide the information required to reproduce the proposed method and experiments throughout the paper and supplementary material. The complete model formulation, trajectory parameterization, motion-dependent horizon construction, and training objective are described in Sec. 3. Experimental settings, model initialization, action-space preprocessing, spline configuration, and evaluation protocols are summarized in Sec. 4.1. Appendix. A.2.1 provides the full optimization hyperparameters, compute setup, and additional implementation details, while Appendix. A.2.2 specifies the benchmark-specific post-training and evaluation procedures. Our ablation studies further document the effects of the main methodological and deployment choices. These details are provided to enable independent reproduction of both the training procedure and the reported evaluations. Upon acceptance, we will release the training and evaluation code, checkpoints, and data interface.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026a.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026b.

Xiaowei Cai, Yunuo Cai, Bingao Chen, Jingxiao Chen, Zhi Chen, Siyuan Feng, Tengyu Hou, Jingshun Huang, Han Jiang, Runkun Ju, et al. tau_0-vla: a hierarchical robot foundation model with world-model-guided test-time computation. *arXiv preprint arXiv:2608.16885*, 2026c.

Carl De Boor. On uniform approximation by splines. *J. Approx. Theory*, 1(1):219–235, 1968.

Carl De Boor and Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

Ning Gao, Jinliang Zheng, Xing Gao, Haoxiang Ma, Hanqing Wang, Yukai Wang, Jiantong Chen, Zanxin Chen, Shujie Zhang, Mingda Jia, et al. Ebench: Elemental diagnosis of generalist mobile manipulation policies. *arXiv preprint arXiv:2606.18239*, 2026.

Xiaoshen Han, Haoyu Xiong, Haonan Chen, Chaoqi Liu, Antonio Torralba, Yuke Zhu, and Yilun Du. B-spline policy: Accelerating manipulation policies via b-spline action representations. *arXiv preprint arXiv:2607.09648*, 2026.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–373, 2013.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Yiming Li, Nael Darwiche, Amirreza Razmjoo, Sichao Liu, Yilun Du, Auke Ijspeert, and Sylvain Calinon. Geometry-aware policy imitation. In *International Conference on Learning Representations*, 2026.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, Chong Shi, Dengke Shang, Dingqian Zhang, Guan Huang, Guangqiang Wang, et al. Gigabrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture. *arXiv preprint arXiv:2608.15875*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Mengze Tian, Yiming Li, Sichao Liu, Auke Ijspeert, and Sylvain Calinon. Spline policy: A structured representation for robot policies. *arXiv preprint arXiv:2606.07386*, 2026.

Weikang Wan, Ziyu Wang, Yufei Wang, Zackory Erickson, and David Held. Difftori: Differentiable trajectory optimization for deep reinforcement and imitation learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Guodong Wang, Chenkai Zhang, Qingjie Liu, Jinjin Zhang, Jiancheng Cai, Junjie Liu, and Xinmin Liu. Libero-x: Robustness litmus for vision-language-action models. *arXiv preprint arXiv:2602.06556*, 2026a.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits: Adaptive execution horizons for robot policies. In *European Conference on Computer Vision*, pp. 607–623. Springer, 2026b.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026c.

Fan Yang, Peiguang Jing, Kaihua Qu, Ningyuan Zhao, and Yuting Su. Abpolicy: Asynchronous b-spline flow policy for real-time and smooth robotic manipulation. *arXiv preprint arXiv:2602.23901*, 2026.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. *Advances in Neural Information Processing Systems*, 38: 172934–172959, 2026.

A Appendix

A.1 Additional Methodology Details

A.1.1 Target Construction

For each sampled transition, the allocator selects the longest candidate horizon that satisfies both the continuous-coordinate and binary or near-deterministic channel (*e.g.,* event channel) fitting criteria. If no candidate satisfies both criteria, we return the shortest candidate and its fitted control points as a fallback. This fallback keeps target construction defined, but does not imply that the adequacy criteria are satisfied. The selected spline fit and horizon form the joint supervision target. Algorithm 1 makes this selection explicit.

Algorithm 1 Motion-Dependent Horizon Adaptation

Require: Candidate horizons $\mathcal{H} = \{H_1 > \cdots > H_K\}$, demonstrated action $\mathbf{A}_{H_1} \in \mathbb{R}^{H_1 \times d_a}$, noise scales $\{\hat{\sigma}_j\}_{j \in \mathcal{J}}$, tolerances κ and η
Ensure: Adaptive trajectory target $(\mathbf{C}^\star, H^\star)$

Globally compute fixed spline basis and pseudoinverse pairs $\{\mathbf{B}_H, \mathbf{B}_H^\dagger\}_{H \in \mathcal{H}}$
for $H \in \mathcal{H}$ **do** ▷ longest to shortest
  $\mathbf{A}_H \leftarrow \mathbf{A}_{H_1}[0{:}H, :]$
  $\mathbf{C}_H \leftarrow \mathbf{B}_H^\dagger \mathbf{A}_H$
  $\mathbf{R}_H \leftarrow \mathbf{A}_H - \mathbf{B}_H \mathbf{C}_H$
  $\psi^2(H) \leftarrow \dfrac{1}{|\mathcal{J}|(H-N)} \displaystyle\sum_{j \in \mathcal{J}} \dfrac{\|\mathbf{R}_H^{(:,j)}\|_2^2}{\hat{\sigma}_j^2}$ ▷ continuous coordinates
  $e_H \leftarrow \wedge_{j \notin \mathcal{J}} [\|\mathbf{R}_H^{(:,j)}\|_2 / \sqrt{H} \le \eta]$ ▷ event coordinates
  if $\psi(H) \le \kappa$ **and** e_H **then**
    return $(\mathbf{C}_H, H)$
  end if
end for
return $(\mathbf{C}_{H_K}, H_K)$ ▷ fallback to the shortest horizon

Computation. Although Algorithm 1 is written as a descending scan to make the target-selection rule explicit, all candidate horizons can be evaluated jointly in implementation. The spline basis matrices and their pseudoinverses depend only on the configured candidate horizons and the fixed spline configuration, and are therefore precomputed once and reused throughout training. In practice, we evaluate all candidates in a single batched pass and select the largest admissible horizon by a masked reduction.

A.1.2 Objective Details

Sec. 3.4 defines DT-FM by applying the spline decoder to the control point flow residual. Here we make explicit its relation to CP-FM. Since $\hat{A}_H = B_H C$, the control point flow error $\mathbf{E}_C$ induces the decoded action trajectory error $\mathbf{B}_{H^\star}\mathbf{E}_C$ at the target horizon $H^\star$. Thus, DT-FM can be viewed as applying flow supervision in the decoded action space rather than directly in the control point space,

$$\ell_{\mathrm{DT}} = \frac{\|\mathbf{B}_{H^\star}\mathbf{E}_C\|_F^2}{H^\star d_a} = \frac{1}{d_a} \operatorname{tr}\left[(\mathbf{E}_C)^\top \frac{\mathbf{B}_{H^\star}^\top \mathbf{B}_{H^\star}}{H^\star}(\mathbf{E}_C)\right]. \tag{13}$$

Thus, DT-FM differs from isotropic control point supervision only through the positive-semidefinite metric induced by spline decoding. For the spline configurations used in our experiments, $\mathbf{B}_{H^\star}$ has full column rank, such that $\mathbf{B}_{H^\star}^\top \mathbf{B}_{H^\star}$ is positive definite. Consequently,

$$\ell_{\mathrm{DT}} = 0 \quad \Longleftrightarrow \quad \mathbf{E}_C = \mathbf{0} \quad \Longleftrightarrow \quad \ell_{\mathrm{CP}} = 0.$$

This establishes that both losses detect the same nonzero control-point flow errors while weighting their directions differently. It does not, by itself, establish identical minimizers of the expected training objectives: the decoder metric depends on the target horizon, which may remain random conditional on the noisy flow input. Accordingly, our comparison treats DT-FM as a decoder-induced reweighting and evaluates its effect empirically.

A.2 Additional Implementation Details

A.2.1 Trajectory-Space Pretraining Details

This section describes the large-scale trajectory-space pretraining of Vela from pretrained PaliGemma VLM weights. $\pi_{0.5}$-Vela does not undergo this stage: it is adapted from the $\pi_{0.5}$ base checkpoint using the benchmark-specific post-training procedures below.

Vela is optimized with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}) and global gradient clipping at 1.0. The learning rate is linearly warmed up to 5×10^{-5} over the first 10k optimization steps and kept constant thereafter. We apply a $0.5\times$ learning-rate multiplier to the PaliGemma backbone, corresponding to 2.5×10^{-5}, while the action expert and output projection use the full learning rate. Training proceeds for 300k optimization steps.

Normalization statistics are computed independently for each data source and action dimension using a quantile transform that maps the 1st and 99th percentiles to $[-1, 1]$, while unavailable action dimensions are excluded from supervision through validity masks.

For adaptive target construction, only the two gripper coordinates are treated as event channels, all remaining valid action coordinates use the noise-calibrated adequacy criterion. We set the global adequacy threshold to $\kappa = 2.5$ and the event-channel reconstruction tolerance to $\eta = 0.05$.

Training is distributed over 64 NVIDIA H200 GPUs using FSDP with gradient checkpointing, with 24 samples per GPU, yielding a global batch size of 1,536. Transformer computation is performed under bfloat16 autocast, whereas B-spline target construction, flow interpolation, and loss computation are carried out in fp32 for numerical stability.

A.2.2 Benchmark Fine-tuning Details And Evaluation

Initialization and post-training. For each benchmark, $\pi_{0.5}$, $\pi_{0.5}$-Vela, and Vela use the same training split and optimization hyperparameters. The $\pi_{0.5}$ baseline and $\pi_{0.5}$-Vela start from the same $\pi_{0.5}$ base checkpoint and are post-trained with action chunk and trajectory-space supervision, respectively. Vela starts from the trajectory-space pretrained checkpoint described in Appendix A.2. The optimization settings and execution configurations are specified separately below.

LIBERO-X. LIBERO-X (Wang et al., 2026a) provides 2,520 human-teleoperated training demonstrations spanning 600 tasks and 100 scenes. The demonstrations are collected at 20 Hz, with third-person and wrist-camera observations and Cartesian end-effector actions together with gripper commands. Following the official $\pi_{0.5}$ configuration, visual inputs are resized to 224×224.

For benchmark-specific fine-tuning, we follow the official $\pi_{0.5}$ full-parameter fine-tuning recipe. Fine-tuning is performed for 30k optimization steps with a global batch size of 256. We use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}) with global gradient clipping at 1.0. The learning rate is linearly warmed up to 5×10^{-5} over the first 10k steps and kept constant thereafter, and an exponential moving average with decay 0.999 is maintained throughout fine-tuning.

Evaluation follows the official five-level protocol. Level 1 introduces local spatial perturbations, Level 2 expands the spatial variation, Level 3 reconstructs scene topology, Level 4 further varies visual attributes and introduces unseen or confounding objects, and Level 5 additionally reformulates instructions while preserving their task semantics. The perturbations are cumulative, such that each successive level introduces additional sources of distribution shift. Each task is evaluated over 10 rollouts with the benchmark-defined initial-state configuration and rollout perturbations. Following the official setting, the task-specific time limit is set to $1.1\times$ the average human completion time. We use the benchmark implementation without modifying environment initialization, perturbation ranges, or success conditions. The LIBERO-X evaluator executes at most 25 actions from each prediction. For a predicted action sequence containing h_t actions, the executed length is $\min(h_t, 25)$.

The official $\pi_{0.5}$ benchmark configuration uses 10-step execution. For the 25-step post-hoc comparison in Tab. 3, we retrain a separate $\pi_{0.5}$ baseline from the same base checkpoint with its horizon configuration adjusted to support execution of up to 25 actions, while retaining the other optimization hyperparameters. Spline fitting is then applied to this model's predicted action chunks at inference.

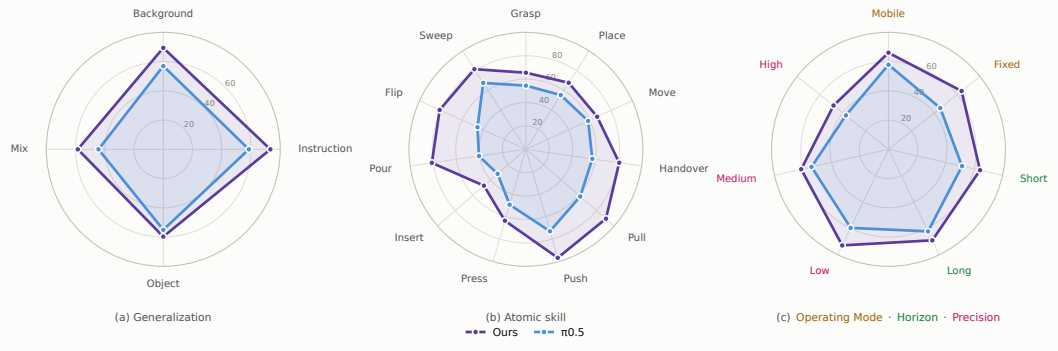


Figure 4: Detailed capability profile of Vela and $\pi_{0.5}$ on EBench. We report the official task progress Score (%) across (a) the four generalization conditions, (b) atomic skills, and (c) operating mode, temporal horizon, and precision. Vela consistently improves over $\pi_{0.5}$ across the displayed categories

EBench. We follow the official $\pi_{0.5}$ fine-tuning recipe released with EBench (Gao et al., 2026). The benchmark training set contains 6,600 demonstrations (91.4 hours) across all 26 tasks, including 2,800 human-teleoperated trajectories for the 7 tabletop dexterous-and-precise tasks and 3,800 motion-planned trajectories for the 10 mobile pick-and-place and 9 mobile long-horizon tasks.

All parameters are fine-tuned from the corresponding initialization for 200k optimization steps with a global batch size of 128. We use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}) with global gradient clipping at 1.0. The learning rate is linearly warmed up for 10k steps to 5×10^{-5} and subsequently cosine-decayed to 5×10^{-6} over the full 200k-step schedule. An exponential moving average with decay 0.999 is maintained throughout fine-tuning.

The Test set contains 510 episodes spanning the four generalization dimensions. The EBench evaluator executes at most 30 actions from each prediction. For a predicted action sequence containing h_t actions, the executed length is $\min(h_t, 30)$.

A.3 Further Analysis on Experiments

Additional results on EBench. The main text summarizes EBench performance by its three manipulation task families. Here, we provide a finer-grained view by exploiting the benchmark's task-level and capability annotations. Table 6 reports task-level success rates, while Figures 4 and 5 reorganize the evaluation along operating mode, temporal horizon, precision, generalization, and atomic skill. These views address complementary questions: task-level SR measures complete task success, whereas the grouped Score profiles describe progress across broader evaluation regimes.

Figure 4 further isolates the comparison with $\pi_{0.5}$. Across all three profiles, Vela consistently exceeds $\pi_{0.5}$: the improvement holds across all four generalization conditions, across the atomic skill categories, and across operating mode, horizon, and precision. The resulting profile indicates that the overall gain is broad-based rather than concentrated in a particular task family or capability category.

Figure 5 places this breadth in the context of other policies: Vela performs strongly across operating modes, task horizons, and generalization conditions, without leading every precision category. Together, the task-level and grouped results support the use of one trajectory representation across varied manipulation demands, rather than an advantage confined to a small task subset.

Robustness Across Execution Caps. Temporal support is learned as part of Vela's action representation, whereas the execution cap limits how many decoded actions are consumed before replanning. To explore whether the advantage of Vela depends on a particular execution schedule, we evaluate both $\pi_{0.5}$ and Vela with caps of 10, 15, 20, and 25 actions. For Vela, a cap r gives an executed length of $\min(h_t, r)$ without fixing the predicted trajectory span. As shown in Fig. 6, Vela consistently outperforms $\pi_{0.5}$ across the entire range of execution caps and exhibits less variation with this

Table 6: Task-level results on EBench. Success rate (SR, %) for each of the 26 tasks, grouped into dexterous-and-precise tabletop, mobile pick-and-place (PnP), and mobile long-horizon tasks. The final row reports benchmark-wide performance.

Task	**Ours**	Qwen-RobotManip	$\pi_{0.5}$	InternVLA-A1.5	π_0	GigaBrain-0.7	Cosmos3-Edge	FastWAM	X-VLA
TableTop									
collect coffee beans	15	5	30	0	20	**50**	0	0	0
flip cup collect cookies	60	**90**	25	30	30	40	30	0	5
frame against pen holder	65	**95**	45	0	25	70	20	0	30
install gear	25	**50**	10	15	30	20	15	15	5
peg in hole	15	10	10	0	15	15	**30**	0	5
put glass in glassbox	**90**	**90**	65	0	85	55	25	5	15
tighten nut	**40**	10	20	**40**	25	15	10	0	0
PnP									
apple from shelf	45	65	**75**	35	40	55	15	40	70
apple to fruit bowl	50	50	25	20	10	20	5	**70**	50
bookmark on book	**55**	40	**55**	45	25	30	25	30	40
bowl to plate	80	75	**85**	60	60	65	65	75	80
perfume to cosmetics rack	**80**	55	60	45	25	25	60	50	35
remote to holder	**80**	65	60	60	70	50	75	50	35
salt to spice rack	**100**	70	70	90	80	75	80	80	65
soap to dish	85	85	**100**	95	95	70	85	90	70
teacup to saucer teapot to tray	**50**	45	30	30	10	20	0	10	10
utensils to holder	5	15	5	20	0	10	5	0	**45**
LongHorizon									
bottle	0	0	0	0	0	0	0	0	0
detergent	90	85	**100**	95	75	95	95	35	0
dish	5	20	**25**	20	0	5	0	0	0
dishwasher	**90**	15	65	**90**	**90**	0	15	70	50
fruit	20	**45**	10	5	0	0	0	0	0
make sandwich	73	67	67	33	20	**80**	27	47	7
microwave	53	33	40	60	47	0	**80**	0	0
pen	0	**5**	0	0	0	0	0	0	0
shop	**20**	0	0	0	0	0	0	0	0
Overall	**49.7**	45.6	41.4	34.2	33.7	33.3	29.3	25.6	23.7

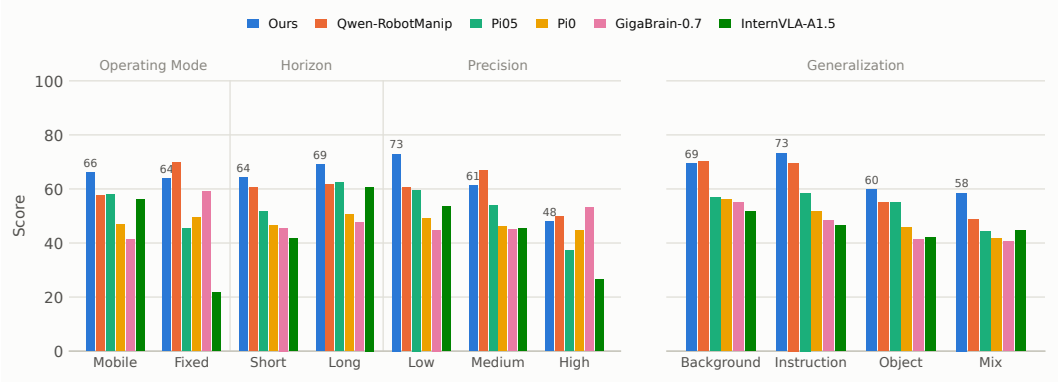


Figure 5: Fine-grained capability breakdown on EBench. We compare Vela with representative baselines using the official task progress Score (%) across operating mode (Mobile / Fixed-base), temporal horizon (Short / Long), precision (Low / Medium / High), and four generalization conditions (Background / Instruction / Object / Mix).

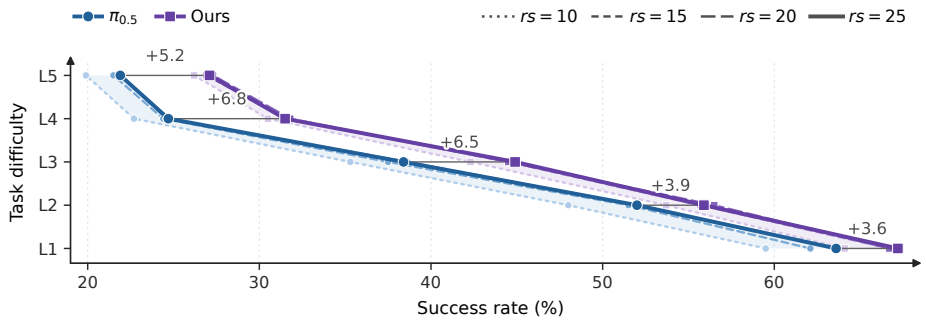


Figure 6: Robustness to replanning interval on LIBERO-X. We vary the maximum number of executed actions before replanning for both $\pi_{0.5}$ and Vela.

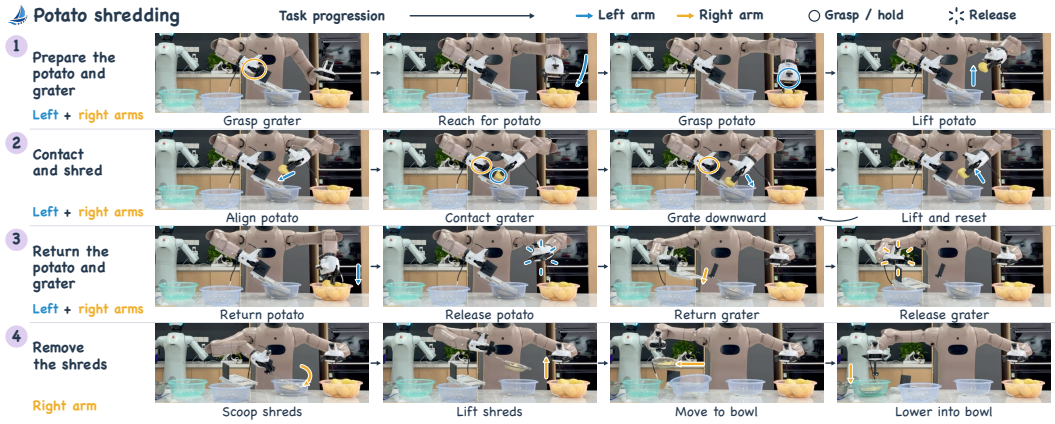


Figure 7: Vela executing *shredding potato*, which spans four subtasks (top to bottom).

deployment choice. These results show that the benefit of Vela's modeling is not tied to a narrowly tuned execution schedule, but persists across different execution cap settings.

A.4 Additional Real-World Evaluation Details

Robot platform and evaluation protocol. All real-world experiments are conducted on the same wheeled dual-arm robot introduced in Sec. 4.1. The platform has two 7-DoF arms and three RGB cameras, with one camera mounted on the head and one on each wrist. We evaluate Vela in two manipulation settings: the four-stage egg-cake cooking task reported in the main paper and an additional four-stage potato-shredding task.

Real-World Evaluation on Potato Shredding. We additionally evaluate Vela on the *shredding potato* task using the same wheeled dual-arm robot. The task requires coordinated bimanual tool manipulation and repeated physical contact between the potato and grater. The task is decomposed into four subtasks as follows:

1. **Prepare tools.** Pick up a potato with the left gripper and the grater with the right gripper.
2. **Contact and shred.** Adjust the potato against the grater, then repeatedly grate it so the shreds fall into the water.
3. **Return tools.** Return the potato and grater to their designated locations.
4. **Remove shreds.** Use the strainer to lift the potato shreds out of the water and transfer them into the bowl.

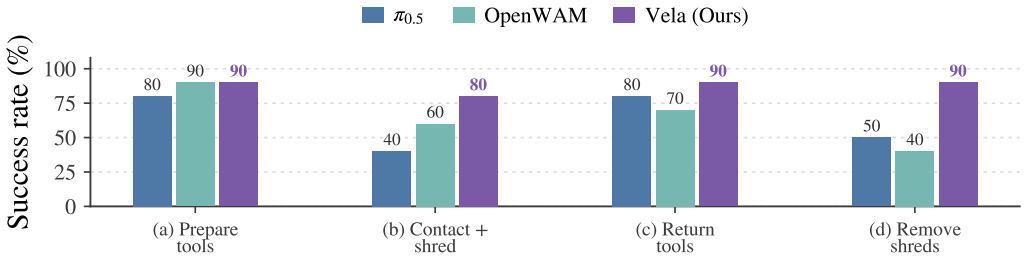


Figure 8: Additional real-world potato-shredding evaluation. Each bar reports the success rate over ten trials for one of the four ordered subtasks.

For $\pi_{0.5}$ and OpenWAM, we report the corresponding results under the same four-subtask protocol. Vela achieves a mean subtask success rate of 87.5%, compared with 62.5% for $\pi_{0.5}$ and 65.0% for OpenWAM, while matching or exceeding both baselines on every stage. The gains are most evident in stages involving sustained physical contact and coordinated tool manipulation, providing complementary real-world evidence to the egg-cake experiment in the main paper.

A.5 Limitations

Our study leaves three scaling directions open. First, Vela currently uses a model at the same scale as $\pi_{0.5}$. Experiments with larger vision-language backbones and more extensive pretraining are needed to determine whether the gains from trajectory-space action modeling continue to scale with model and data size. Second, we retain the action-expert capacity of the base model. A larger or more specialized action expert may provide additional capacity for fine contact dynamics and long-horizon bimanual coordination, but this setting has not been evaluated. Third, the policy predicts a fixed number of spline control points for every motion. Although the temporal horizon adapts to the motion, the control-point budget does not; allowing the number of control points to vary with motion complexity could improve representation efficiency, but requires a corresponding variable-length prediction and training design.