WAVE: LEARNING TO VERIFY ACTIONS IN 3D WORLD SPACE FOR GENERATIVE ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative robot policies, including vision-language-action models (VLAs) and world-action models (WAMs), produce temporally extended action chunks from observations and instructions. At deployment, native sampling exposes multiple candidate chunks, yet standard control commits to one without comparing how the candidates would change the scene. We introduce **WAVE** (**W**orld-space **A**ction **VE**rification), an overall policy enhancement for generative robot policies. WAVE reconstructs the current scene once as a dense Gaussian field, compresses it into a compact representation, and predicts a separate compact future rollout for every candidate chunk. A unified score fuses the policy's native consistency score with learned verifier scores and structured verifier adjustments in the shared world-space frame. Across three simulation benchmarks and nine matched frozen-policy comparisons, WAVE reaches 99.0% on LIBERO, 79.5% on RoboCasa, and 95.6% average success on RoboTwin 2.0. These results show that consequence comparison converts native candidate diversity into reliable manipulation decisions.

1 INTRODUCTION

Generative robot policies now span vision-language-action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Physical Intelligence, 2025) and world-action models (WAMs) (Yuan et al., 2026; Bi et al., 2025; Li et al., 2026a), which map observations and task instructions to predicted action chunks, each a short sequence of controls. For policies with stochastic or iterative samplers, repeated queries under the same policy input can produce different chunks, although standard deployment typically executes one sample. The native sampler supplies a signal of how closely each candidate follows the policy's own generation process, but does not explicitly evaluate the outcome each action would produce. Different predictions can therefore lead to different task outcomes even when they are similarly favored by the policy. *The policy encodes behavioral plausibility, while predicted consequences indicate which action better advances the task.* This motivates our central question: how can we represent and compare the consequences of actions generated by a policy before execution? Figure 1 contrasts direct native deployment with consequence-aware action selection.

One way to answer this question is to predict task evolution under alternative actions. Action-conditioned world models use predicted videos, latent trajectories, or future values to guide selection (Wang et al., 2025; Kim et al., 2026; Zhou et al., 2026). Complementarily, other robot policies and predictive models make future geometry explicit in 3D, using metric point clouds, object-centric geometry, or Gaussian fields as policy inputs or predicted future states (Zhen et al., 2024; Li et al., 2026b; Zhang et al., 2026a; Lu et al., 2024; 2025; Huang et al., 2026; Zhang et al., 2026c). Comparing action chunks requires the same evaluation criteria for every candidate, such as goal progress, spatial relations, contact, and physical plausibility. For instance, the distance between a gripper and its target and their contact state should be measured in the same metric coordinates for every candidate. Videos and latent futures can encode these changes, but they typically do not directly expose the same measurements in a shared calibrated metric frame. An explicit 3D state places each candidate's predicted geometry in one shared calibrated world-space frame, so the same measurements can be reused across candidates; learned verifier scores provide complementary task-semantic evidence beyond geometry. *Shared 3D state makes predicted consequences directly comparable.*

To keep this comparison affordable as the number of candidate chunks grows, we introduce **WAVE** (**W**orld-space **A**ction **VE**rification), an overall policy enhancement that reranks action chunks already

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.

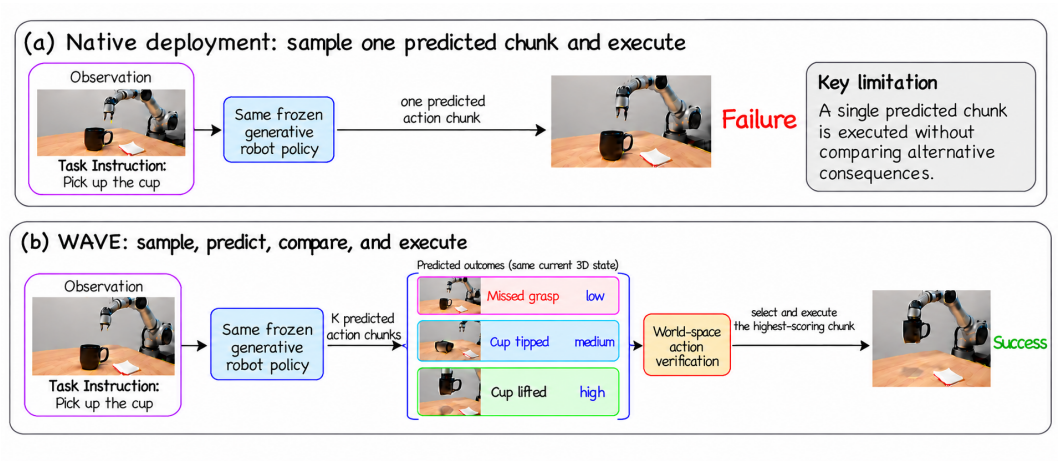

Figure 1: (a) Native deployment commits to one predicted action chunk and executes it without comparing alternatives. A chunk favored by the policy can therefore produce the wrong consequence, as illustrated by the missed grasp. (b) WAVE samples K predicted chunks, predicts their consequences from the same 3D state, and compares them with world-space action verification. This exposes the choice before execution: select the highest-scoring consequence, here the "Cup lifted" candidate.

proposed by a generative robot policy before execution. WAVE strengthens policy selection while leaving native action generation unchanged. It reconstructs the current state once and branches only compact, action-dependent futures for each candidate, so every future is scored in the same calibrated world-space frame. Sharing the reconstructed state removes repeated scene reconstruction and most shared scene-context computation from candidate comparison.

WAVE implements this design with a dense-to-compact Gaussian representation. It reconstructs one calibrated 3D state from the observation and policy context, compresses that state into a fixed set of Gaussian slots, and predicts a separate compact future rollout for each policy-generated candidate chunk. Each rollout supplies the predicted consequence used for comparison. A unified verifier combines learned verifier scores with structured verifier adjustments, then fuses this comparative evidence with a sampler-trajectory native consistency signal that preserves the policy's own action-generation preference. The policy remains frozen, and WAVE ranks an existing action chunk rather than synthesizing a new one. *WAVE adds consequence evidence to policy selection without changing the policy that generated the actions.*

Across nine frozen-policy comparisons on LIBERO, RoboCasa, and RoboTwin 2.0, WAVE configurations reach 99.0%, 79.5%, and 95.6% success, and all nine policy–WAVE pairings improve over native deployment. Selection-rule and verifier-channel ablations support complementary roles for native consistency, learned verifier scores, and structured verifier adjustments. Representation and future-prediction ablations support the 3D future representation and action-conditioned prediction, while candidate-budget results identify $K = 4$ as a strong operating point on LIBERO-Long and RoboCasa, where normalized inference time remains below $2\times$. *The results indicate gains from consequence-aware selection among the policy's native predictions.* Our contributions are:

- We introduce WAVE, an overall policy enhancement that selects among a generative policy's predicted action chunks by comparing their consequences in a shared calibrated world-space state, while leaving the policy's native action generator unchanged.
- We develop an efficient dense-to-compact Gaussian consequence model that reconstructs the current state once, branches compact action-conditioned futures for each candidate, and unifies sampler-trajectory native consistency with learned verifier scores and structured 3D adjustments for consequence-aware selection.
- We show consistent gains across nine matched policy–benchmark pairings on LIBERO, RoboCasa, and RoboTwin 2.0, demonstrating that consequence-aware selection turns native candidate diversity into more reliable manipulation decisions.

2 Related Work

Generative robot policies and candidate generation. Robot policies combine visual-language grounding with temporally extended action generation. ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2023) established chunked behavior generation; RT-1 (Brohan et al., 2022) and RT-2 (Brohan et al., 2023) demonstrated generalist vision-language-action control; and Octo (Octo Model Team, 2024), OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), and $\pi_{0.5}$ (Physical Intelligence, 2025) extended it across tasks and embodiments. World-action systems such as Fast-WAM (Yuan et al., 2026), LingBot-VA (Li et al., 2026a), and Cosmos Policy (Kim et al., 2026) couple action generation with learned scene evolution. Stochastic or iterative samplers can be queried repeatedly under one input to obtain alternatives, and explicit iterative paths expose native generation traces. WAVE keeps the native generator fixed and adds world-space comparison over the candidates.

Predictive selection and action verification. Model-based control uses imagined futures and values for decision making, from PlaNet (Hafner et al., 2018) and Dreamer (Hafner et al., 2019) to systems that search latent or predictive spaces. Latent Policy Steering (Wang et al., 2025), Generative Predictive Control (Qi et al., 2026), VLA-Reasoner (Guo et al., 2025), and Cosmos Policy (Kim et al., 2026) reason over predicted outcomes, latent rollouts, or future values; ω-EVA (Sun et al., 2026), SC3-Eval (Tseng et al., 2026), Masked Visual Actions (Alzayer et al., 2026), and τ_0-WM (Zhou et al., 2026) use imagined futures for consistency, verification, or refinement. Other verifiers assess semantic or visual–action compatibility, as in CoVer-VLA (Kwok et al., 2026), while World Action Verifier (Liu et al., 2026) and Harness VLA (Zhang et al., 2026b) use forward–inverse or execution-level feedback. Prior methods select through latent, video, value, semantic, or execution-level evidence. WAVE instead rolls every candidate from one calibrated world-space state, reuses identical geometric measurements, and combines them with learned verifier scores and native policy consistency.

3D representations and predictive dynamics for manipulation. 3D geometry serves as a policy input, supervision target, or predictive state. DP3 (Ze et al., 2024), 3D-VLA (Zhen et al., 2024), 3DVLA (Xia et al., 2026), ConsisVLA-4D (Li et al., 2026b), and EvoScene-VLA (Zhang et al., 2026a) inject metric, instance-level, or evolving geometry into action prediction. Differentiable 3D Gaussian fields (Kerbl et al., 2023) represent or predict manipulation scenes and dynamics in ManiGaussian (Lu et al., 2024), GWM (Lu et al., 2025), GaussianDream (Zhang et al., 2026c), and GaussianWAM (Zhang et al., 2026d); PointWorld (Huang et al., 2026) and LAWM-3D (Yang et al., 2026) use point-based or 3D-aware latent-action representations. WAVE places compact Gaussian futures in the selection loop: it reconstructs the dense scene once, branches candidate-conditioned futures, and applies identical spatial operators to rank the resulting chunks.

3 Method

3.1 Decision Interface: Predict, Verify, and Select

WAVE is an overall policy enhancement for generative robot policies, including VLA interfaces and WAM architectures. For the evaluated policies, the native interface exposes observation features and the sampler state used by its consistency signal. WAVE reuses these representations to construct a shared 3D scene and compares predicted consequences of candidate action chunks. The policy remains frozen: WAVE ranks its candidate bank, adds no simulator at decision time, and executes only a selected chunk prefix. The policy supplies representations and candidate actions; WAVE turns them into geometric predictions and comparative ranking evidence.

Let o_t denote the observation context, including calibrated current views, with instruction l and robot state q_t. Retaining its native input interface and preprocessing, the base policy Π_ϕ samples K action chunks with independent noise initializations to form the candidate bank $\mathcal{B}_t$:

$$\mathbf{A}_t^{(k)} \sim \Pi_\phi(\cdot \mid o_t, l, q_t), \qquad \mathcal{B}_t = \big(\mathbf{A}_t^{(1)}, \ldots, \mathbf{A}_t^{(K)}\big). \tag{1}$$

Here ϕ collects the base-policy parameters, and each $\mathbf{A}_t^{(k)} \in \mathbb{R}^{H_a \times d_a}$ contains H_a controls.

To construct a shared scene for evaluating these candidates, WAVE reuses the base policy's internal representation of the observed scene and task. We denote these backbone features by c_t and extract them once, independently of candidate actions and their sampling noise. Any auxiliary inputs and

diffusion-time settings used for this extraction are held fixed. The scene constructor S_η, comprising a feature adapter, a dense scene decoder, and a compact pooler, combines c_t with the calibrated observations. The candidate-conditioned compact dynamics model W_θ then predicts a separate compact future rollout for each candidate:

$$\mathcal{G}_t = S_\eta(o_t; c_t), \qquad \widehat{\mathcal{G}}^{(k)}_{t+\Delta_1:t+\Delta_{H_w}} = W_\theta\Big(\mathcal{G}_t, \mathbf{A}^{(k)}_t, l\Big), \qquad k = 1, \dots, K. \tag{2}$$

Here $0 < \Delta_1 < \cdots < \Delta_{H_w}$ index the future time points represented by the compact rollout. They are prediction offsets rather than a second action chunk: the compact dynamics model encodes the complete H_a-step candidate chunk and resamples its temporal features to H_w future time points. The configured mapping between action controls and world time points is given in Appendix B. Every candidate rollout starts from the same compact scene $\mathcal{G}_t$ and uses the same calibrated world-space frame and Gaussian-slot representation, so the verifier compares how different actions change one scene rather than comparing separately reconstructed scenes.

For each candidate bank, WAVE combines the policy-native consistency score with comparative verifier evidence. The native consistency score $e^{(k)}_\Pi$ is read from the policy's sampler trajectory (Appendix B); flow-matching and rectified-flow samplers use a native flow residual, whereas Cosmos Policy uses an action-latent denoising residual. The verifier score is $s^{(k)}_{\text{ver}} = s^{(k)}_{\text{learn}} + s^{(k)}_{\text{struct}}$, with its two evidence sources defined in Section 3.3. Each score is standardized within the current bank, giving $\tilde{e}^{(k)}_\Pi$ and $\tilde{s}^{(k)}_{\text{ver}}$ (Appendix B.7). This puts the terms on a comparable scale, while $\beta \geq 0$ controls the influence of predicted consequences. WAVE selects

$$k^* = \underset{k \in \{1,\dots,K\}}{\arg\max} \left[\tilde{e}^{(k)}_\Pi + \beta \tilde{s}^{(k)}_{\text{ver}}\right]. \tag{3}$$

The native-score selector uses $\tilde{e}^{(k)}_\Pi$, whereas the fused WAVE selector uses Equation 3; both return an unchanged member of $\mathcal{B}_t$. A positive β lets predicted consequences change the policy's native ranking without synthesizing an action outside the bank. After selecting k^*, WAVE executes the first $H_e \leq H_a$ controls from $\mathbf{A}^{(k^*)}_t$, re-observes the scene, and repeats at the next decision.

3.2 Shared 3D State and Candidate-Conditioned Consequences

A learned feature adapter maps c_t into the scene decoder's feature space. The decoder combines this policy context with visual features from o_t to predict a dense Gaussian field, using camera calibration to express its geometry in the calibrated world-space frame. This couples the base policy's learned representations with observation-grounded geometry; backbone-specific feature adaptation is contained within S_η. Figure 2 summarizes this shared-state construction and the candidate-conditioned rollout. Appendix B.4 details the $\pi_{0.5}$ configuration.

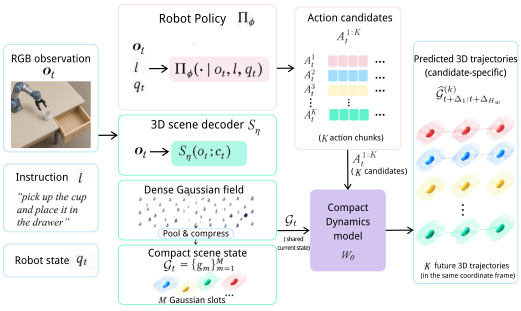

Figure 2: **Shared-state candidate rollout.** The policy produces K action chunks and context c_t. S_η combines c_t with the observation to reconstruct and pool a compact Gaussian state $\mathcal{G}_t$ for all candidates. W_θ rolls chunks independently, yielding K candidate-specific 3D trajectories in a calibrated world-space frame.

The resulting field is $\mathcal{G}^{\text{dense}}_t = \{\mathbf{g}_{i,t}\}_{i=1}^N$. Each primitive $\mathbf{g}_{i,t} = (\boldsymbol{\mu}_{i,t}, \boldsymbol{\sigma}_{i,t}, \alpha_{i,t}, \mathbf{r}_{i,t}, \mathbf{h}_{i,t})$ stores position, extent, opacity, orientation, and appearance. These attributes support both candidate-conditioned prediction and geometry-based verification. We construct the dense field once and compress it before branching over candidates, so every rollout starts from the same scene state.

Let M be the number of compact slots. To pool the dense field, WAVE maps each primitive's geometric and appearance attributes to a descriptor $\mathbf{k}_{i,t}$, while each slot m has a learned query $\mathbf{q}_m$. Their normalized dot-product affinity determines how much primitive i contributes to slot m, and the opacity $\alpha_{i,t}$ acts as a soft reliability prior. Writing d_k for the descriptor width and using

unit-normalized queries and keys, the pooling weights and compact primitives are

$$a_{mi,t} = \mathrm{softmax}_i\left(\frac{\langle \mathbf{q}_m, \mathbf{k}_{i,t}\rangle}{\sqrt{d_k}} + \log\max(\alpha_{i,t}, \varepsilon)\right), \qquad \tilde{\mathbf{g}}_{m,t} = \mathrm{Proj}\left(\sum_{i=1}^{N} a_{mi,t}\mathbf{g}_{i,t}\right). \tag{4}$$

Here $\varepsilon > 0$ keeps the log-opacity finite, $a_{mi,t}$ is normalized over primitives, and Proj restores valid field ranges and unit quaternion norm (Appendix B.4). The pooled slots form the compact scene state $\mathcal{G}_t = \{\tilde{\mathbf{g}}_{m,t}\}_{m=1}^{M}$. The same slot weights aggregate all fields into one geometric and visual summary; pooling approximates the dense field rather than preserving its exact Gaussian mixture. The fixed-size state is a compact scene representation rather than a persistent object tracker: it preserves the fields needed by the verifier while removing the dense cost from every candidate branch.

For candidate k, WAVE applies a shared compact transition operator to the pooled state and the full candidate action chunk, using the instruction, a global scene summary, and temporally resampled action features as context. Let $\mathbf{c}_{k,h}$ denote this fused condition at future time point Δ_h; each candidate is rolled forward as an independent branch:

$$\Delta\mathbf{g}^{(k)}_{m,h} = \mathbf{b} \odot \tanh f_\theta(\tilde{\mathbf{g}}_{m,t}, \mathbf{c}_{k,h}), \qquad \widehat{\mathbf{g}}^{(k)}_{m,t+\Delta_h} = \mathcal{U}\left(\tilde{\mathbf{g}}_{m,t}, \sum_{j=1}^{h} \Delta\mathbf{g}^{(k)}_{m,j}\right). \tag{5}$$

Here f_θ is shared across candidates and time points. The residuals are conditioned on the initial scene, not recursively on predicted states; accumulation produces each branch trajectory. It predicts a raw field residual from the pooled slot and fused candidate condition; $\tanh$ bounds that residual and $\mathbf{b}$ sets its field-wise scale. The residuals accumulate from the current primitive, and $\mathcal{U}$ applies the corresponding field-specific parameterization, including valid-range constraints and rotation normalization, to produce $\widehat{\mathbf{g}}^{(k)}_{m,t+\Delta_h}$. Dense decoding and pooling are shared, so only the M-slot dynamics are replicated over the K candidates.

3.3 World-Space Action Verification

The verifier compares each predicted outcome with the task and observed scene. For candidate k, it receives the current state $\mathcal{G}_t$, the predicted future $\widehat{\mathcal{G}}^{(k)}_{t+\Delta_1:t+\Delta_{H_w}}$, the action chunk $\mathbf{A}^{(k)}_t$, and instruction l. We decompose this judgment into four channels, $\mathcal{J} = g, s, c, p$: goal measures instruction-relevant progress; spatial measures consistency with scene geometry and object relations; contact measures interaction with object extents; and plausibility captures smoothness, bounds, and collision-related changes. For channel j, let $\ell^{(k)}_j$ be its learned score and $\delta^{(k)}_j$ its structured scene/action adjustment. Each channel contributes to both branches; channel names do not partition it by branch. With nonnegative mixture weights ω_j normalized by a softmax, we define

$$s^{(k)}_{\mathrm{learn}} = \sum_{j\in\mathcal{J}} \omega_j \ell^{(k)}_j, \qquad s^{(k)}_{\mathrm{struct}} = \sum_{j\in\mathcal{J}} \omega_j \delta^{(k)}_j, \qquad s^{(k)}_{\mathrm{ver}} = s^{(k)}_{\mathrm{learn}} + s^{(k)}_{\mathrm{struct}}. \tag{6}$$

The learned verifier produces channel scores; structured adjustments add geometry-, contact-, and constraint-based terms from the predicted Gaussian states and action, reusing semantic cues when geometric tests require grounding. The two components complement one another rather than forming a strict partition. A positive adjustment supports a candidate on that channel; a negative adjustment penalizes inconsistency. Whether used alone or fused, the active score is standardized within the current candidate bank before ranking. Scene adjustments use shared coordinates from $\mathcal{G}_t$. Action–scene and contact terms construct an embodiment-dependent action-path proxy rather than a physical end-effector trajectory (Appendix B). Representative cues include action-path/scene-displacement consistency, Gaussian-overlap changes indicating collisions, language-grounded goal progress, contact proximity, and transition smoothness. These cues instantiate the $\delta^{(k)}_j$ terms in Equation 6; they operate on compact Gaussian states, providing comparative ranking evidence without a decision-time simulator. Detailed operators appear in Appendix B.6.

The resulting $s^{(k)}_{\mathrm{ver}}$ summarizes how well candidate k satisfies the instruction while remaining consistent with the current scene and the scene evolution implied by its action. Its standardized value enters Equation 3 alongside native action-generation consistency. A channel ablation masks its corresponding ω_j before normalization, removing both verifier components of that channel. At fixed slot count and rollout horizon, dense encoding and pooling are shared and the additional rollout and verification computation grows linearly with K.

Decision-time cost. At each observation, WAVE extracts context and reconstructs the current scene once. Native action generation reuses observation features and attention caches where the backbone permits. Let C_{shared} include context extraction, feature adaptation, dense decoding, and pooling, including any additional backbone pass required for scene context. Let C_{act} denote the remaining cost of generating one candidate and evaluating its native consistency score, and let C_{roll} and C_{ver} denote its compact rollout and verification costs. Candidate-dependent backbone computation remains in C_{act}. The resulting compute and rollout memory are

$$C_{\text{WAVE}} = C_{\text{shared}} + K(C_{\text{act}} + C_{\text{roll}} + C_{\text{ver}}), \qquad \text{Mem}_{\text{rollout}} = \mathcal{O}(BKH_wMC_f). \tag{7}$$

Here B is the batch size and C_f is the number of channels in one compact slot. This expression counts work, not wall-clock latency: batching and device utilization affect measured time. Pairwise geometric costs can scale as M^2 within C_{ver}, while M is fixed in the candidate-count comparison.

3.4 Training Objectives and Schedule

We train WAVE as an overall policy enhancement around a frozen base policy. Each example is an expert demonstration segment with a current observation, instruction, robot state, and demonstration action chunk. Future RGB, depth, and motion observations supervise consequence prediction; the same segment also yields one positive training action and perturbed alternatives for relative verifier learning. These future observations and annotations are training targets only and are never supplied to the decision-time selector.

Training-only dense future decoding supplies the compact rollout targets; its conditioning path and supervision are specified in Appendix C. Future observations enter only this training path.

For flow and rectified-flow policies, an expert chunk $\mathbf{A}_t^{\text{E}}$ is paired with noise $\boldsymbol{\epsilon}$ and a flow time $\tau \in [0, 1]$. The resulting interpolated action $\mathbf{x}_\tau$ and target flow $\mathbf{u}$ define the native parameterization used to read the policy's consistency trace:

$$\mathbf{x}_\tau = \tau\boldsymbol{\epsilon} + (1-\tau)\mathbf{A}_t^{\text{E}}, \qquad \mathbf{u} = \boldsymbol{\epsilon} - \mathbf{A}_t^{\text{E}}. \tag{8}$$

The native vector field v_ϕ predicts $\mathbf{u}$ from $(\mathbf{x}_\tau, \tau, c_t)$; WAVE reads this trace but never optimizes the base policy. The same demonstration supplies dense future targets for the compact rollout. We use $\widehat{\mathcal{G}}^{\text{E}}$ to denote the rollout of W_θ with the demonstration action substituted for $\mathbf{A}_t^{(k)}$; it is matched to the pooled future target, with gradients stopped through the pooled target:

$$\mathcal{L}_{\text{world}} = \mathcal{L}_{\text{dense}} + \lambda_{\text{compact}} d_{\mathcal{G}}\left(\widehat{\mathcal{G}}^{\text{E}}_{t+\Delta_1:t+\Delta_{H_w}}, \text{sg}\left[\text{Pool}(\mathcal{G}^{\text{dense}}_{t+\Delta_1:t+\Delta_{H_w}})\right]\right). \tag{9}$$

Here the target fields are dense decoder predictions supervised by future observations, not ground-truth Gaussian states. $\mathcal{L}_{\text{dense}}$ is the dense-field reconstruction loss, $\text{sg}[\cdot]$ stops gradients through the dense target, λ_{compact} weights compact-state matching, and $d_{\mathcal{G}}$ compares Gaussian fields in their natural parameterizations. The target sequence is sampled at the same configured offsets $\Delta_1{:}\Delta_{H_w}$ as the rollout. For verifier training, s^+ is the score of the demonstrated candidate and s^{-r} is the score of alternative r:

$$\mathcal{L}_{\text{ver}} = -\log \frac{\exp(s^+/\tau_v)}{\exp(s^+/\tau_v) + \sum_{r=1}^{R}\exp(s^{-r}/\tau_v)} + \frac{\lambda_{\text{rank}}}{R}\sum_{r=1}^{R}[\gamma - s^+ + s^{-r}]_+. \tag{10}$$

Here R is the number of perturbed training alternatives, τ_v controls the ranking softness, γ is the required margin, λ_{rank} weights the margin penalty, and $[\cdot]_+$ is the positive-part operator. The first term ranks the positive training action above the alternatives; the second enforces the margin. Task, contact, collision, stability, relation, and object-state annotations form the auxiliary objective $\mathcal{L}_{\text{aux}}$ (Appendix C); these simulator-derived labels are used only for training and are unavailable to the deployed selector. The complete WAVE enhancement is optimized with

$$\mathcal{L}_{\text{WAVE}} = \lambda_{\text{world}}\mathcal{L}_{\text{world}} + \lambda_{\text{ver}}\mathcal{L}_{\text{ver}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}. \tag{11}$$

We use $\lambda_{\text{world}} = 1$, $\lambda_{\text{ver}} = 0.1$, and $\lambda_{\text{aux}} = 0.1$; unavailable annotations simply omit their corresponding terms from $\mathcal{L}_{\text{aux}}$.

All base-policy parameters, native sampler parameters, preprocessing, attention pattern, and action-generation objectives remain fixed. Gradients update only WAVE's feature adapter, scene decoder and pooler, compact dynamics, verifier, and their adapters. WAVE therefore reuses each policy's representations while preserving its native sampling process.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Benchmarks and frozen base-policy pairings. We evaluate WAVE on three simulation benchmarks. LIBERO (Liu et al., 2023) supplies four language-conditioned suites and is paired with π_0 (Black et al., 2024), $\pi_{0.5}$ (Physical Intelligence, 2025), and Fast-WAM (Yuan et al., 2026). RoboCasa (Nasiriany et al., 2024) provides the Human-50 kitchen setting and is paired with Cosmos Policy (Kim et al., 2026). RoboTwin 2.0 (Chen et al., 2025) evaluates bimanual manipulation under two scene conditions and is paired with π_0, $\pi_{0.5}$, Fast-WAM, Motus (Bi et al., 2025), and LingBot-VA (Li et al., 2026a).

Success rate is the fraction of episodes satisfying the benchmark's terminal criterion. For the execution diagnostics, if T_i is the number of native environment controls issued in episode i and $\mathcal{S}$ is the set of successful episodes, *Steps* denotes $\bar{T}_{\text{succ}} = |\mathcal{S}|^{-1} \sum_{i \in \mathcal{S}} T_i$. The invalid-action rate is $\text{IAR} = N_{\text{invalid}}/N_{\text{attempted}} \times 100\%$, where an invalid action is rejected by the simulator or execution interface; internal denoising and scoring operations are excluded. LIBERO reports an equal-weighted average over its four suites, and RoboTwin reports the two condition-wise scores and their mean. Detailed benchmark protocols appear in Appendix A.

Implementation details. All nine policy–benchmark pairs freeze base-policy parameters, preprocessing, native samplers, and action representations. Native deployment samples and executes one chunk ($K = 1$). WAVE deployment uses $K = 4$ chunks; it reconstructs a dense Gaussian scene capped at $N_{\max} = 4096$ primitives, pools $M = 64$ compact slots, predicts $H_w = 5$ candidate-conditioned rollouts, and ranks candidates using native consistency, learned scores, and structured adjustments. Flow/rectified-flow policies (π_0, $\pi_{0.5}$, Fast-WAM, Motus, and LingBot-VA) use native flow residuals; Cosmos Policy uses an action-latent EDM denoising residual. Only WAVE's feature adapter, scene decoder/pooler, compact dynamics, and verifier are trained; the selected chunk's native execution prefix is preserved. Tensor, temporal, and scoring details appear in Appendix B.

4.2 OVERALL POLICY ENHANCEMENT

Direct native deployment samples one candidate chunk ($K = 1$) and executes it directly. WAVE keeps each base policy fixed, samples $K = 4$ candidate chunks, and changes which chunk prefix is executed after selection. The main table establishes the cross-benchmark effect across nine matched pairs: three LIBERO policies, one RoboCasa policy, and five RoboTwin policies. Additional ablations of verifier channels, 3D prediction fidelity, compact-state capacity, and training supervision appear in Appendix D.

Table 1: **Policy enhancement across simulation benchmarks.** Each subtable compares frozen-policy native deployment ($K = 1$) with WAVE ($K = 4$); entries are success rates (%). RoboTwin averages use unrounded values from its two official conditions. Layout: (a) LIBERO upper left, (b) RoboCasa lower left, and (c) RoboTwin 2.0 right.

Methods	LIBERO				
	Spatial ↑	Object ↑	Goal ↑	Long ↑	Average ↑
π_0	96.8	98.8	95.8	85.2	94.2
π_0 + **WAVE**	**98.0**	**98.9**	**96.1**	**90.5**	**95.9**
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9
$\pi_{0.5}$ + **WAVE**	**99.7**	**99.1**	**99.2**	**97.6**	**98.9**
Fast-WAM	98.2	**100.0**	97.0	95.2	97.6
Fast-WAM + **WAVE**	**99.8**	**100.0**	**98.5**	**97.8**	**99.0**

Methods	RoboCasa Success rate ↑
Cosmos Policy	67.1
Cosmos Policy + **WAVE**	**79.5**

Methods	RoboTwin 2.0		
	Clean ↑	Randomized ↑	Average ↑
π_0	65.9	58.4	62.2
π_0 + **WAVE**	**73.2**	**69.8**	**71.5**
$\pi_{0.5}$	82.7	76.8	79.8
$\pi_{0.5}$ + **WAVE**	**90.0**	**86.1**	**88.1**
Fast-WAM	91.9	91.8	91.8
Fast-WAM + **WAVE**	**95.1**	**96.0**	**95.6**
Motus	88.7	87.0	87.8
Motus + **WAVE**	**93.6**	**90.9**	**92.3**
LingBot-VA	92.9	91.5	92.2
LingBot-VA + **WAVE**	**94.6**	**93.1**	**93.9**

Across all nine matched pairs, WAVE improves the reported success rate. On LIBERO (Table 1a), the average rises by 1.7, 2.0, and 1.4 percentage points for π_0, $\pi_{0.5}$, and Fast-WAM, respectively. Every

suite improves or is maintained for all three policies; the largest gains occur on LIBERO-Long (+5.3, +5.2, and +2.6 points), where longer horizons amplify the value of selecting among candidate chunks.

The pattern extends beyond LIBERO. On RoboCasa, Cosmos Policy rises from 67.1% to 79.5% (+12.4 points), showing candidate verification helps in cluttered kitchen scenes. On RoboTwin 2.0, average success rises for every policy: π_0 by 9.3 points, $\pi_{0.5}$ by 8.3, Fast-WAM by 3.8, Motus by 4.5, and LingBot-VA by 1.7. Both reported RoboTwin condition scores rise for all five policies, excluding scene-condition dependence. Since each pair shares model parameters, task split, episode budget, and execution interface, these comparisons quantify WAVE's overall enhancement of each native policy.

4.3 Selection Rules and Candidate Budget

Table 2 evaluates candidate-selection rules on RoboTwin 2.0. The all-dash row is direct native deployment: one candidate chunk is sampled and executed directly ($K = 1$). Every subsequent selector uses the same frozen base policy, $K = 4$ candidate bank, task set, episode budget, and execution interface under the same closed-loop evaluation protocol.

Selection-rule controls. Table 2 compares direct native deployment with five selectors across four frozen policies. The first row is direct deployment with $K = 1$; subsequent rows use the same $K = 4$ candidate bank. Checkmarks activate native consistency $\tilde{e}_{\Pi}^{(k)}$, the learned verifier, and the structured verifier, each standardized within the candidate bank before selection; the final row combines all three. Thus scored rows compare ranking rules under a fixed bank, while the first row provides the direct-deployment reference.

Table 2: **Selection-rule ablation on RoboTwin 2.0.** Mean success rates (%) over two conditions for four frozen policies. The dash row is native deployment ($K = 1$); other rows use a fixed $K = 4$ candidate bank. Checkmarks activate native consistency, learned verifier evidence, and structured verifier evidence; the final row combines all three.

Active selection terms			RoboTwin 2.0 Success rate ↑			
$\tilde{e}_{\Pi}^{(k)}$	$s_{\text{learn}}^{(k)}$	$s_{\text{struct}}^{(k)}$	π_0-based WAVE	$\pi_{0.5}$-based WAVE	Fast-WAM-based WAVE	Motus-based WAVE
–	–	–	62.2	79.8	91.8	87.8
✓			63.0	80.0	91.0	88.6
	✓		65.2	82.1	93.8	91.5
		✓	66.8	83.1	93.4	91.3
	✓	✓	70.0	86.9	94.1	92.0
✓	✓	✓	**71.5**	**88.1**	**95.6**	**92.3**

Within the common $K = 4$ bank, learned verifier scores and structured verifier adjustments are complementary: their fusion is strongest in every column. Relative to direct native deployment, the fused WAVE selector improves mean success by 9.3, 8.3, 3.8, and 4.5 points for π_0, $\pi_{0.5}$, Fast-WAM, and Motus, respectively; this end-to-end comparison includes the move from one direct sample to four candidates.

Candidate budget and end-to-end effect. Increasing the number of candidate chunks can improve both the chance that a useful action is present and the chance that the selector chooses it. Table 3 therefore reports the complete closed-loop effect together with decision time and invalid-action rate; it is not by itself a pure ranking attribution.

At the $K = 4$, success improves by 5.2 points on LIBERO-Long and 12.4 points on RoboCasa relative to baseline, while normalized inference time remains below $2\times$. Increasing to $K = 8$ adds only 0.4 and 1.4 points, respectively, at $3.42\times$ and $3.67\times$ time; RoboCasa's invalid-action rate changes only from 7.3% to 7.1%. This identifies $K = 4$ as the practical candidate-budget operating point.

4.4 Ablations with a Frozen Base Policy

The main-text ablation keeps base-policy parameters, the native action sampler, and execution interface fixed while varying the future-prediction representation used to score the candidate bank.

Table 3: **Candidate budget and decision-time cost.** Fixed $\pi_{0.5}$-based / LIBERO-Long and Cosmos Policy-based / RoboCasa configurations use $K \in \{1, 2, 4, 8\}$. $K = 1$ is native deployment; larger K uses candidate comparison. Inference time is wall-clock time per decision, normalized to $K = 1$ and including generation, shared reconstruction, rollout, verification, and selection; invalid-action rate is reported only for RoboCasa. $K = 4$ is the default.

K	LIBERO-Long		RoboCasa		
	Success rate ↑	Inference time ↓	Success rate ↑	Inference time ↓	Invalid-action rate (%) ↓
1	92.4	**1.00**×	67.1	**1.00**×	14.1
2	94.7	1.43×	70.8	1.56×	10.5
4 (default)	97.6	1.91×	79.5	1.98×	7.3
8	**98.0**	3.42×	**80.9**	3.67×	**7.1**

Future prediction and representation modes. Table 4 connects representation choice to closed-loop behavior. The static-initial-frame control repeats the current scene at every future time point. The 2D-video, latent-space, object-token, point-primitive, and 3D-VLA rows replace the Gaussian future predictor while keeping the candidate bank, action-conditioning interface, future offsets, and execution protocol fixed. Intermediate representations provide partial gains, but the full Gaussian configuration achieves the best joint outcome: success rises from 63.9% to 79.5% on RoboCasa and from 92.5% to 98.9% on LIBERO, while RoboCasa invalid-action rate falls to 7.3% and successful-episode Steps to 106.8. The corresponding backbone choices are specified in Appendix B.2.

Table 4: **Controlled future-prediction ablation.** The frozen base policy, candidate bank, and execution protocol are fixed; only the future representation or predictor changes. "Static initial frame" repeats the current scene at every future point. Invalid-action rate and *Steps* use Cosmos Policy / RoboCasa; success columns use the matched $\pi_{0.5}$ / LIBERO and Cosmos Policy / RoboCasa settings.

Future prediction mode	$\pi_{0.5}$-based / LIBERO	Cosmos Policy-based / RoboCasa		
	Success rate ↑	Success rate ↑	Invalid-action rate (%) ↓	Steps ↓
Static initial frame	92.5	63.9	14.2	136.7
2D Video Future	95.6	69.8	10.6	128.2
Latent-space Future	96.1	70.5	10.1	126.9
Object-centric tokens	96.8	72.9	9.0	123.7
Point primitives	96.1	74.1	8.8	126.8
3D-VLA Prediction Future	96.2	72.3	9.7	126.1
3D Gaussian Future (Ours)	**98.9**	**79.5**	**7.3**	**106.8**

5 Conclusion

WAVE turns the candidate diversity of a generative robot policy into decision-time consequence comparison. It reconstructs the current scene once, branches compact action-conditioned Gaussian futures for the candidate chunks, and fuses the native consistency score with learned verifier scores and structured verifier adjustments before execution. Across nine matched frozen-policy comparisons on LIBERO, RoboCasa, and RoboTwin, WAVE reaches 99.0% LIBERO success, 79.5% RoboCasa success, and 95.6% RoboTwin success, improving every reported base-policy comparison. The ablations further show that four candidates provide a strong accuracy–compute operating point and that dense-to-compact future prediction is central to the gain.

WAVE therefore provides a compact decision interface for generative robot policies: share the present scene, predict one consequence per candidate chunk, and rank the candidates before execution. This interface leaves the underlying policy and its native sampling process unchanged while making geometric consequences available for selection, providing a basis for calibrated verification budgets and future real-robot deployment.

More broadly, these results show that candidate diversity need not remain an uncontrolled sampling artifact. WAVE turns it into a measurable decision resource: a frozen policy retains its native behavior while gaining a shared geometric criterion for choosing among its own predictions. This makes consequence-aware selection portable across policy families, embodiments, and task suites.

AI USE STATEMENT

Generative AI tools were used to assist with drafting, condensation, structural editing, and copy-editing of the manuscript. They were also used to improve wording and check local consistency during revision. The authors verified the final technical content, citations, equations, tables, figures, and numerical results, and remain responsible for the complete manuscript and all reported claims.

ETHICS STATEMENT

This work studies robot manipulation in controlled simulation using LIBERO, RoboCasa, and RoboTwin 2.0. The study involves no new human or animal subjects, no personal data collection, and no physical-robot deployment. Simulator-derived task, contact, collision, relation, and object-state labels are used only as training supervision; they are not available to the deployed WAVE selector. The predicted consequences and verification scores are therefore evaluated as decision-support signals rather than safety guarantees.

REPRODUCIBILITY STATEMENT

The method definition and learning objectives are given in Section 3. Benchmark definitions and evaluation contracts are documented in Appendix A; implementation, policy interfaces, future-prediction backbones, and scoring details are provided in Appendix B; training objectives and optimization are described in Appendix C; and the additional ablations are collected in Appendix D. The experiments use frozen base-policy parameters and fixed benchmark splits within each matched comparison. Reproduction therefore requires the stated policy configurations, preprocessing and action interfaces, split manifests, and evaluator versions.

REFERENCES

Hadi Alzayer, Wenlong Huang, Haonan Chen, Christopher Luey, Lvmin Zhang, Maneesh Agrawala, Gordon Wetzstein, Li Fei-Fei, Yilun Du, Jiajun Wu, and Jia-Bin Huang. Masked Visual Actions for Unified World Modeling. *arXiv preprint arXiv:2607.19343*, 2026. URL https://arxiv.org/abs/2607.19343.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A Unified Latent Action World Model. *arXiv preprint arXiv:2512.13030v2*, 2025. URL https://arxiv.org/abs/2512.13030v2.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *arXiv preprint arXiv:2303.04137*, 2023. URL https://arxiv.org/abs/2303.04137.

Wenkai Guo, Guanxing Lu, Haoyuan Deng, Zhenyu Wu, Yansong Tang, and Ziwei Wang. VLA-Reasoner: Empowering Vision-Language-Action Models with Reasoning via Online Monte Carlo Tree Search. *arXiv preprint arXiv:2509.22643*, 2025. URL https://arxiv.org/abs/2509.22643.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. *arXiv preprint arXiv:1811.04551*, 2018. URL https://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. *arXiv preprint arXiv:1912.01603*, 2019. URL https://arxiv.org/abs/1912.01603.

Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. PointWorld: Scaling 3D World Models for In-The-Wild Robotic Manipulation. *arXiv preprint arXiv:2601.03782*, 2026. URL https://arxiv.org/abs/2601.03782.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuhler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *arXiv preprint arXiv:2308.04079*, 2023. URL https://arxiv.org/abs/2308.04079.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling Verification Can Be More Effective than Scaling Policy Learning for Vision-Language-Action Alignment. *arXiv preprint arXiv:2602.12281*, 2026. URL https://arxiv.org/abs/2602.12281.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. *arXiv preprint arXiv:2601.21998*, 2026a. URL https://arxiv.org/abs/2601.21998.

Wei Li, Jizhihui Liu, Li Yixing, Junwen Tong, Rui Shao, and Liqiang Nie. ConsisVLA-4D: Advancing Spatiotemporal Consistency in Efficient 3D-Perception and 4D-Reasoning for Robotic Manipulation. *arXiv preprint arXiv:2605.05126*, 2026b. URL https://arxiv.org/abs/2605.05126.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. *arXiv preprint arXiv:2306.03310*, 2023. URL https://arxiv.org/abs/2306.03310.

Yuejiang Liu, Fan Feng, Lingjing Kong, Weifeng Lu, Jinzhou Tang, Kun Zhang, Kevin Murphy, Chelsea Finn, and Yilun Du. World Action Verifier: Self-Improving World Models via Forward-Inverse Asymmetry. *arXiv preprint arXiv:2604.01985*, 2026. URL https://arxiv.org/abs/2604.01985.

Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. ManiGaussian: Dynamic Gaussian Splatting for Multi-task Robotic Manipulation. *arXiv preprint arXiv:2403.08321*, 2024. URL https://arxiv.org/abs/2403.08321.

Guanxing Lu, Baoxiong Jia, Puhao Li, Yixin Chen, Ziwei Wang, Yansong Tang, and Siyuan Huang. GWM: Towards Scalable Gaussian World Models for Robotic Manipulation. *arXiv preprint arXiv:2508.17600*, 2025. URL https://arxiv.org/abs/2508.17600.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. *arXiv preprint arXiv:2406.02523*, 2024. URL https://arxiv.org/abs/2406.02523.

Octo Model Team. Octo: An Open-Source Generalist Robot Policy. *arXiv preprint arXiv:2405.12213*, 2024. URL https://arxiv.org/abs/2405.12213.

Physical Intelligence. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-Time Enhancement of Generative Robot Policies via Predictive World Modeling. *arXiv preprint arXiv:2502.00622v4*, 2026. URL https://arxiv.org/abs/2502.00622v4.

Zhenguo Sun, Yu Sun, Hande Huang, and Alois Knoll. ω-EVA: Envision, Verify, and Act with Latent Interactive World Models. *arXiv preprint arXiv:2606.09457*, 2026. URL https://arxiv.org/abs/2606.09457.

Wei-Cheng Tseng, Gashon Hussein, Yuzhu Dong, Allen Z. Ren, Lucy X. Shi, XuDong Wang, Sergey Levine, Zhaoshuo Li, Jinwei Gu, Florian Shkurti, Ming-Yu Liu, and Quan Vuong. SC3-Eval: Evaluating Robot Foundation Models via Self-Consistent Video Generation. *arXiv preprint arXiv:2606.18610*, 2026. URL https://arxiv.org/abs/2606.18610.

Yiqi Wang, Mrinal Verghese, and Jeff Schneider. Latent Policy Steering with Embodiment-Agnostic Pretrained World Models. *arXiv preprint arXiv:2507.13340*, 2025. URL https://arxiv.org/abs/2507.13340.

Zhongyu Xia, Yousen Tang, Bingqing Wei, and Yongtao Wang. 3DVLA: Enhancing Vision-Language-Action Models via 3D Spatial and Instance Understanding. *arXiv preprint arXiv:2605.29416*, 2026. URL https://arxiv.org/abs/2605.29416.

Jiarui Yang, Jiale Zhange, Jiawei Li, Hang Guo, Wen Huang, Jinpeng Wang, Peidong Liu, and Shu-Tao Xia. LAWM-3D: Learning 3D-Aware Latent Actions from Human Videos for Generalizable Robot World Models. *arXiv preprint arXiv:2608.05706*, 2026. URL https://arxiv.org/abs/2608.05706.

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*, 2026. URL https://arxiv.org/abs/2603.16666.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. *arXiv preprint arXiv:2403.03954*, 2024. URL https://arxiv.org/abs/2403.03954.

Chushan Zhang, Ruihan Lu, Jinguang Tong, Xuesong Li, Yikai Wang, and Hongdong Li. EvoScene-VLA: Evolving Scene Beliefs Inside the Action Decoder for Chunked Robot Control. *arXiv preprint arXiv:2605.21862*, 2026a. URL https://arxiv.org/abs/2605.21862.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness VLA: Steering Frozen VLAs into Reliable Manipulation Primitives via Memory-Guided Agents. *arXiv preprint arXiv:2607.08448*, 2026b. URL https://arxiv.org/abs/2607.08448.

Zijian Zhang, Yuqing Jiang, Qian Cheng, Xiaofan Li, Si Liu, Ding Zhao, Ping Luo, Weitao Zhou, and Haibao Yu. GaussianDream: A Feed-Forward 3D Gaussian World Model for Robotic Manipulation. *arXiv preprint arXiv:2605.20752*, 2026c. URL https://arxiv.org/abs/2605.20752v2.

Zijian Zhang, Yuqing Jiang, Weitao Zhou, Minglei Li, Jinhao Zhang, Yao Mu, Xiaofan Li, Hao Zhao, and Haibao Yu. GaussianWAM: Distilling Geometry and Semantics from 3D Gaussian Fields into World-Action Models. *arXiv preprint arXiv:2608.24714*, 2026d. URL https://arxiv.org/abs/2608.24714.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. *arXiv preprint arXiv:2304.13705*, 2023. URL https://arxiv.org/abs/2304.13705.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D Vision-Language-Action Generative World Model. *arXiv preprint arXiv:2403.09631*, 2024. URL https://arxiv.org/abs/2403.09631.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, et al. τ_0-WM: A Unified Video-Action World Model for Robotic Manipulation. *arXiv preprint arXiv:2606.01027*, 2026. URL https://arxiv.org/abs/2606.01027.

A BENCHMARK PROTOCOLS

This appendix records the benchmark definitions and evaluation contracts used by Tables 1a–1c. The main text lists only the benchmark and baseline sets.

Across all three benchmarks, direct native and +WAVE deployments use the same base-policy parameters, task and instruction splits, episode budget, replanning cadence, and execution interface. The base policy remains frozen; only the WAVE enhancement changes.

A.1 LIBERO

LIBERO (Liu et al., 2023) is a language-conditioned tabletop-manipulation benchmark with four standard suites: Spatial, Object, Goal, and Long. Each suite contains 10 tasks and 500 training demonstrations, with 50 demonstrations per task. We evaluate 50 episodes per task, or 500 episodes per suite, and keep the suite, task and instruction splits and episode budget identical within each native/+WAVE pair.

A.2 ROBOCASA

RoboCasa (Nasiriany et al., 2024) is a simulated kitchen-manipulation benchmark. We use the 24-task non-navigation subset of the Human-50 setting, excluding the navigation task, with 50 human demonstrations per task. Evaluation uses 50 trials per task across five held-out scenes (10 trials per scene); object instances are unseen, and two scenes per task use visual styles absent from training.

A.3 ROBOTWIN 2.0

RoboTwin 2.0 (Chen et al., 2025) evaluates coordinated bimanual manipulation under two scene conditions. We use the mixed-training track with 2,500 clean and 25,000 additional demonstrations spanning 50 tasks. The corresponding policies use 30k training steps and are evaluated for 100 episodes per task under each condition.

B ARCHITECTURE AND INFERENCE DETAILS

This appendix follows the decision pipeline in Section 3: candidate generation, shared scene construction, compact prediction, geometric verification, and final selection. Policy-specific interfaces are instantiated per base policy; the $\pi_{0.5}$ configuration is shown only as an example. The frozen-policy enhancement is summarized in Section 3.4.

B.1 WAVE CONFIGURATION

WAVE compares $K = 4$ candidates using one shared current scene. The dense Gaussian field is capped at $N_{\max} = 4096$ primitives before pooling into $M = 64$ compact slots; each candidate then predicts $H_w = 5$ future time points aligned to its action chunk (Appendix B.5). The predictive verification pathway uses a hidden width of 128, $d_{\mathrm{SH}} = 9$ spherical-harmonic appearance coefficients per slot, and eight soft object groups for geometric evidence. When combining native action-generation consistency and verification scores, WAVE standardizes each across candidates and sets the verification-score weight to $\beta = 1$ by default (Appendix B.7).

B.2 FUTURE-PREDICTION BACKBONES

The representation ablations use a common action-conditioned interface, future offsets, target frames, and training budget; the future representation/predictor branch changes under this matched protocol. The 2D-video row uses Wan2.2-TI2V-5B (**?**) with a lightweight action-conditioning adapter. The latent row instantiates a V-JEPA 2-AC-style latent predictor (**?**). The object-token row uses a SAVi slot encoder with a SlotFormer-style action-conditioned predictor (**?**). The point-primitive row uses a DINOv3 feature encoder with a Point Transformer V3 point-flow predictor, following the representation family of PointWorld (Huang et al., 2026). The 3D-VLA row instantiates a 3D-VLA-style point-cloud future predictor (Zhen et al., 2024). Each branch is trained with an interface adapter

that maps its output to the common compact future-state representation consumed by the verifier; the structured operators, fixed-view renderer, and evaluation code are then shared. These are controlled future-prediction branches rather than replacement policy baselines: all branches consume the same observation, instruction, and candidate action chunk. Wan2.2 is adapted with action conditioning before evaluation; its native text/image-to-video interface is not used as an action predictor.

B.3 Shared Context and Native Action Sampling

For the $\pi_{0.5}$ sampling configuration, the PaliGemma prefix contains visual and language tokens together with a world-query segment from the GaussianDream encoder. SigLIP encodes visual inputs and projects them to the Gemma prefix width; language inputs use the backbone's tokenizer and embeddings. Prefix attention and its key-value cache are computed once per observation. The action suffix then reuses this cache for independently initialized Gaussian noise tensors, producing a candidate bank of shape $[B, K, H_a, d_a]$ for batch size B. Current-scene decoding uses the shared observation context and does not take the sampled action chunks as input.

For the flow and rectified-flow policies, the action sampler follows the noise-to-action path in Equation 8, integrating from $\tau = 1$ to $\tau = 0$ with $J = 10$ Euler steps. The flow-policy configurations use chunk lengths $H_a = 10/50$ on LIBERO/RoboTwin, respectively, and pad $7/14$ active action dimensions to $d_a = 32$; Cosmos Policy uses its native action-latent sequence length and EDM sampler on RoboCasa. Each update refines the entire chunk, so integration steps and proposed controls are distinct quantities. With $\Delta\tau = -1/J$, the update is

$$\mathbf{x}^{(k)}_{\tau+\Delta\tau} = \mathbf{x}^{(k)}_{\tau} + \Delta\tau\, v_\phi(\mathbf{x}^{(k)}_{\tau}, \tau; c_t). \tag{12}$$

The sampled initial-noise tensors specify the candidate bank. Distinct samples can still decode to similar actions; K controls the number of candidates rather than guaranteeing distinct behaviors.

The same bank records the native denoising trace used by the consistency readout for each evaluated policy. Flow and rectified-flow policies retain the action vector field in their native action or latent-action coordinates; Cosmos Policy retains the action-frame states along its EDM sampler path. For Cosmos, each sampled action-latent sequence is decoded by its native action decoder into executable controls before compact rollout and action–scene verification, while e_Π remains evaluated in latent coordinates. These readouts reuse the candidate-generation evaluations and add no candidate passes. Future-state and value frames in the Cosmos sequence are excluded from the action score.

B.4 Dense Scene Decoding and Compact Pooling

The GaussianDream encoder uses three sparsely sampled observation frames. A 3D-aware visual encoder extracts patch features that interact across spatial and temporal axes with 1024 learnable world queries. The resulting world segment occupies the 2048-dimensional multimodal prefix space. A convolutional decoder reshapes and upsamples it into a 256×256 Gaussian field. Its geometry branch predicts depth, rotation, scale, and opacity; its appearance branch combines decoder features with the current RGB observation to predict spherical-harmonic coefficients.

Camera intrinsics back-project depth to camera-frame centers. Extrinsics are supplied as world-to-camera matrices and inverted to transform centers into the calibrated world-space frame; Gaussian orientations are rotated consistently. Quaternions use scalar-first ordering (w, x, y, z) and are normalized. Comparisons in the calibrated world-space frame require the intrinsics and extrinsics for the actual camera views, with workspace bounds and geometric thresholds expressed in the same units.

Before learned pooling, adaptive average pooling for square inputs, or deterministic subsampling otherwise, reduces the dense field to at most $N_{\max} = 4096$ primitives. Equation 4 then assigns them to $M = 64$ compact slots using unit-normalized keys and queries, scaled by the inverse square root of their width, and an opacity floor of 10^{-6}. Each slot uses the same assignment weights for position, scale, opacity, rotation, and appearance, preserving correspondence among its fields. The projection in Equation 4 clamps scales to $[10^{-6}, 10^{3}]$ and opacities to $[0, 1]$, and normalizes pooled quaternions; a near-zero quaternion maps to the identity rotation. This is a componentwise field summary, not an intrinsic rotation average or an exact mixture reduction. These slots summarize scene regions; their indices do not establish persistent object identity.

B.5 Compact Dynamics and Temporal Alignment

The candidate-conditioned compact dynamics model W_θ receives current Gaussian fields of shape $[B, M, C_f]$, actions of shape $[B, K, H_a, d_a]$, and a pooled language vector. Here $C_f = 3 + 3 + 1 + 4 + d_{\mathrm{SH}} = 20$ concatenates center, scale, opacity, quaternion, and appearance. Each control is projected to width 128 and processed by a GRU over the complete chunk. Nearest-neighbor resampling selects H_w encoded positions spanning that sequence. A global scene feature is formed by opacity-weighted averaging: if $\mathbf{p}_{m,t}$ is the encoded feature of slot m, then

$$\bar{\mathbf{p}}_t = \frac{\sum_{m=1}^{M} \alpha_{m,t} \mathbf{p}_{m,t}}{\max(\sum_{m=1}^{M} \alpha_{m,t}, \varepsilon)}. \tag{13}$$

Here $\varepsilon = 10^{-6}$ bounds the denominator. Let E_A include the action projection and GRU, E_L denote the language projection, and $\mathbf{e}_h$ be a learned embedding of prediction index h. The candidate condition in Equation 5 is

$$\mathbf{c}_{k,h} = \mathrm{Fuse}_\theta\Big(\mathrm{Resample}_h[E_A(\mathbf{A}_t^{(k)})], E_L(l), \bar{\mathbf{p}}_t, \mathbf{e}_h\Big). \tag{14}$$

The resampling operator selects approximately equally spaced encoded action indices, while $\mathbf{e}_h$ identifies the output prediction time point. The target at offset Δ_h supplies its temporal meaning during training. Prediction offsets and action-feature indices are therefore distinct: changing chunk timing requires corresponding target alignment, rather than only changing the number of embeddings. For the LIBERO defaults, context offsets are $(-10, -5, 0)$ and future offsets are $(1, 2, 3, 4, 5)$ in the dataset time base. With $H_a = 10$ and $H_w = 5$, the selected encoded action indices are $(0, 2, 4, 7, 9)$. The predicted trajectory therefore uses encoded features spanning the full candidate chunk; the selected indices are not controls executed at those five offsets.

The fused 128-dimensional condition generates FiLM scale and shift parameters for the encoded initial slots. Predictions at different time points share this initial representation; they do not feed predicted states back through the dynamics network. Five heads predict residuals with field widths $(3, 3, 1, 4, d_{\mathrm{SH}})$, yielding trajectories with leading dimensions $[B, K, H_w, M]$. Center residuals use $\tanh$ with a default per-coordinate step bound of $0.25/H_w$; attribute residuals use a default step scale of 0.05. Residuals accumulate from the shared current field as in Equation 5. Since each center increment is bounded by $0.25/H_w$, the triangle inequality bounds displacement at time point h by $0.25h/H_w$ per coordinate, and hence by 0.25 over the full horizon. This is a componentwise update budget in scene coordinates. The update operator $\mathcal{U}$ adds cumulative center and appearance residuals directly. It adds scale residuals to the logarithm of the initial scales and exponentiates, and adds opacity residuals to the initial logits before applying a sigmoid. Initial opacity is clipped to $[10^{-5}, 1 - 10^{-5}]$ for finite logits. Quaternion components are updated additively and renormalized. These parameterizations maintain numeric field constraints while leaving physical plausibility to comparative verification.

B.6 Verifier Channels and Geometric Operators

The learned verifier encodes current and future fields at width 128. Separate GRUs summarize the future time points and action sequence. A joint encoder combines the current scene, final and mean future scenes, the temporal summary, the action summary, and language; a linear head produces four learned verifier scores $\boldsymbol{\ell}^{(k)} \in \mathbb{R}^4$. Structured verifier adjustments augment these scores with measurements of progress, spatial consistency, interaction, and motion.

For object-level measurements, the verifier selects eight high-opacity primitives as anchors and forms soft groups using distance and opacity weights. A group's extent averages the distance from its center to each member plus that primitive's scale norm. Groups are recomputed at each time point without temporal identity matching. Their indices follow anchor ordering rather than persistent object identity, so group displacement and relation drift are comparative scene descriptors. Primitive-level overlap and bounds complement these group-level measurements.

Writing $u_j^{(k)} = \ell_j^{(k)} + \delta_j^{(k)}$ for the combined channel score, the default corrections are

$$\begin{aligned}
u_g^{(k)} &= \ell_g^{(k)} + \Delta^{(k)}_{\text{scene-lang}} + \Delta^{(k)}_{\text{object-lang}}, \\
u_s^{(k)} &= \ell_s^{(k)} + 0.1R^{(k)}_{\text{relation}} - C^{(k)}_{\text{act-scene}} - 0.5C^{(k)}_{\text{bounds}} - 0.5C^{(k)}_{\text{relation drift}}, \\
u_c^{(k)} &= \ell_c^{(k)} - C^{(k)}_{\text{contact}} - C^{(k)}_{\text{collision}}, \\
u_p^{(k)} &= \ell_p^{(k)} - C^{(k)}_{\text{bounds}} - C^{(k)}_{\text{smooth}} - C^{(k)}_{\text{collision}} \\
&\quad - C^{(k)}_{\text{stability}} - 0.25C^{(k)}_{\text{relation drift}}.
\end{aligned} \tag{15}$$

The mixture weights in Equation 6 are learned global logits normalized by a softmax. Channel ablations mask the corresponding logits before normalization, leaving disabled weights exactly zero; each mask therefore removes both components of that channel. Goal, spatial, contact, and plausibility describe evidence roles, rather than a strict learned-versus-structured partition. The measurements in Equation 15 have the following meanings.

Goal and relation evidence. Global goal progress is the change in scene–language cosine alignment. Object-level progress compares the maximum object–language alignment in the current and final scenes, without requiring the maximizing group to retain an identity. A learned relation head evaluates ordered, non-self group pairs at the final time point for *inside*, *on top of*, *near*, and *grasped by*. Its outputs are aggregated by maxima over pairs; R_{relation} is the maximum sigmoid confidence over the four labels. Thus R_{relation} supplies relation confidence, while the learned goal channel and alignment changes supply instruction relevance; high confidence alone does not establish task completion.

Action–scene and contact evidence. In the default WAVE enhancement, the action–scene cost is the mean squared discrepancy between opacity-weighted scene-centroid displacement and the temporally resampled cumulative sum of the first three normalized action channels, scaled by 0.25. Contact evidence anchors this path at the current scene centroid and measures approach and proximity to current soft-group extents. The path is a ranking proxy, not a forward-kinematic end-effector trajectory. For a joint-action base policy, these channels are joint commands rather than Cartesian displacement. Interpreting these terms as physical distances requires an embodiment-specific action-to-workspace conversion.

Overlap, motion, and support evidence. Collision cost averages the positive increase in opacity-weighted pairwise overlap over future time points relative to the current scene. Overlap uses a smooth positive-part function of the sum of scale-norm radius proxies minus center distance, plus a margin; only distinct primitive pairs contribute. Bounds penalize centers outside the workspace box; smoothness measures second differences over prediction indices; relation drift measures changes in distances between soft groups. Endpoint stability combines final inter-time-point group displacement with support from a $z = 0$ floor or neighboring group extents; this floor convention assumes the scene coordinates place the support plane at zero. Finite differences are measured per prediction time point, without division by elapsed seconds. These costs describe coarse geometric consistency rather than mesh penetration or contact certificates; their physical interpretation depends on coordinate units, time offsets, and the configured support plane.

B.7 Policy Scoring, Normalization, and Selection

For every evaluated policy, $e_{\Pi}^{(k)}$ is evaluated in the generator's native action-generation coordinates and standardized within the current candidate bank. It measures agreement between a candidate and the denoising dynamics that produced it; it is a bank-relative native consistency score rather than a likelihood. For flow and rectified-flow policies, the implementation uses the midpoint vector-field readout defined below as a compact native-consistency proxy for the recorded sampler path. The fused WAVE selector combines this score with verifier evidence, while the native-score selector uses the consistency score alone. Both configurations select an unchanged candidate from the policy's candidate bank.

Flow and rectified-flow action experts. For π_0, $\pi_{0.5}$, Fast-WAM, Motus, and LingBot-VA, the action generator exposes a flow or rectified-flow trace. Let $\mathbf{z}_{\Pi}^{(k)}$ denote the candidate endpoint in the

generator's native action-generation coordinates, which may be a raw action or a latent action, and let Ω_a mask valid action entries. With the convention in Equation 8, the straight interpolation between sampled noise and endpoint has constant velocity $\boldsymbol{\epsilon}^{(k)} - \mathbf{z}_\Pi^{(k)}$. Comparing the learned velocity at its midpoint with this displacement gives

$$e_{\Pi,\text{flow}}^{(k)} = -\frac{1}{|\Omega_a|} \left\| \Omega_a \odot \left[v_{\phi,a}\left(\tfrac{1}{2}(\boldsymbol{\epsilon}^{(k)} + \mathbf{z}_\Pi^{(k)}), \tfrac{1}{2}; c_t \right) - \left(\boldsymbol{\epsilon}^{(k)} - \mathbf{z}_\Pi^{(k)} \right) \right] \right\|_F^2 . \tag{16}$$

Here $\boldsymbol{\epsilon}^{(k)}$ is the candidate's initial noise and $v_{\phi,a}$ is the action-coordinate projection of the native vector field. The score reuses the same conditioning and averages over valid native action entries. Model-specific time directions, latent-action parameterizations, and velocity conventions are converted before evaluation; the score is not numerical integration error or an action likelihood.

Cosmos Policy's latent diffusion. Cosmos Policy uses EDM denoising in a joint latent sequence containing action, future-state, and value frames. Let $z_{0,a}^{(k)}$ be the generated action-latent endpoint for candidate k, $z_{\sigma_j}^{(k)}$ the recorded joint latent state along that candidate's native denoising path, $D_{\phi,a}$ the action-frame projection of the denoiser, and $\mathcal{M}_a$ the mask of valid action-latent entries. Its native consistency score is the negative action-masked reconstruction error

$$e_{\Pi,\text{EDM}}^{(k)} = -\frac{1}{|\mathcal{T}|\,|\mathcal{M}_a|} \sum_{\sigma_j \in \mathcal{T}} \left\| \mathcal{M}_a \odot \left[D_{\phi,a}\left(z_{\sigma_j}^{(k)}; \sigma_j, c_t \right) - z_{0,a}^{(k)} \right] \right\|_2^2 . \tag{17}$$

The states $z_{\sigma_j}^{(k)}$ and action endpoint $z_{0,a}^{(k)}$ are taken from the same native sampler trajectory, and $\mathcal{M}_a$ excludes future-state and value entries. This is the diffusion counterpart of Equation 16; it uses the policy's native noise levels, preconditioning, and scheduler.

The generic score in Equation 3 is $e_\Pi = e_{\Pi,\text{flow}}$ for the five flow or rectified-flow policies and $e_\Pi = e_{\Pi,\text{EDM}}$ for Cosmos Policy.

Native and verifier scores are standardized separately across the K candidates of each observation, using the population standard deviation clamped below at 10^{-6}. For a score vector $z_{1:K}$, normalization subtracts its mean and divides by $\max(\text{std}_{\text{pop}}(z_{1:K}), 10^{-6})$. Batch elements are never mixed, and a singleton bank retains its raw scores. The resulting bank-relative scores enter Equation 3, with $\beta = 1$ by default. Disabling normalization makes the tilded terms equal to the raw scores. Because the normalization statistics depend on the bank, adding candidates can change the balance between the two terms. For $K = 2$ and $\beta = 1$, opposite rankings give equal combined scores in exact arithmetic when neither standard-deviation floor is active; exact ties are resolved by the first index.

If scene decoding, compact rollout, or verification raises an exception, the system falls back to the candidate with the highest native action-generation consistency score. This exception handling occurs after candidate generation and native-consistency score evaluation; it does not define a learned abstention rule. The decision record retains raw and combined scores, geometric costs, the selected index, and any fallback reason. Both normal and fallback paths return a complete chunk. The external controller executes its configured prefix $H_e \le H_a$, then re-observes and replans; the sampler does not itself determine the physical replanning interval.

C Learning Objectives and Schedule

The experiments train only the WAVE enhancement: the base-policy parameters, native sampler parameters, and candidate generation remain fixed, while the scene, dynamics, and verification pathway are optimized as summarized in Section 3.4.

C.1 Training Data and Supervision Paths

A demonstration segment from the benchmark training split provides observation context, an instruction, robot state, and an expert action chunk. Evaluation episodes and scenes are never used to update WAVE or to supply its simulator-derived labels. WAVE constructs one current compact scene from the observation context. In the frozen-policy experiments, the expert chunk conditions the positive compact rollout and the verifier receives R perturbed training alternatives derived from it;

the alternatives are passed through the same compact dynamics and verifier, but only the positive future receives dense reconstruction targets. Neither operation updates the base policy. This training set of one positive and R alternatives differs from the K candidate chunks compared at inference.

Current and future RGB, depth, and motion targets supervise the dense Gaussian branch. Its future predictions are pooled with stopped gradients to provide compact rollout targets at the same offsets. The expert-conditioned future is matched to the demonstration; perturbed alternatives contribute through relative verifier ranking. Task, interaction, relation, and object losses use their corresponding annotations. Future observations and these annotations serve as training targets and are not inputs to the decision-time procedure.

C.2 Dense Supervision and Compact Targets

Appearance, geometry, and motion. Dense supervision follows GaussianDream (Zhang et al., 2026c) and constrains complementary properties of the predicted field. With rendered RGB $\hat{I}$, rendered depth $\hat{D}$, targets I, D, a valid-depth mask M_D, predicted center displacement $\Delta\hat{\boldsymbol{\mu}}$, target displacement $\Delta\boldsymbol{\mu}$, and motion mask M_F, representative terms are

$$\begin{aligned} \mathcal{L}_{\text{rgb}} &= \|\hat{I} - I\|_1, \\ \mathcal{L}_{\text{depth}} &= \frac{\|M_D \odot (\hat{D} - D)\|_1}{\|M_D\|_1 + \varepsilon}, \\ \mathcal{L}_{\text{motion}} &= \text{smoothL1}(M_F \odot (\Delta\hat{\boldsymbol{\mu}} - \Delta\boldsymbol{\mu})) . \end{aligned} \tag{18}$$

The dense objective combines these terms with the available appearance, depth, mask, future-depth, and Gaussian-regularization terms, i.e., $\mathcal{L}_{\text{dense}} = \mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{motion}} + \mathcal{L}_{\text{other}}$, where $\mathcal{L}_{\text{photo}} = 0.15\mathcal{L}_{\text{rgb}} + 0.85\mathcal{L}_{\text{SSIM}} + 0.1\mathcal{L}_{\text{LPIPS}}$. Depth-head regression uses L_1, while rendered Gaussian depth uses smooth-L_1 on valid depth and opacity support; these are included in the depth terms above. Pseudo scene flow constrains center displacement through masked regression, with coordinate weights accounting for the target convention. This geometric motion loss is distinct from the policy's flow-matching objective. Target availability determines which supervised terms enter the loss; each active term is applied at its current or future time point.

Online dense-to-compact targets. During training, a dense future branch receives the current observation context, policy features, instruction, and demonstrated expert chunk through its FiLM conditioning and predicts a field at each configured offset,

$$\mathcal{G}^{\text{dense,E}}_{t+\Delta_h} = F^{\text{dense}}_{\eta}\big(o_t, c_t, \mathbf{A}^{\text{E}}_t, l, \Delta_h\big) , \qquad h = 1, \ldots, H_w. \tag{19}$$

Its rendered RGB, depth, mask, and motion outputs are matched to future observations by $\mathcal{L}_{\text{dense}}$. The target for the expert compact rollout is then the dense decoder's predicted future field, pooled with stopped gradients at the same offsets. This dense future branch is training-only: deployment uses S_η for the current field and W_θ for candidate rollouts. It is an online target from the trained dense branch, not a separately pretrained teacher or a ground-truth Gaussian reconstruction. Future observations are never supplied to the decision-time selector. Shared queries define the slot correspondence used for matching; they do not assume tracked object identities. For the fields defined in Section 3.2, the distance in Equation 9 is

$$\begin{aligned} d_{\mathcal{G}}(\hat{\mathcal{G}}, \mathcal{G}) = \lambda_x \ \text{smoothL1}(\hat{\boldsymbol{\mu}}, \boldsymbol{\mu}) + \lambda_s \ \text{smoothL1}(\hat{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) + \lambda_\alpha \ \text{smoothL1}(\hat{\alpha}, \alpha) \\ + \lambda_r \left(1 - |\langle \hat{\mathbf{r}}, \mathbf{r} \rangle|\right) + \lambda_h \ \text{smoothL1}(\hat{\mathbf{h}}, \mathbf{h}). \end{aligned} \tag{20}$$

Each field term is averaged over its batch, temporal, slot, and applicable channel dimensions. Default weights $(\lambda_x, \lambda_s, \lambda_\alpha, \lambda_r, \lambda_h)$ are $(1, 0.25, 0.25, 0.25, 0.1)$. The absolute quaternion inner product makes the rotation term invariant to the equivalent signs $\mathbf{r}$ and $-\mathbf{r}$.

Coupling and gradient flow. Dense reconstruction trains the scene representation, while compact matching transfers its future geometry and appearance to the candidate model. Gradients are stopped through the pooled future target, as specified in Equation 9; the current-state pooler and compact predictor remain on the prediction side of this objective. Expert-conditioned FiLM features connect the compact action condition to dense future reconstruction. The current dense template is detached

before pooling, blocking ranking gradients through the current dense template while retaining a trainable pooler. Dense future decoding supplies training supervision; decision-time comparisons operate on the compact states. This transfers dense predictive supervision to the small state used in each candidate branch.

C.3 Verifier Ranking and Auxiliary Supervision

Perturbed training alternatives. Perturbed alternatives modify the demonstrated chunk along complementary axes: direction reversal, temporal reordering, cross-example mismatch, magnitude scaling, and inactivity. Gaussian noise in normalized action units adds local variation to the transformed chunks. The default contrastive set uses $R = 4$ alternatives; transformations operate on valid action dimensions while preserving padding and chunk shape. For singleton batches, a within-chunk magnitude perturbation replaces cross-example mismatch. This construction challenges the verifier with changes in action intent and timing without requiring online failure collection for every training example.

Equation 10 operates on raw verifier scores for the expert and perturbed-alternative rollouts, before inference-time bank normalization or native-score fusion. All training rollouts share the observation and instruction, so training compares action-dependent consequences under a common context. The contrastive term distributes preference over the alternatives, while the hinge term penalizes expert–alternative score gaps below γ. Adding a common constant to all scores leaves both terms unchanged: the objective learns relative gaps, with temperature τ_v controlling the contrastive scale and γ setting the raw-score margin. Defaults are $\tau_v = 0.1$, $\gamma = 0.5$, and $\lambda_{\text{rank}} = 1$. The demonstrated chunk is the positive for relative ranking; an alternative's perturbation type does not supply a measured failure label.

Simulator-derived auxiliary supervision. We define the auxiliary objective as the sum of the active label-based terms,

$$\mathcal{L}_{\text{aux}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{contact}} + \mathcal{L}_{\text{collision}} + \mathcal{L}_{\text{stability}} + \mathcal{L}_{\text{relation}} + \mathcal{L}_{\text{object}}. \tag{21}$$

Binary cross entropy applies to task-success, contact, collision-absence, and stability targets, while the four relation outputs use multi-label binary cross entropy. Object-state supervision matches soft groups to simulator objects by nearest center before comparing annotated position, rotation, scale, motion, and semantic fields. These simulator-derived labels are training supervision only and are unavailable to the deployed selector. The active terms are summed in $\mathcal{L}_{\text{aux}}$ and weighted once by $\lambda_{\text{aux}} = 0.1$ in Equation 11. Each loss is evaluated only when its target is present; an absent annotation is omitted rather than encoded as a negative label. The predicted motion fields are center and quaternion-vector-part differences per prediction time point; simulator linear and angular velocity annotations require conversion to this convention before use. Nearest-center matching does not establish a persistent correspondence across time.

C.4 Training Schedule and Optimization

The frozen-policy schedule in Section 3.4 trains consequence prediction and selection while leaving the base policy fixed.

Joint enhancement training. The action expert and its policy-side projections are frozen. The trainable WAVE enhancement optimizes observation-conditioned scene reconstruction, expert-conditioned future prediction, and verifier ranking under Equation 11; available annotations add the corresponding auxiliary terms. The number of compact states remains fixed for a given capacity setting.

Optimization defaults. We use global batch size 8, AdamW, gradient-norm clipping at 1, and a base learning-rate warm-up over 10k updates to 5×10^{-5}. The default coefficients are $\lambda_{\text{world}} = 1$, $\lambda_{\text{compact}} = 0.1$ for compact matching, $\lambda_{\text{ver}} = 0.1$ for the complete verifier objective, and $\lambda_{\text{aux}} = 0.1$ for the auxiliary objective. Within $\mathcal{L}_{\text{ver}}$, the margin penalty uses $\lambda_{\text{rank}} = 1$.

D ADDITIONAL ABLATIONS

Complementary verifier channels. Table 5 tests evidence composition with an end-to-end channel ablation on RoboTwin 2.0. Each checkmark enables a complete channel, including its learned score and structured adjustment. Adding spatial evidence to goal progress raises success by 1.4, 3.2, and 0.4 points for the π_0-, $\pi_{0.5}$-, and Fast-WAM-based configurations; adding contact or plausibility gives further gains, and enabling all four channels reaches 71.5%, 88.1%, and 95.6%. This experiment complements the selection-rule comparison by testing channel composition in closed loop.

Table 5: **Complementary verifier channels.** RoboTwin 2.0 success rates (%) averaged over the two benchmark conditions for three frozen base policies. Contact and plausibility are added separately to the same goal-plus-spatial configuration, then combined in the full verifier.

Verification evidence				RoboTwin 2.0 Success rate ↑		
Goal	Spatial	Contact	Plausibility	π_0-based WAVE	$\pi_{0.5}$-based WAVE	Fast-WAM-based WAVE
✓				63.6	80.2	91.9
✓	✓			65.0	83.4	92.3
✓	✓	✓		70.5	86.5	93.6
✓	✓		✓	69.3	87.1	93.5
✓	✓	✓	✓	**71.5**	**88.1**	**95.6**

Compact representation and 3D prediction fidelity. Table 6 evaluates all representations on RoboTwin 2.0 under the same current-state, action-conditioning, and future-offset protocol. It reports position error, rotation error, relation accuracy, fixed-view PSNR, and LPIPS. Position and rotation errors are computed after matching predicted 3D entities to the target state; relation accuracy measures correct spatial-relation labels; PSNR and LPIPS compare rendered RGB with the target image. The Gaussian representation is strongest on every reported axis, reducing position error from 3.47 to 2.61 cm and rotation error from 9.8 to 7.2 degrees relative to the 3D-VLA head, while increasing relation accuracy from 80.6% to 87.9%, PSNR from 24.9 to 26.7 dB, and reducing LPIPS from 0.138 to 0.104. Table 4 tests whether these prediction differences translate into closed-loop decisions.

Table 6: **Compact representation and 3D prediction fidelity on RoboTwin 2.0.** All rows use the same current-state, action-conditioning, and future-offset protocol. Position error is measured in centimeters, rotation error in degrees, relation accuracy in percent, PSNR in dB, and LPIPS is unitless. Rendered-view metrics use the same fixed camera protocol.

Representation	Position error (cm) ↓	Rotation error (deg.) ↓	Relation acc. ↑	PSNR (dB) ↑	LPIPS ↓
Object-centric tokens	3.18	8.8	83.6	25.2	0.126
Point primitives	2.94	8.5	85.1	24.7	0.130
3D-VLA prediction head	3.47	9.8	80.6	24.9	0.138
3D Gaussian primitives (Ours)	**2.61**	**7.2**	**87.9**	**26.7**	**0.104**

Compact capacity and decision-time trade-off. The capacity study asks how much compact capacity is justified by the resulting decision quality. Each comparison follows the same frozen-policy interface and evaluation protocol as the preceding ablations.

Compare $M \in \{16, 64, 128, 256\}$ at $K = 4$ and $H_w = 5$, retaining the same future offsets, action-feature indices, hidden width, residual parameterization, data, and update budget. Each pooler constructs targets from the same dense fields, while the common candidate bank and native action-generation scores are retained for evaluation. Reducing the slots of a trained model at inference is not a substitute for this capacity study.

Table 7 reports closed-loop success together with wall-clock inference time for each frozen base-policy setting. Within each policy group, time is normalized to the $M = 16$ configuration, which is the $1.00\times$ reference. This exposes the deployment trade-off directly: larger compact states may improve selection while increasing candidate-rollout and verification cost.

Table 7: **Compact capacity and decision-time trade-off.** Each row gives the compact-slot count M. The two groups report closed-loop success averaged over the two RoboTwin 2.0 conditions and wall-clock decision time for the matched $\pi_{0.5}$-based and Fast-WAM-based settings. Time includes the shared scene pass, candidate rollouts, verification, and selection, and is normalized within each group to its own $M = 16$ run ($1.00\times$); $M = 64$ is the default configuration.

M	$\pi_{0.5}$-based / RoboTwin 2.0		Fast-WAM-based / RoboTwin 2.0	
	Success rate (%) ↑	Inference time ↓	Success rate (%) ↑	Inference time ↓
16	77.3	**1.00**×	85.2	**1.00**×
64 (default)	88.1	3.27×	95.6	4.62×
128	89.0	6.98×	96.1	7.30×
256	**89.8**	13.19×	**96.6**	15.12×

We use $M = 64$ as the practical operating point. Relative to $M = 16$, it raises success by 10.8 points for $\pi_{0.5}$ and 10.4 points for Fast-WAM at $3.27\times$ and $4.62\times$ normalized time. Increasing to $M = 128$ adds only 0.9 and 0.5 points while increasing time to $6.98\times$ and $7.30\times$; $M = 256$ adds a further 0.8 and 0.5 points at $13.19\times$ and $15.12\times$. Varying H_w is deferred because it changes action resampling, time-point alignment, and temporal geometric costs; the present study holds those factors fixed.

Training supervision with inference held intact. Table 8 isolates the training signals while retaining the same inference pipeline.

Train each configuration from the same initialization using identical data order, augmentations, schedule, and update budgets. The compact-matching control sets only $\lambda_{\text{compact}} = 0$; dense reconstruction and verifier gradients remain. The margin control sets only $\lambda_{\text{rank}} = 0$, retaining the contrastive ranking term. The auxiliary control sets only $\lambda_{\text{aux}} = 0$, removing the active task, contact, collision, stability, relation, and object-state objectives while retaining reconstruction and verifier ranking. The auxiliary comparison is meaningful only when those annotations are active.

Table 8: **Training-supervision ablation.** The frozen base policy, candidate bank, and full WAVE inference pipeline remain fixed within each benchmark. Each row sets only the indicated coefficient in Equation 11 to zero; all other active objectives retain their defaults. Success rates and invalid-action rates are percentages.

Training objective	$\pi_{0.5}$-based / LIBERO	Cosmos Policy-based / RoboCasa		
	Success rate ↑	Success rate ↑	Invalid-action rate (%) ↓	Steps ↓
Without compact target matching ($\lambda_{\text{compact}} = 0$)	95.9	73.0	8.6	113.5
Without ranking margin ($\lambda_{\text{rank}} = 0$)	96.8	72.6	8.3	112.9
Without simulator-derived auxiliary objectives ($\lambda_{\text{aux}} = 0$)	96.2	73.1	7.9	112.6
Complete objectives (Ours) ($\mathcal{L}_{\text{WAVE}}$)	**98.9**	**79.5**	**7.3**	**106.8**

The complete objective is best on both benchmarks. Relative to it, removing compact target matching lowers LIBERO success by 3.0 points and RoboCasa success by 6.5 points, while increasing RoboCasa invalid-action rate by 1.3 points and *Steps* by 6.7. Removing the ranking margin lowers LIBERO success by 2.1 points and RoboCasa success by 6.9 points, with a 1.0-point increase in invalid-action rate and a 6.1-step increase in *Steps*. Removing auxiliary labels lowers LIBERO success by 2.7 points and RoboCasa success by 6.4 points, with a 0.6-point increase in invalid-action rate and a 5.8-step increase in *Steps*. Compact matching has the largest effect on LIBERO, whereas the ranking margin has the largest effect on RoboCasa; all three supervision paths improve the closed-loop decision quality.

Experimental synthesis. The results form a single progression: WAVE improves frozen policies across all three benchmarks; the fused native and consequence scores account for the selection gain within a fixed candidate bank; candidate count and compact capacity expose the computation frontier; richer future representations improve the evidence used for ranking; and the complete supervision objective yields the strongest closed-loop behavior. Together, the tables distinguish the overall policy enhancement from the computation and training choices that shape it.