TIMEVALUE: SCALING ROBOTIC VALUE FOUNDATION MODELS WITH TEMPORAL DISTANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

General-purpose reward models are increasingly the bottleneck for scaling robot learning, yet the recipe for learning value-related capabilities from large-scale heterogeneous corpora remains underexplored. Existing approaches tie supervision to task-internal anchors such as preferences or normalized progress, none of which transfer cleanly across embodiments and data sources. We introduce **TimeValue**, a value foundation model for robotic manipulation that replaces these anchors with temporal distance, the directed cost-to-go from an observation to the language-specified goal. Because temporal-distance labels can be derived directly from timestamps, TimeValue scales to over 7,000 hours and roughly 3M instruction-conditioned clips without preference or progress annotations. To make temporal-value learning reliable at scale, we combine random temporal sampling, temporal-order shuffling, and value-isolation attention, suppressing shortcuts that would leave predictions insensitive to failures and regressions. Trained without preference labels, TimeValue attains an average Kendall's τ_a of 0.704 on RBM-EVAL-OOD, surpassing the fully preference-supervised state of the art (0.655) and more than doubling a progress-only counterpart (0.292), while generalizing zero-shot to unseen tasks, embodiments, and viewpoints. As a zero-shot reward model, TimeValue serves a range of downstream applications. Converted into dense rewards via potential-based shaping, it raises real-world policy success from 52.5% to 72.5% online and from 63.8% to 82.5% offline; used for data filtering, it improves multi-task behavior cloning success from 35.0% to 42.5%; and applied as inference-time value guidance, it lifts a frozen policy's success from 67.5% to 80.0%. These results establish temporal distance as a scalable supervision target and practical reward interface for generalist robot policies.



Figure 1: **Overview of TimeValue.** TimeValue is a language-conditioned value model trained on over 7,000 hours of heterogeneous embodied data, comprising roughly 3M instruction-conditioned trajectory clips across diverse embodiments, viewpoints, and task domains. Given a language instruction and a sequence of sampled observations, the model builds an interleaved multimodal sequence of repeated absolute-value and relative-value queries, which RynnBrain encodes in a single forward pass. Two distributional heads then predict the absolute temporal distance to task completion and the signed relative temporal displacement between observations, while the language branch produces video analysis and language-conditioned verification. The resulting temporal values serve both as a unified interface for progress estimation, failure detection, and reward specification in reinforcement learning, and as a plug-and-play module for value guidance at inference time.

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.

1 INTRODUCTION

Generalist robot policies are increasingly trained with reinforcement learning [5, 26], yet scaling this loop is bottlenecked not by policy capacity but by reward supervision. Hand-designed rewards hardly generalize to open-ended tasks, while sparse success signals offer limited guidance for long-horizon behavior [58]. Existing general-purpose reward models rely on task-internal anchors such as preferences, reference demonstrations, or local state comparisons [58, 61, 33, 51], which tie supervision to particular trajectories or comparison sets and are difficult to reuse across heterogeneous data. The common fallback, normalized [0,1] progress [31, 30], is an intra-trajectory coordinate rather than a goal-conditioned cost-to-go [48, 47], making it poorly aligned with the standard notion of value in control and difficult to maintain consistently across varying durations, embodiments, and task structures.

We argue for a shift in framing: from a reward model scoring trajectory-level anchors to a **value foundation model** predicting goal-conditioned cost-to-go as a single reusable interface. We instantiate this view in **TimeValue**, which adopts *temporal distance* as its scaling target. Rather than normalizing progress within a trajectory, TimeValue estimates the directed temporal cost from the current observation to the language-specified goal. Under a minimum-time objective, this corresponds to the hitting-time cost-to-go, yielding clear directionality and task conditioning. It can further be converted into dense rewards through potential-based shaping [43], serving as a practical value interface rather than a task-specific heuristic.

Crucially, temporal distance scales naturally to heterogeneous data, since labels can be derived directly from timestamps once a completion cutoff is identified. Using this recipe, we construct a large training mixture of real-world, simulated, and egocentric trajectories spanning diverse embodiments, viewpoints, and task families. After preprocessing, the corpus exceeds 7,000 hours and comprises approximately 3M instruction-conditioned clips. Raw episode boundaries often do not reflect semantic task completion; we therefore apply subtask segmentation and cutoff relabeling to define coherent completion points. Unlike progress and comparison-based supervision [31], this recipe requires only instructions, timestamps, and relabeled cutoffs, without constructing preference pairs or progress annotations.

Scaling data alone, however, does not guarantee reliable temporal-value learning. In a multi-frame setting, the model may exploit regular sampling intervals, the presentation order of observations, or other value-query representations, rather than grounding predictions in task-relevant visual evidence. Such shortcuts can leave learned values insensitive to regressions, failures, and other non-monotonic events [21]. TimeValue therefore incorporates complementary designs at two levels. At the visual level, we sample observations at irregular timestamps and perturb their temporal order during training, breaking the correspondence between sequence position and temporal distance. At the value level, we introduce value-isolation attention, so that each value-query group attends only to its own language-visual context and cannot access the value queries of other observations. Together, these designs suppress both temporal-order and value-extrapolation shortcuts, forcing each prediction to rely on the corresponding visual observation and task semantics. TimeValue further jointly supervises natural-language video description, instruction matching, and success prediction, adding sequence-level semantic cues alongside continuous temporal-value learning.

We evaluate TimeValue both as a standalone value model and as a reward interface for downstream policy learning. Trained without preference labels, TimeValue-8B attains an average Kendall's τ_a of 0.704 on RBM-EVAL-OOD, surpassing the fully preference-supervised state of the art (0.655) and more than doubling a progress-only counterpart (0.292), while generalizing zero-shot to unseen tasks, embodiments, and viewpoints. Converted into dense rewards via potential-based shaping, it raises real-world policy success from 52.5% to 72.5% online and from 63.8% to 82.5% offline over the strongest baseline. Moreover, TimeValue offers two scalable, plug-and-play routes to stronger real-world policies: filtering low-quality segments from demonstrations raises multi-task behavior-cloning success from 35.0% to 42.5%, while reranking candidate action chunks lifts a frozen policy's success from 67.5% to 80.0% without any policy or value-model updates.

In summary, our work makes the following contributions:

- We identify normalized progress as a supervision bottleneck for general-purpose robotic reward modeling, and reframe the problem as learning **a value foundation model** whose scaling target

is **temporal distance (cost-to-go)**, a scalable, preference-free value target that unifies heterogeneous data under a single interface.

- We present **TimeValue**, a robotic value foundation model trained with a label-cheap data recipe: labels are derived directly from timestamps, aided by subtask segmentation and cutoff relabeling, over 7,000 hours and roughly 3M instruction-conditioned clips spanning diverse embodiments, viewpoints, and task families.
- We introduce complementary shortcut-suppression designs, **random temporal sampling**, **temporal-order shuffling**, and **value-isolation attention**, that keep temporal-value predictions grounded in visual evidence rather than spurious correlations, and pair them with dual distributional heads for absolute and relative temporal values.
- We show that TimeValue **surpasses preference-supervised state-of-the-art** on both out-of-distribution trajectory ranking and instruction-trajectory alignment without any preference annotation, and serves as a practical value interface for downstream policy learning, improving real-world reinforcement learning via potential-based shaping, multi-task behavior cloning via data filtering, and frozen policies via inference-time value guidance.

2 Related Work

Reward learning for robotic manipulation. Reward learning infers task objectives from demonstrations, preferences, or visual and temporal patterns, reducing the need for manual reward design. Inverse reinforcement learning recovers rewards from expert behavior [42, 1, 63] through joint cost-and-policy learning [19] or adversarial distribution matching [23, 20], whereas preference-based methods use trajectory comparisons or ranked demonstrations [12, 7]. Visual approaches support reward construction through temporal correspondence [49], video representations [40, 35], and language–image alignment [34]. Moving from representations to reward signals, video-based methods derive rewards from human videos [10], progress ranking [57], language-guided transfer [60], frame-wise temporal distance [33], or video-language contrastive learning [50]. Despite these advances, their applicability to diverse robotic scenarios remains a challenge. TimeValue aims to address this gap by learning a temporal value function that can generalize across tasks, embodiments, and viewpoints, providing a more versatile and reusable reward signal for robotic manipulation.

Robotic value foundation models. Recent efforts aim to provide reusable reward and value signals across robotic tasks, using trajectory-level comparisons or step-level process supervision. RoboReward [30] learns trajectory scores from annotated robotic corpora, while Robometer [31] learns progress through trajectory comparisons. Step-level methods provide finer-grained feedback via normalized progress regression [51, 58, 61, 36] or advantage reward modeling [37]. TOPReward [11] instead extracts progress rewards from pretrained vision-language models' task-completion token probabilities without reward-specific training. A central challenge is to achieve both scalability and strong performance: RoboReward and Robometer perform strongly but require large-scale annotations, whereas self-supervised methods derive progress targets from within-trajectory timestamps but remain limited on unseen tasks. TimeValue addresses this tradeoff by using temporal distance as a scalable, self-supervised target across heterogeneous trajectories. It further identifies and mitigates temporal-value shortcuts through a training recipe designed for robust value learning at scale.

Temporal distance learning. Temporal distance measures the time or transitions needed to reach a goal, giving goal-conditioned values a temporal interpretation. Goal-conditioned reinforcement learning [28] supports this formulation through universal value function approximators [48] and hindsight goal relabeling [2]. Temporal-difference models learn distances to future observations [46], and learned distances support planning via replay-buffer graph search [16]. Transferable value representations include successor representations [14], successor features with generalized policy improvement [4], and universal successor features approximators [6]. Contrastive learning connects representation inner products to goal-conditioned values [17], quasimetric reinforcement learning captures asymmetric goal-reaching costs [54], and contrastive successor features structure temporal distances for trajectory stitching [39]. Translating them into a general-purpose robotic value foundation model requires handling heterogeneous robotic data and grounding temporal predictions in task semantics. TimeValue bridges this gap, providing the first robotic value foundation model that learns from temporal distance.

3 METHOD

As shown in Figure 1, TimeValue is a language-conditioned value model that estimates the temporal distance from robot observations to task completion. TimeValue is built on RynnBrain [13] and further pretrained on large-scale robot data, which provides representations adapted to embodied scenes, robot states, and language-conditioned manipulation tasks.

3.1 TIMEVALUE DATASET

TimeValue unifies heterogeneous trajectories into state-level temporal-distance supervision. Our mixture exceeds 7,000 hours of robot, simulation, and egocentric trajectories [8, 24, 27, 45], covering single-arm, dual-arm mobile, bimanual tabletop, and dexterous-hand systems, simulated embodiments, and first-person human demonstrations. It spans diverse viewpoints, control frequencies, execution speeds, task durations, objects, and language annotations. Subtask segmentation and cutoff-based relabeling convert 1.67M original episodes into over 3M instruction-conditioned segments, each with a language goal and a completion cutoff. To accommodate varying episode and subtask definitions [8, 45, 9], we split demonstrations using native temporal annotations where available and otherwise retain complete episodes. Each segment's endpoint serves as its default completion cutoff; dataset-specific ratio- or duration-based trimming removes post-completion motions when needed to approximate the first semantically completed observation [31]. Timestamp-derived labels measure temporal distance to the cutoff and are zero at or after it, providing dense supervision without dataset-specific progress normalization. Separately, Qwen3-VL-27B [3] generates segment-level descriptions of observed behavior and task-relevant events. These supervise `Video Description` in Equation 2 without affecting temporal targets.

3.2 MODEL ARCHITECTURE

Multi-granularity value estimation. Given a text prompt ℓ comprising the task instruction and system prompts for absolute and relative temporal estimation, embodiment metadata m, and a sequence of K observations $\mathcal{I} = \{I_{t_i}\}_{i=1}^{K}$, TimeValue jointly estimates the temporal distance to task completion for each observation and the signed temporal displacement between adjacent observations in the presented sequence. For each observation I_{t_i}, we insert an absolute-value query group $\mathbf{V}_i = [V_{i,1}, \ldots, V_{i,N}]$ consisting of repeated <value> tokens. For $i \geq 2$, we also insert a relative-value query group $\mathbf{R}_{i-1} = [R_{i-1,1}, \ldots, R_{i-1,N}]$ of repeated <relative_value> tokens before $\mathbf{V}_i$ to estimate the displacement from $I_{t_{i-1}}$ to I_{t_i}. Following [25], we use N repeated query tokens for each temporal prediction to alleviate the information bottleneck of a single token and support a richer temporal representation. Our main setting uses $N = 8$ tokens per query group and $K = 8$ observations sampled from each video. After the final temporal-query group, we append a natural-language verification prompt $\mathbf{p}_{\text{ver}}$, yielding the input sequence:

$$\mathbf{x} = \big[m, \ell,\, I_{t_1}, \mathbf{V}_1,\, I_{t_2}, \mathbf{R}_1, \mathbf{V}_2,\, \ldots,\, I_{t_K}, \mathbf{R}_{K-1}, \mathbf{V}_K,\, \mathbf{p}_{\text{ver}}\big]. \tag{1}$$

The verification prompt instructs the model to first analyze the video in natural language, then determine whether the observed behavior matches the task instruction and whether the task is successfully completed:

$$\mathbf{y}_{\text{lang}} = \big[\texttt{Video Description:}\quad \mathbf{y}^{\text{vid}}, \texttt{Match:}\quad \mathbf{y}^{\text{match}}, \texttt{Success:}\quad \mathbf{y}^{\text{succ}}\big]. \tag{2}$$

This autoregressive ordering encourages the model to describe robot behavior, object interactions, and task-relevant events, including evidence of completion, before producing the final task-matching and task-success judgments.

Distributional temporal readouts. The N tokens in each temporal-query group attend bidirectionally. We concatenate their hidden states to retain complementary information across positions:

$$\widetilde{\mathbf{h}}_i^V = H_\theta(\mathbf{x})_{\text{pos}(V_{i,1})} \| \cdots \| H_\theta(\mathbf{x})_{\text{pos}(V_{i,N})}, \qquad \widetilde{\mathbf{h}}_i^R = H_\theta(\mathbf{x})_{\text{pos}(R_{i,1})} \| \cdots \| H_\theta(\mathbf{x})_{\text{pos}(R_{i,N})}, \tag{3}$$

where H_θ denotes RynnBrain contextual representations and $\|$ denotes feature concatenation. With per-token hidden size d, two heads map the Nd-dimensional representations to the bin logits:

$$\mathbf{z}_i^V = \text{ValueHead}\left(\widetilde{\mathbf{h}}_i^V\right), \qquad \mathbf{z}_i^R = \text{RelativeHead}\left(\widetilde{\mathbf{h}}_i^R\right). \tag{4}$$

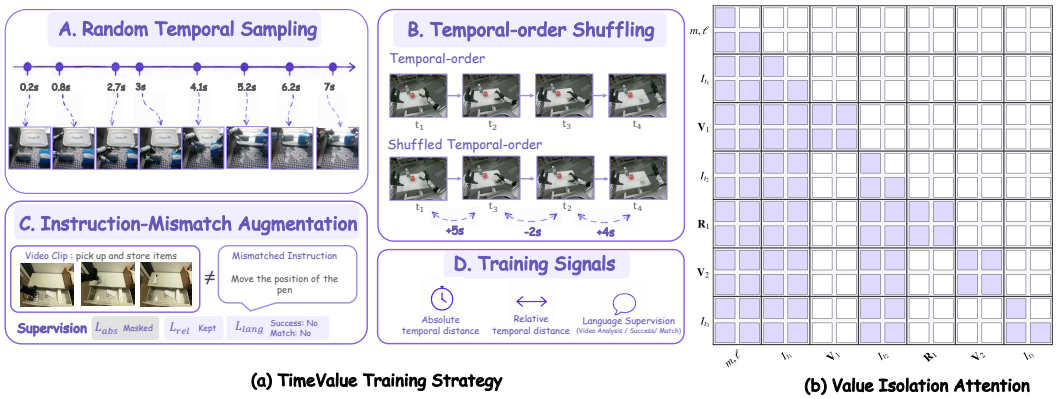

Figure 2: TimeValue training pipeline and value-isolation attention. **(a)** Training strategy. Random temporal sampling and temporal-order shuffling suppress shortcuts tied to sampling intervals and sequence position, while instruction-mismatch augmentation strengthens language–visual grounding. TimeValue jointly learns absolute temporal distance, relative temporal displacement, and natural-language supervision. **(b)** Value-isolation attention. Within each value-query group, repeated queries attend to one another and to the language–visual context, while remaining isolated from other value-query groups. Colored cells denote visible attention connections.

The heads estimate temporal distance to the relabeled completion cutoff and signed temporal displacement between adjacent observations in the presented sequence, providing global and local temporal supervision, respectively.

Absolute targets span $[0, 636.3]$ seconds and concentrate near zero with a long right tail. A fixed quantile support provides finer resolution near zero while covering the tail: 256 bin centers sample the inverse empirical cumulative distribution function (CDF) of temporal targets at equally spaced probability levels. Mirroring the non-negative quantile knots around zero gives a symmetric relative support over $[-636.3, 636.3]$ seconds. Following Dreamerv3 [22], we encode continuous targets as two-hot distributions over adjacent bin centers and train with cross-entropy, avoiding gradients that scale directly with temporal error. Inference takes the expected bin centers [18]:

$$v_i = \sum_{b=1}^{|\mathcal{B}_V|} c_b^V \ \mathrm{softmax}(\mathbf{z}_i^V)_b, \qquad \Delta_i = \sum_{b=1}^{|\mathcal{B}_R|} c_b^R \ \mathrm{softmax}(\mathbf{z}_i^R)_b, \tag{5}$$

where $\mathcal{B}_V$ and $\mathcal{B}_R$ are the absolute and relative bin sets, and c_b^V and c_b^R are their centers in seconds.

3.3 Shortcut Learning

Scaling data alone does not guarantee reliable temporal-value learning, because a multi-frame model can exploit regularities in the input rather than the visual evidence. We combine random temporal sampling, temporal-order shuffling and value-isolated attention to suppress shortcuts tied to temporal intervals, sequence positions, and value extrapolation.

Random temporal sampling. For each instruction-conditioned trajectory clip, we randomly sample $K = 8$ observations at irregular timestamps. The resulting non-uniform temporal gaps break the near-arithmetic value patterns induced by uniform sampling and discourage the model from exploiting fixed sampling intervals as a shortcut.

Temporal-order shuffling. As illustrated in Figure 2(a), we shuffle the chronological order of sampled observations. Half of the training sequences are independently sampled without temporal sorting, while the remainder follow a forward-biased temporal walk with occasional backward transitions, whose per-step probability we refer to as the rewind probability. Relative temporal targets are computed between adjacent observations in the presented order and can therefore be either positive or negative. Shuffling prevents the model from regressing a stereotypical value curve from sequence positions.

Value-isolation attention. Without an explicit attention constraint, a temporal query can predict its value by extrapolating from previously exposed value tokens rather than by interpreting the corresponding visual evidence. This shortcut produces smooth value curves while leaving predictions insensitive to regressions, failures, and other non-monotonic events. Inspired by attention masking strategies that explicitly control information flow and prevent target leakage [59, 15], we introduce value-isolation attention, illustrated in Figure 2(b). Absolute and relative temporal queries serve as pooling tokens that aggregate information from the preceding language–visual context. Within each temporal-query group, tokens attend to one another to integrate complementary features before concatenation, while remaining isolated from queries in all other groups. Subsequent observation tokens are also prevented from attending to temporal queries, blocking indirect information leakage to later temporal-query groups through the observation representations.

3.4 Instruction-mismatch augmentation.

A value model should also recognize when the language instruction does not describe the observed video, rather than always reporting smooth progress toward the stated goal. As shown in Figure 2, we therefore introduce instruction-mismatch augmentation. For 10% of the training samples, we replace the original instruction with one sampled from a different trajectory and supervise the language branch to predict `Match: No` and `Success: No`. Because the original completion cutoff is no longer valid under the substituted instruction, we mask the absolute temporal-distance loss (Equation 6) to avoid introducing incorrect supervision. The relative temporal loss (Equation 7) is retained, as the relative target measures the temporal displacement between observations and is independent of the task instruction. This augmentation teaches TimeValue to detect instruction-video mismatches and remain reliable under substantial language-visual inconsistency.

3.5 Joint Training Objectives

TimeValue is jointly optimized using three cross-entropy objectives: an absolute temporal-distance loss, a relative temporal-distance loss, and a causal language-modeling loss. Although all three objectives use cross-entropy, the first two operate over distributional temporal bins, whereas the third operates over the language vocabulary.

Temporal-distance objectives. For observation I_{t_i} at timestamp t_i and relabeled completion cutoff t_G, the absolute target $v_i^\star = \max(0, t_G - t_i)$ anchors supervision to task completion. The relative target $\Delta_i^\star = t_{i+1} - t_i$ captures temporal displacement between adjacent observations in the presented multimodal sequence; positive and negative values indicate progress and regression, respectively. Therefore, the absolute and relative objectives are defined as:

$$\mathcal{L}_{\text{abs}} = -\frac{\omega}{K} \sum_{i=1}^{K} \sum_{b=1}^{|\mathcal{B}_V|} \left[\text{TwoHot}_{\mathcal{B}_V}(v_i^\star)\right]_b \log \left[\text{softmax}(\mathbf{z}_i^V)\right]_b . \tag{6}$$

$$\mathcal{L}_{\text{rel}} = -\frac{1}{K-1} \sum_{i=1}^{K-1} \sum_{b=1}^{|\mathcal{B}_R|} \left[\text{TwoHot}_{\mathcal{B}_R}(\Delta_i^\star)\right]_b \log \left[\text{softmax}(\mathbf{z}_i^R)\right]_b . \tag{7}$$

where $\mathbf{z}_i^V$ and $\mathbf{z}_i^R$ are the absolute- and relative-value head logits, decoded into continuous predictions v_i and Δ_i, respectively. The sample-level mask ω is 1 for instruction-matched examples and 0 for mismatches, which invalidate the original completion cutoff.

Natural-language objective. For natural-language supervision, TimeValue autoregressively predicts the structured output defined in Equation 2 and is optimized using standard token-level cross-entropy loss:

$$\mathcal{L}_{\text{lang}} = -\frac{1}{|\mathcal{Y}|} \sum_{j \in \mathcal{Y}} \log p_\theta \left(y_j \mid \mathbf{x}, y_{<j}\right), \tag{8}$$

where $\mathcal{Y}$ denotes the supervised language-token positions. Success prediction is therefore learned through the natural-language objective, without introducing a separate success-classification head.

Joint objective. The final training objective combines the three cross-entropy losses:

$$\mathcal{L} = \mathcal{L}_{\text{abs}} + \mathcal{L}_{\text{rel}} + \lambda_{\text{lang}} \mathcal{L}_{\text{lang}}, \qquad \lambda_{\text{lang}} = 2. \tag{9}$$

Table 1: Per-dataset trajectory-ranking results on the RBM-EVAL-OOD test suite, measured by Kendall's τ_a (↑). Bold values indicate the best overall results. † denotes the best result among methods trained without explicit trajectory-level preference supervision, i.e., progress/value-only methods. Baseline results are taken from Robometer [31].

Method	USC Franka	USC Koch	USC Trossen	USC xArm	MIT Franka	UTD SO101	Average
GVL [36]	0.250	−0.008	0.292	0.056	0.306	0.300	0.199
VLAC-2B [61]	0.292	0.167	−0.111	0.167	−0.017	−0.033	0.077
VLAC-8B [61]	0.271	0.064	−0.417	0.139	0.072	0.167	0.049
RoboDopamine [51]	0.167	0.175	0.000	0.014	0.220	0.067	0.107
Dopamine-GRM-2.0-8B-Preview [51]	0.479	0.442	0.333	0.431	0.431	0.700	0.453
RoboReward-4B [30]	0.625	0.332	0.333	0.528	0.494	0.700	0.502
RoboReward-8B [30]	0.625	0.264	0.389	0.347	0.396	0.767	0.465
Robometer (RoboReward data) [31]	0.583	0.533	0.646	0.403	0.479	0.667	0.552
ReWiND [60]	−0.125	0.336	0.028	−0.167	0.080	−0.067	0.014
Robometer (RBM-1M) [31]	0.646	0.471	0.653	0.694	**0.601**	**0.867**	0.655
Robometer (Progress only) [31]	0.083	0.231	0.333	0.389	0.183	0.533	0.292
TimeValue-4B	0.542	**0.645**†	**1.000**†	0.500	0.518†	**0.867**†	0.679
TimeValue-8B	**0.750**†	0.504	**1.000**†	**0.722**†	0.450	0.800	**0.704**†

The absolute and relative losses update their corresponding distributional heads and the visual-language backbone. More details regarding datasets, training hyperparameters, optimization, inference are provided in Section B.

4 Benchmark Evaluation

We evaluate TimeValue on the RBM-EVAL-OOD test suite introduced by Robometer [31]. The benchmark contains 976 trajectories from six out-of-distribution datasets spanning different institutions, robot embodiments, camera viewpoints, and task families. For each task, the benchmark provides trajectories annotated with different execution-quality levels, including failed, suboptimal, and successful executions. Performance is measured by Kendall's τ_a between the ground-truth trajectory-quality ordering and the ordering induced by the predicted scores. We describe the baselines in Section C.1 and provide additional reward evaluation details in Section C.

Main results. As shown in Table 1, TimeValue-8B achieves the highest average Kendall's τ_a of 0.704, and even the smaller TimeValue-4B reaches 0.679; both exceed the 0.655 obtained by the full Robometer model trained with both progress and trajectory-preference supervision. In terms of the best overall results, TimeValue-8B ranks first on USC Franka and USC xArm. TimeValue-4B ranks first on USC Koch, ties TimeValue-8B at 1.000 on USC Trossen, and ties Robometer (RBM-1M) at 0.867 on UTD SO101. Among methods trained without explicit trajectory-level preference supervision, TimeValue variants attain the best results on all six datasets, improving the strongest preference-free prior average of 0.502 to 0.679 at 4B and 0.704 at 8B. TimeValue also substantially outperforms the progress-only Robometer ablation, which achieves an average score of 0.292. Notably, performance is already strong at 4B and improves only marginally at 8B, suggesting that the gains stem from the temporal-distance formulation and its accompanying training and architectural designs rather than sheer model scale.

Ablation study. We ablate the principal training and architectural components of TimeValue in Table 2; all variants are evaluated at the same training checkpoint. Further ablations in Section C.2 compare quantile, fixed-bin, and normalized value tokenizers, as well as inference-time attention mask. The full model achieves the highest average Kendall's τ_a of 0.704. Removing temporal-order shuffling causes the largest degradation, reducing the average score to 0.159—even turning negative on MIT Franka—as sequence position again becomes a strong proxy for task progress. Replacing random temporal sampling with uniform sampling reduces the average score to 0.207. The regular intervals it produces let the model exploit a stereotypical value pattern rather than reason from the observed visual content. Beyond sampling, replacing value-isolation attention with standard eager attention during training decreases performance to 0.611, confirming that isolation prevents prediction queries from extrapolating values across query groups. Removing the auxiliary natural-language supervision reduces the average score to 0.622, demonstrating that video-description, instruction-matching, and success supervision provide useful semantic grounding. Fi-

Table 2: **Ablation study on RBM-EVAL-OOD.** We report Kendall's τ_a across six out-of-distribution robot datasets; all variants are evaluated at the same training checkpoint. Shuffle denotes temporal-order shuffling, Isolation denotes training-time value-isolation attention, Language denotes the auxiliary natural-language supervision, Random denotes random temporal sampling, and Relative denotes the relative modeling component.

Variant	Design Components					Kendall's τ_a						
	Shuffle	Isolation	Language	Random	Relative	USC Franka	USC Koch	USC Trossen	USC xArm	MIT Franka	UTD SO101	Average
w/o Shuffle	✗	✓	✓	✓	✓	0.500	0.037	0.111	0.111	−0.104	0.300	0.159
w/o Isolation	✓	✗	✓	✓	✓	0.708	0.472	0.778	0.667	0.476	0.567	0.611
w/o Language	✓	✓	✗	✓	✓	0.625	0.504	0.917	0.472	**0.483**	0.733	0.622
Uniform Sampling	✓	✓	✓	✗	✓	0.208	0.104	0.111	0.056	0.093	0.667	0.207
w/o Relative	✓	✓	✓	✓	✗	0.542	**0.584**	0.542	**0.778**	0.420	0.600	0.578
Full Model (8B)	✓	✓	✓	✓	✓	**0.750**	0.504	**1.000**	0.722	0.450	**0.800**	**0.704**

nally, removing relative temporal-distance supervision yields a score of 0.578: by modeling local forward and backward temporal changes, this objective complements the globally anchored absolute prediction and improves the shared visual representation.

5 Real-world Policy Learning

Beyond benchmarks, we evaluate whether TimeValue can improve policy learning on a diverse set of challenging real-world manipulation tasks. Since none of these tasks, their manipulated objects, or the workspace scenes appear in the reward-model training corpus, and no target-domain fine-tuning is performed, TimeValue serves as a zero-shot reward annotator. This setting assesses its open-world generalization and its effectiveness for value estimation under physical execution noise. In Sections D and E, we provide additional details on the robotic platform, tasks, datasets, policy backbones, and policy-learning algorithms.

Performance analysis. As shown in Table 3, TimeValue consistently achieves the highest success rate across all four real-world tasks in both online and offline RL. In online OOD evaluation, it reaches an average success rate of 72.5%, substantially outperforming Robometer at 52.5% and sparse rewards at 48.8%. The improvement is particularly pronounced on *Steak Serving with a Spatula* and *Bimanual Box Transfer*, where TimeValue improves upon Robometer by 30 and 35 percentage points, respectively. Under offline RL, TimeValue achieves an average success rate of 82.5%, compared with 63.8% for Robometer and only 23.8% for plain SFT.

TimeValue also provides a strong balance between task success and execution efficiency. On *Bread Basket Placement*, it achieves 100% offline success using 16.8 action chunks on average, compared with 80% and 18.9 chunks for Robometer and 30% and 30.2 chunks for multi-task SFT. Similar improvements are observed on *Steak Serving with a Spatula* and *Box-in-Drawer Placement*, where TimeValue reaches 90% success while requiring only 14.9 chunks. Notably, reinforcement learning with TimeValue successfully solves *Bimanual Box Transfer*, for which the multi-task SFT policy records no successful executions, and lifts *Box-in-Drawer Placement* from 40% to 90% offline.

Algorithm	Baseline	Bread Basket Placement		Steak Serving with a Spatula		Box-in-Drawer Placement		Bimanual Box Transfer		Average Success
		Success ↑	Avg. Steps ↓	Success ↑	Avg. Steps ↓	Success ↑	Avg. Steps ↓	Success ↑	Avg. Steps ↓	
Online RL	**TimeValue**	**45.0%**	25.9 ± 8.2	**75.0%**	18.6 ± 13.1	**70.0%**	$\mathbf{27.0 \pm 8.1}$	**100.0%**	$\mathbf{22.8 \pm 4.7}$	**72.5%**
	Robometer	35.0%	$\mathbf{22.7 \pm 5.5}$	45.0%	$\mathbf{15.2 \pm 2.7}$	65.0%	27.7 ± 5.8	65.0%	25.6 ± 7.4	52.5%
	Sparse	40.0%	56.0 ± 31.7	45.0%	18.4 ± 4.9	40.0%	27.4 ± 6.6	70.0%	23.5 ± 2.7	48.8%
Offline RL	**TimeValue**	**100.0%**	$\mathbf{16.8 \pm 3.1}$	**90.0%**	$\mathbf{14.9 \pm 4.0}$	**90.0%**	$\mathbf{14.9 \pm 4.0}$	**50.0%**	33.6 ± 10.5	**82.5%**
	Robometer	80.0%	18.9 ± 2.7	80.0%	19.4 ± 7.2	50.0%	27.3 ± 6.3	45.0%	$\mathbf{28.7 \pm 9.3}$	63.8%
	Sparse	70.0%	26.1 ± 9.1	20.0%	30.2 ± 3.3	0.0%	–	0.0%	–	22.5%
SFT	–	70.0%	24.8 ± 8.0	25.0%	18.6 ± 6.2	0.0%	–	0.0%	–	23.8%

Table 3: **Real-world reinforcement-learning results.** We report success rates and the average number of action chunks over successful episodes. Average denotes the unweighted mean success rate across four tasks.

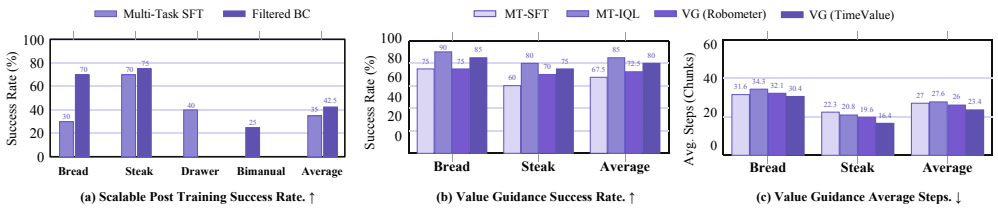

Figure 3: **Multi-task data filtering and value guidance results.** (a) Data filtering success rates; missing bars denote tasks with zero successful episodes. (b–c) Value guidance success rates and the average number of action chunks over successful episodes, with multi-task BC and multi-task IQL as references.

These results show that TimeValue enables substantial improvements over the imitation-learning baseline while maintaining efficient task completion.

Scalable post-training. While offline and online RL yield our strongest policies, both require a separate policy-training run for each task. We therefore evaluate a single jointly trained multi-task policy as a more scalable alternative. In Figure 3(a), multi-task SFT and Filtered BC share the same policy initialization, source dataset, and optimization recipe; only Filtered BC removes segments scored as low quality by TimeValue (please refer to Section E.6 for implementation details), isolating the effect of value-based selection. The filtering raises the average success rate from 35.0% to 42.5%, with the largest gains on Bread ($30\% \rightarrow 70\%$) and Bimanual ($0\% \rightarrow 25\%$), and a modest improvement on Steak ($70\% \rightarrow 75\%$). However, Drawer drops from 40% to 0%, and we attribute this degradation to the limited ability of third-person RGB observations to capture task-relevant details, such as grasp stability and box-drawer alignment, which causes useful segments to be assigned low-quality scores and subsequently discarded. Overall, without separate task-specific fine-tuning, TimeValue-based filtering obtains a higher average success rate than unfiltered multi-task SFT in a single joint training run. TimeValue thus supports both reward shaping and scalable data curation for post-training generalist robot policies.

Inference-time value guidance. Inference-time value guidance ranks candidate action chunks from a frozen policy using predicted values. We instantiate this approach with Cosmos3-Edge-2B [44] (Section E.1), rescoring frozen multi-task BC (MT-SFT) proposals with Robometer or TimeValue to obtain VG (ROBOMETER) and VG (TIMEVALUE). A trained multi-task IQL policy (MT-IQL) serves as our upper-bound reference. Figure 3(b–c) supports three observations. First, changing only the scorer from Robometer to TimeValue raises average success from 72.5% to 80.0% and shortens successful episodes from 26.0 to 23.4 action chunks on average. Second, gains over unguided MT-SFT reach 12.5 percentage points with TimeValue versus 5.0 with Robometer: with the candidate budget fixed, the benefit of best-of-N selection hinges on scorer fidelity. Finally, VG (TIMEVALUE) comes within 5 points of MT-IQL on both tasks without updating either the policy or the value model. These results show that TimeValue provides an effective ranking signal for inference-time action selection, approaching the performance of a policy that requires additional training.

6 CONCLUSION AND FUTURE WORKS

We presented TimeValue, a robotic value foundation model trained with timestamp-derived, goal-conditioned temporal distance, without preference or dataset-specific progress labels. Random temporal sampling, order shuffling, and value-isolation attention reduce shortcuts and improve sensitivity to failures and regressions. TimeValue outperforms preference-supervised and progress-only baselines on RBM-EVAL-OOD, more than doubles a progress-only counterpart, and generalizes zero-shot across unseen tasks, embodiments, and viewpoints. As a reusable value interface, it improves online and offline real-world policy learning through potential-based shaping, supports multi-task post-training through value-based data filtering, and guides inference-time candidate selection without policy updates. Future work will support longer horizons and streaming inference, incorporate energy, safety, and precision costs, and extend to dexterous and mobile manipulation.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to clean and reformat datasets. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for proofs, write proofs, propose or refine hypotheses, design or give feedback on research methodology or experiments, assist with translation, support qualitative or thematic data analysis, or interpret results. Additionally, we used generative AI tools to suggest the paper structure, title, and keywords, and to edit the manuscript for readability. We have reviewed all AI-assisted work, including the implemented methods, the cleaned and reformatted data, and the edited text. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2004.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Andre Barreto, Will Dabney, Remi Munos, Jonathan J. Hunt, Tom Schaul, Hado P. van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 4055–4065, 2017.

[5] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. In *Proceedings of the Conference on Robot Learning*, 2025.

[6] Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Remi Munos, Hado P. van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. In *Proceedings of the International Conference on Learning Representations*, 2019.

[7] Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the International Conference on Machine Learning*, 2019.

[8] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. AgiBot World Colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

[9] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in PyTorch. `https://github.com/huggingface/lerobot`, 2024.

[10] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from in-the-wild human videos. In *Robotics: Science and Systems*, 2021.

[11] Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. `https://topreward.github.io/webpage/`, 2026. Project page.

[12] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

[13] Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, et al. RynnBrain: Open embodied foundation models. *arXiv preprint arXiv:2602.14979*, 2026.

[14] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[15] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[16] Benjamin Eysenbach, Ruslan R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[17] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2022.

[18] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep RL. In *Proceedings of the International Conference on Machine Learning*, pp. 13049–13071, 2024.

[19] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the International Conference on Machine Learning*, 2016.

[20] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2018.

[21] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[22] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.

[23] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2016.

[24] Ryan Hoque, Peide Huang, David J Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[25] Bohan Hou, Haoqiang Lin, Xuemeng Song, Haokun Wen, Meng Liu, Yupeng Hu, and Xiangyu Zhao. FiRE: Enhancing MLLMs with fine-grained context learning for complex image retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 803–812, 2025.

[26] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[27] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025.

[28] Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, pp. 1094–1099, 1993.

[29] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *Proceedings of the International Conference on Learning Representations*, 2022.

[30] Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

[31] Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

[32] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. In *Proceedings of the International Conference on Learning Representations*, pp. 29982–30009, 2025.

[33] Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. TimeRewarder: Learning dense reward from passive videos via frame-wise temporal distance. *arXiv preprint arXiv:2509.26627*, 2025.

[34] Yecheng Jason Ma, William Liang, Guanxi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. LIV: Language-image representations and rewards for robotic control. In *Proceedings of the International Conference on Machine Learning*, 2023.

[35] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *Proceedings of the International Conference on Learning Representations*, 2023.

[36] Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *Proceedings of the International Conference on Learning Representations*, 2025.

[37] Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. Arm: Advantage reward modeling for long-horizon manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4468–4477, 2026.

[38] Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27649–27660, 2025.

[39] Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the International Conference on Machine Learning*, pp. 37076–37096, 2024.

[40] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Proceedings of the Conference on Robot Learning*, 2022.

[41] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Mił oś, and Marek Cygan. Bigger, Regularized, Optimistic: scaling for compute and sample efficient continuous control. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 113038–113071, 2024.

[42] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

[43] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, pp. 278–287, 1999.

[44] NVIDIA. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026. URL `https://research.nvidia.com/labs/cosmos-lab/cosmos3/technical-report.pdf`.

[45] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, and et.al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 6892–6903, 2023.

[46] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. In *Proceedings of the International Conference on Learning Representations*, 2018.

[47] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[48] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the International Conference on Machine Learning*, pp. 1312–1320, 2015.

[49] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[50] Sumedh A. Sontakke, Jesse Zhang, Sebastien M. R. Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 55681–55693, 2023.

[51] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-Dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.

[52] Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. InternData-A1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

[53] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the Conference on Robot Learning*, 2025.

[54] Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proceedings of the International Conference on Machine Learning*, pp. 36411–36430, 2023.

[55] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[56] Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

[57] Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2Reward: Learning shaped reward functions from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2806–2813. IEEE, 2024.

[58] Pengzhi Yang, Xinyu Wang, Pengyu Jing, Kehan Wen, Yiduo Qu, Zhenhao Huang, Minghao Fu, Xin Liu, Yaheng Shen, and Fan Shi. RARM: Confidence-gated progress reward modeling for RL in manipulation. *arXiv preprint arXiv:2606.22027*, 2026.

[59] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[60] Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. In *Proceedings of the Conference on Robot Learning*, 2025.

[61] Qi Zhang, Shaopeng Zhai, Shengzhe Zhang, Litao Liu, Tianyi Zhang, Fuxian Huang, and Ming Zhou. A generalist pair-wise progress critic model for vision-language-action robots. In *Proceedings of the International Conference on Machine Learning*, 2026.

[62] 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. *arXiv preprint arXiv:2510.10274*, 2025.

[63] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.

A TIMEVALUE DATASET

Learning temporal distance requires more than a large collection of robot videos: each trajectory segment must be associated with a meaningful language goal and must describe task-relevant evolution toward that goal. Raw robot corpora do not always satisfy these conditions. Their annotations may contain placeholders, data-quality tags, truncated commands, or descriptions of pure robot motion. We therefore apply a source-aware curation pipeline to Open X-Embodiment (OXE), InternData-A1 (InternA1), Galaxea Open-World (Galaxea), and RoboCOIN before constructing temporal-distance targets. Since OXE and InternA1 are counted in episodes, whereas Galaxea and RoboCOIN are counted in annotated subtask segments, we refer to their entries collectively as *trajectory units*. Therefore, we describe how we curate and segment these four sources, and then summarize the composition of the entire heterogeneous mixture in Table 4.

Table 4: Composition of the heterogeneous data mixture. # Episodes counts the original trajectories before subtask expansion, while # Segments counts the instruction-conditioned trajectory segments obtained after subtask segmentation and cutoff relabeling. Overall, 1.67M original episodes are expanded into 3.09M trajectory segments. For sources without subtask annotations (segmentation source: coarse task), no splitting is applied and each episode is kept as a single segment.

Data Source	# Episodes	# Segments	# Instructions	Segmentation Source
AgiBot [8]	167,535	1,166,042	3,741	sub task
EgoDex [24]	338,234	338,234	2,038	coarse task
Galaxea Open-World [27]	16,979	95,671	11,070	sub task
InternData-A1 [52]	320,905	320,905	348	coarse task
Open X-Embodiment [45]	693,037	693,037	180,090	coarse task
RDT [32]	6,109	6,109	272	coarse task
RoboCOIN [56]	67,420	410,877	2,124	sub task
RoboMIND [55]	32,138	32,138	184	coarse task
RoboTwin [38]	27,414	27,414	23,527	coarse
Soft-FOLD [62]	1,542	1,542	1	coarse task
Total	**1,671,313**	**3,091,969**	**223,395**	–

Heterogeneous data mixture. To scale temporal-distance learning beyond curated task-specific demonstrations, we train TimeValue on a large heterogeneous mixture of robot, simulation, and egocentric data [8, 24, 27, 45]. The mixture exceeds 7,000 hours of trajectories and spans a broad range of manipulation settings, including single-arm platforms, dual-arm mobile manipulators, bimanual tabletop robots, dexterous-hand systems, simulated embodiments, and first-person human demonstrations. It also covers diverse camera configurations. This diversity exposes the model to large variations in morphology, viewpoint, control frequency, execution speed, task duration, object distribution, and language annotation granularity.

Before subtask expansion, the corpus contains 1.67M original episodes. After subtask segmentation and cutoff-based relabeling, these data are further converted into over 3M instruction-conditioned trajectory segments, each associated with a language goal and a temporally localized completion target. Crucially, these sources are not unified by manually defining dataset-specific progress scales. Instead, they are mapped onto the same temporal-distance supervision interface: each trajectory segment provides timestamp-derived state-level labels measuring the remaining temporal distance to its completion cutoff.

Data preprocessing and temporal-distance relabeling. Large-scale robot datasets differ substantially in how episodes and subtasks are defined [8, 45, 9]. We therefore convert each data source into instruction-conditioned trajectory segments. Long demonstrations are split using native temporal annotations when available; otherwise, the complete episode is retained as a coarse segment. Because recorded trajectories may contain post-completion motions, we further assign each segment a completion cutoff. The segment endpoint is used by default, while dataset-specific ratio- or duration-based trimming is applied when necessary to better approximate the first semantically completed observation [31].

Temporal-distance labels are then generated directly from timestamps: observations before the cutoff are labeled by their temporal distance to completion, while observations at or after the cutoff

receive zero. This provides unified, dense supervision without dataset-specific progress normalization. Separately, we use Qwen3-VL-27B [3] to generate a segment-level description of the observed behavior and task-relevant events. These captions supervise the natural-language video-analysis output (*i.e.*, `Video Description` in Equation 2) and do not affect the temporal targets.

Instruction validation. We first retain only instructions that specify a well-formed and actionable robot goal. Source-specific usable-instruction sets are used when canonical task vocabularies are available, while malformed strings, placeholders, data-quality metadata, and ambiguous noun-only labels are removed. We also discard CJK annotations from sources whose task interface is defined in English. This stage removes annotation artifacts without altering the semantics of valid manipulation tasks.

Action-relevance filtering. Some mobile-manipulation trajectories contain segments that describe only locomotion or approach behavior, without a manipulation objective. This issue is most prominent in Galaxea. We identify such segments when motion predicates such as moving, approaching, turning, or walking occur without any manipulation predicate, and remove them from temporal-distance training. This ensures that a retained segment describes observable progress toward an interaction goal rather than incidental platform motion.

Table 5 gives representative examples of the annotation issues addressed by the pipeline, spanning several qualitatively different failure modes: malformed language, dataset metadata, task-irrelevant motion.

Table 5: **Representative annotation issues addressed by data curation.** Non-English examples are described rather than reproduced verbatim to avoid introducing non-Roman fonts into the manuscript.

Issue	Representative annotation	Source
Non-English task annotation	Chinese-language manipulation instruction	Galaxea
Placeholder or truncated label	`P`, `shirts`, `undefined`	InternA1
Data-quality metadata	`no robot motion`, `skip frame`	OXE
Pure-motion instruction	`move to the table`, `approach the cabinet`	Galaxea

Table 6 summarizes the quantitative effect of the complete pipeline. Overall, curation retains 1,436,150 of 1,722,966 trajectory units (83.35%) while preserving 192,989 of 194,967 unique instructions (98.99%). This gap between unit retention and instruction retention indicates that the removed volume is concentrated in invalid annotations rather than in rare valid tasks.

Table 6: **Summary of source-specific data curation.** OXE and InternA1 are counted in episodes, while Galaxea and RoboCOIN are counted in annotated subtask segments. The total therefore denotes an aggregate number of trajectory units rather than a homogeneous episode count.

Source	Raw units	Curated units	Unit retention	Raw instructions	Curated instructions	Instruction retention
OXE (episode)	961,253	693,037	72.10%	180,290	180,090	99.89%
InternA1 (episode)	320,910	320,905	99.99%	350	348	99.43%
Galaxea (segment)	97,287	78,692	80.89%	12,685	10,909	86.00%
RoboCOIN (segment)	343,516	343,516	100.00%	1,642	1,642	100.00%
Total	**1,722,966**	**1,436,150**	**83.35%**	**194,967**	**192,989**	**98.99%**

The resulting mixture retains broad task coverage while reducing several sources of supervision bias. Under a diagnostic first-verb grouping, the miscellaneous *other* category decreases from 4.54% to 1.37%, reflecting the removal of malformed and non-actionable annotations. The dominant *pick-place* category decreases from 43.38% to 33.61% after filtering invalid OXE annotations. The three largest skill groups still account for 68.21% of the curated data, showing that the pipeline does not impose an artificial uniform distribution; instead, it improves task relevance and limits template domination while preserving the natural structure of the robot data.

As an additional post-hoc diagnostic, we normalize the first verb of each instruction into an atomic skill, merging morphological variants such as *pickup*, *picking*, and *picked*. The number of atomic verb categories decreases from 640 in the raw annotations to 549 after curation. The removed categories correspond mainly to empty tokens, termination markers, malformed strings, and other

annotation artifacts. This atomic-skill analysis is used only to diagnose the effect of data curation; it does not define the temporal-distance targets or alter the training sampler.

B Model Details

This section complements Section 3.2 with the remaining details needed to instantiate TimeValue and use it as a reward model: the training configuration and optimizer, construction of the frozen temporal-distance quantile supports, chronological inference with mismatch treated as an out-of-distribution failure, and the conversion of temporal-distance estimates into a sign-reversed observation potential.

B.1 Training Hyperparameters and Optimization

Model configuration. In Table 1 we additionally report a TimeValue-4B variant to probe how performance scales with backbone size; all other experiments use the 8B model. Each training sample contains $K = 8$ observations, with eight repeated query tokens for each absolute or relative prediction slot. Both temporal heads are implemented as BroNet [41] residual MLPs with hidden width 4096, depth 8, and ReLU activations. The absolute and relative temporal targets are represented using 256-bin quantile supports over $[0, 636.3]$ and $[-636.3, 636.3]$ seconds, respectively; their construction and empirical statistics are detailed in Section B.2.

Optimization. We jointly optimize the absolute temporal-distance, relative temporal-distance, and natural-language objectives as shown in Equation 9. TimeValue is trained with AdamW using a learning rate of 1×10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, and $\epsilon = 10^{-8}$. We use a constant learning-rate schedule without warm-up and clip the gradient norm at 100. Training is conducted in bfloat16 with FSDP hybrid sharding and a per-device batch size of 2.

During training, temporal-order shuffling is applied with probability 0.5, and the forward-biased sampling process uses a rewind probability of 0.3. Instruction-mismatch augmentation is applied to 10% of the samples. For mismatched samples, the absolute temporal-distance loss is masked, while the relative and natural-language objectives are retained.

B.2 Temporal-Distance Quantile Support Construction

We construct the temporal heads' quantile supports once from training-data metadata and keep them fixed throughout training and inference. For frame index t in a trajectory with relabeled completion cutoff c and frame rate f, we include the following non-negative label in the support-fitting pool:

$$y_t = \max\left(\frac{c - t}{f}, 0\right), \tag{10}$$

This label measures the remaining time to the relabeled completion cutoff in seconds. We compute it for every training frame, yielding a frame-weighted empirical distribution in which longer trajectories contribute proportionally more samples. We use all available training trajectories without imposing a maximum-duration threshold.

Let $\mathcal{Y}$ be the resulting multiset of $N = 2{,}167{,}034{,}629$ frame-level labels. For $B = 256$ support centers, we define the i-th non-negative knot as

$$b_i = \widehat{Q}_{\mathcal{Y}}\left(\frac{i}{B-1}\right), \qquad i \in \{0, \ldots, B-1\}, \tag{11}$$

where $\widehat{Q}_{\mathcal{Y}}$ is the empirical quantile function. We store the knots in float32. If consecutive quantile levels yield the same float32 value, we advance the later knot to the next representable value to maintain a strictly increasing support. We do not clip the upper tail, so the final knot equals the maximum observed remaining time of 636.33 seconds. The signed relative support uses the same non-negative temporal scale and the symmetric construction described in Section 3.2.

Table 7: **Temporal-distance support statistics.** The distribution statistics are computed over all frame-level labels used to fit the frozen non-negative support. Spacing statistics are computed between consecutive quantile knots.

Statistic	Value
Number of frame-level labels	2,167,034,629
Number of quantile knots	256
Minimum / maximum label	0.00 / 636.33 s
Mean label	25.82 s
50th / 90th / 99th percentile	11.57 / 68.73 / 181.07 s
Minimum adjacent-knot spacing	0.0333 s
Median adjacent-knot spacing	0.1667 s
Maximum adjacent-knot spacing	419.30 s

The empirical quantile transform concentrates support centers at short horizons, where frame-level labels are densest. As quantified in Table 8, nearly half of the centers lie within the first 10 seconds, and 84.8% lie within 50 seconds. The sparse upper tail has correspondingly coarser resolution: the final adjacent-knot interval spans 217.03 to 636.33 seconds and is the widest interval in the support.

Table 8: **Allocation of quantile knots over temporal horizons.** Counts are cumulative and include all support centers no greater than the stated horizon.

Horizon	Knot count	Support share
≤ 1 s	19	7.4%
≤ 5 s	75	29.3%
≤ 10 s	119	46.5%
≤ 20 s	165	64.5%
≤ 50 s	217	84.8%
≤ 100 s	241	94.1%
≤ 200 s	254	99.2%

Each knot index defines a fixed output-bin identity. We use the same frozen centers to construct two-hot training targets and decode predictions as expected bin centers. Changing the knot locations would therefore alter the meaning of the output dimensions even if the bin count stayed the same. Accordingly, the support remains fixed across training stages, fine-tuning runs, and checkpoint evaluation.

B.3 Inference and Reward Interface

Chronological inference. At inference time, the training-time sampling augmentations are disabled and observations are arranged in chronological order. We specify the attention mask explicitly: evaluations intended to match training use value-isolation attention, while Section C.2 separately audits inference with a causal mask. Different attention masks can produce different predictions and ranking scores. TimeValue decodes its distributional outputs into continuous temporal-distance estimates, while the language branch produces the video description, instruction-video matching judgment, and success judgment. In trajectory-ranking evaluation, the matching judgment is used only to gate instruction-video mismatches; the remaining language outputs do not enter the ranking score.

Temporal-distance potential. TimeValue predicts temporal distance rather than reward directly. Let v_t denote the final temporal-distance estimate for observation I_t, after distributional decoding. The original prediction remains non-negative: a large v_t indicates that more time is required to complete the task, whereas $v_t = 0$ corresponds to the predicted completion boundary. To match the conventional value semantics that *larger is better*, we convert temporal distance into an observation potential by reversing its sign:

$$\Phi_t = \Phi_\theta(I_t, \ell, m) = -v_t, \tag{12}$$

where ℓ and m denote the task instruction and metadata, respectively. Consequently, states before task completion have negative potential, while the potential approaches zero as the agent reaches the goal. This sign reversal preserves the temporal scale rather than normalizing predictions to a task-specific $[0, 1]$ interval.

C Reward Evaluation Details

This section complements Section 4 with the RBM-EVAL-OOD inference protocol and further experiments on attention consistency, fixed-interval binning, language grounding, temporal sensitivity, and scaling behavior. A paired comparison of attention implementations quantifies the effect of matching the inference mask to training. Instruction-trajectory alignment tests whether the signal is grounded in the specified language goal rather than generic visual progress; a temporal-value curve case study examines whether the potential responds to regression, recovery, and near-completion disturbances; and a scaling comparison of episode volume versus task diversity investigates which factor drives generalization to unseen tasks. Together, these experiments show that TimeValue provides a language-conditioned, temporally sensitive value that transfers through task diversity rather than repeated observation of the same tasks.

C.1 RBM-EVAL-OOD Evaluation Protocol

The methods in Table 1 cover prompted value estimation, progress prediction, and trajectory-level reward learning. Their reported ranking scores are taken from Robometer [31]; the instruction-trajectory analysis in Figure 4 re-evaluates the subset shown there from released weights.

GVL. Generative Value Learning (GVL) [36] prompts a pretrained vision-language model to estimate task progress from video frames without reward-model fine-tuning. The benchmark implementation shuffles the sampled frames before prompting and uses GPT-5-mini to produce progress estimates.

VLAC. VLAC [61] jointly learns action generation and a language-conditioned progress critic from human and robot trajectories. Its critic compares two observations and predicts their relative progress. We report both released model sizes, VLAC-2B and VLAC-8B.

Robo-Dopamine. Robo-Dopamine [51] learns a process reward from fine-grained progress changes between observations, including forward and rewound frame pairs. The table includes the original RoboDopamine checkpoint and the later Dopamine-GRM-2.0-8B-Preview checkpoint; these are distinct model versions.

RoboReward. RoboReward [30] predicts a discrete task-execution score from a trajectory and instruction. Its supervision is at the trajectory level rather than a dense within-trajectory progress target. We report its 4B and 8B checkpoints separately.

ReWiND. ReWiND [60] learns language-conditioned progress from visual and text features and uses rewound video segments to expose task regression. The benchmark trains this baseline on RBM-1M for the reported comparison.

Robometer. Robometer [31] predicts frame-level progress and success while learning to compare which of two trajectories better satisfies an instruction. The "RoboReward data" and "RBM-1M" rows distinguish its training data; the "Progress only" row removes trajectory-comparison supervision and isolates progress-based training. These variants should not be conflated with the separate RoboReward model above.

We use Robometer's processed trajectory inputs and trajectory-ranking evaluator for RBM-EVAL-OOD [31].

Metric and scoring. We use the absolute temporal-distance head to obtain the ranking signal. Let $v_{\text{end}}(\tau, \ell)$ be the predicted remaining time at the final queried observation of trajectory τ under instruction ℓ. Following the potential definition in Equation 12, its ungated score is $-v_{\text{end}}$, so larger scores indicate that the trajectory ends closer to task completion. We enable instruction-mismatch gating: when the language branch predicts `Match: No`, the evaluator replaces the trajectory score

with -100. The resulting score is

$$s(\tau, \ell) = \begin{cases} -100, & \text{if the model predicts } \texttt{Match: No}, \\ -v_{\text{end}}(\tau, \ell), & \text{otherwise.} \end{cases} \tag{13}$$

Relative temporal predictions are not fused into this score. Neither the generated video description nor the success judgment contributes an additional scoring term; the matching judgment is used only for the mismatch gate. Within each task, we compute Kendall's τ_a between predicted scores and the annotated quality ordering, average over tasks within each suite, and report the unweighted mean over the six suites. We additionally report τ_{rw}, the same ranking statistic obtained with rewind-augmented scoring.

C.2 Ablation Studies

Table 9 compares value tokenizers and inference-time attention on RBM-EVAL-OOD. All models were trained with value-isolation attention; paired rows change only the inference mask on the same checkpoint and 567 trajectories. This differs from the 976-trajectory benchmark in Table 1 and the training-time ablation in Table 2.

Table 9: **Ablation of value tokenizers and attention implementations.** All models were trained with value-isolation attention; the Attention column specifies the inference-time mask. We apply the causal mask with Hugging Face Transformers' built-in scaled dot-product attention (SDPA) backend (`attn_implementation="sdpa"`). Paired attention runs use the same checkpoint and the same 567 processed trajectories. Value tokenizer is the support used for the temporal-distance head. Fixed-bin rows replace the quantile-knot support with fixed-interval bins. Average is the unweighted mean over suites. Bold values indicate the best result in each suite, including ties.

Design Components			Kendall's τ_a						
Parameters	Value Tokenizer	Attention	USC Franka	USC Koch	USC Trossen	USC xArm	MIT Franka	UTD SO101	Average
8B	quantile	value-isolation	0.667	0.437	0.972	0.556	0.442	0.800	0.646
4B	quantile	value-isolation	0.583	0.485	0.931	0.500	0.450	0.733	0.614
8B	normalized	value-isolation	0.583	0.416	0.778	0.500	0.493	0.733	0.584
8B	normalized	causal mask	0.583	0.635	0.806	0.639	0.475	0.867	0.667
8B	fixed bins	causal mask	0.667	0.544	**1.000**	0.500	0.503	0.833	0.675
4B	fixed bins	causal mask	0.542	0.488	0.917	0.667	0.473	**0.933**	0.670
8B	quantile	causal mask	**0.750**	0.504	**1.000**	**0.722**	0.450	0.800	**0.704**
4B	quantile	causal mask	0.542	**0.645**	**1.000**	0.500	**0.518**	0.867	0.679

Attention. Isolation lets value-query tokens interact within a prediction group but blocks their representations from later observation and value groups. The causal mask opens that path at inference, allowing later queries to use earlier value-query states, though not their decoded scalar predictions. On paired trajectories, the causal mask raises mean Kendall's τ_a from 0.646 to 0.704 for 8B quantile, 0.614 to 0.679 for 4B quantile, and 0.584 to 0.667 for 8B normalized. The effects vary by suite: the 8B quantile gain is 0.166 on USC xArm but only 0.008 on MIT Franka; 4B quantile gains 0.160 on USC Koch yet loses 0.041 on USC Franka. Earlier value states may disambiguate the final queried score by carrying trajectory context. Terminal-score ranking cannot, however, establish better frame-level distance estimates or sensitivity to local regression. The "w/o Isolation" model in Table 2 instead removes the mask during training and lowers the reported mean from 0.704 to 0.611: unlike the inference switch here, that intervention permits the cross-group shortcut throughout learning.

Value tokenizer. The strongly skewed training labels (median 11.57 s, 90th percentile 68.73 s, maximum 636.33 s; Table 7) motivate non-uniform temporal resolution. Of 256 quantile centers, 119 fall within 10 s and 217 within 50 s, versus about 20 of the same 256 fixed-interval centers within 50 s when spread to the same maximum (Table 8). Two-hot targets thus resolve short horizons finely but coarsen the sparse tail. Normalized progress changes the target scale: the same episode fraction can represent different remaining durations, weakening a shared cost-to-go interpretation.

Consistent with these mechanisms, 8B quantile exceeds normalized under isolation (0.646 versus 0.584) and both normalized and fixed bins with a causal mask (0.704 versus 0.667 and 0.675); at 4B, its mean with a causal mask is 0.679 versus 0.670 for fixed bins. The suite-level reversals qualify this mean advantage: normalized 8B leads on USC Koch; fixed-bin 4B leads on USC xArm and UTD SO101; quantile 4B leads on USC Koch and USC Trossen. Fixed bins may help at long horizons; normalized progress may help when fraction completed matters. The table lacks pairwise horizons and errors needed to test either explanation, so label density remains a plausible account of the mean rather than a demonstrated cause of each suite-level difference.

C.3 Instruction-Trajectory Alignment

To assess whether the predicted reward tracks the specified language goal rather than generic visual progress, we perform an instruction-trajectory alignment analysis: we score every instruction against every trajectory and inspect the resulting matrix, following the evaluation protocol of Robometer [31]. A well-grounded model should assign higher rewards to matched instruction-trajectory pairs along the diagonal and lower rewards to mismatched pairs. As shown in Figure 4, to ensure that all confusion matrices are produced under an identical protocol, we re-evaluate every baseline with its publicly released model weights, and report the resulting diagonal-margin scores. TimeValue produces the clearest diagonal structure and achieves the highest normalized diagonal margin of **0.79**, outperforming the strongest baseline at 0.67. In contrast, several baselines exhibit more diffuse off-diagonal responses. These results show that TimeValue more effectively distinguishes matched from mismatched instruction-trajectory pairs, grounding its temporal-distance estimates in the language-specified goal rather than generic visual progress alone.

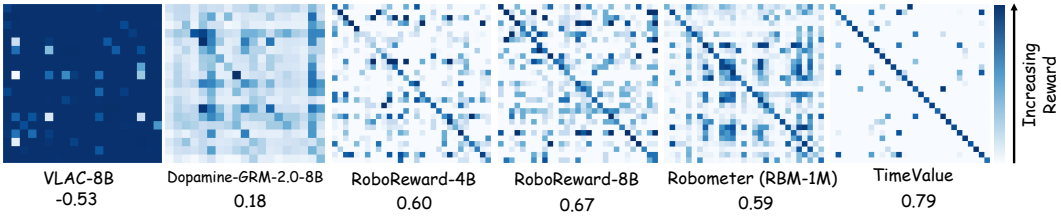

Figure 4: Instruction-trajectory confusion matrices. Each cell shows the predicted reward when an instruction (rows) is paired with a trajectory (columns); a well-grounded model concentrates mass on the diagonal. Values below each matrix report the normalized diagonal margin. All models are re-evaluated under a unified protocol from their publicly released weights.

C.4 Temporal-Value Curve Case Study

Figure 5 compares the value curves predicted by TimeValue and Robometer along the same real-world trajectory. Because Robometer outputs normalized progress while TimeValue outputs temporal-distance potential ($\Phi_t = -v_t$), the two scales are not directly comparable; we therefore orient both so that higher values indicate closer proximity to task completion. The visualization reveals three notable differences.

First, TimeValue responds more strongly to trial-and-error behavior and task regression. During the highlighted interval, its potential decreases substantially as the robot moves away from a productive state, whereas Robometer shows a weaker response. This sensitivity is consistent with temporal-order shuffling and value-isolation attention, which encourage predictions to rely on visual evidence rather than sequence position or previously exposed values.

Second, after the robot recovers, TimeValue exhibits a steadier increase toward completion, while Robometer contains longer plateaus and abrupt jumps. This behavior benefits from jointly learning absolute and relative temporal values, which capture global remaining cost and local temporal changes, respectively.

Third, TimeValue remains sensitive near task completion: late disturbances cause an immediate potential decrease followed by recovery. This avoids premature reward saturation and better distinguishes intermediate progress from actual completion.

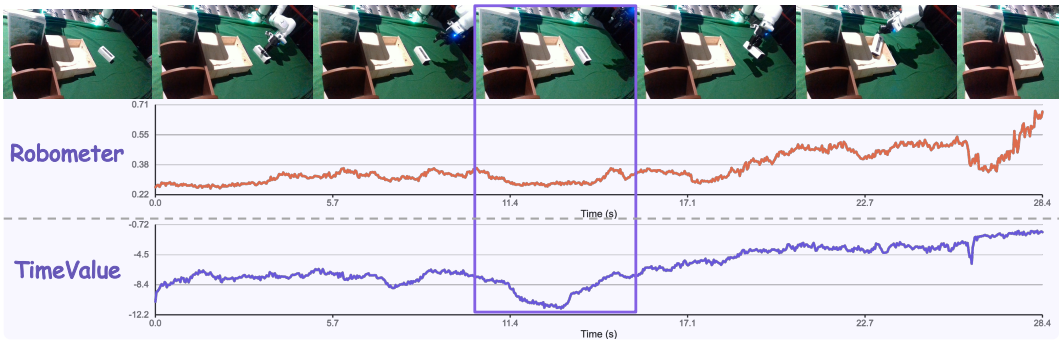

Figure 5: Temporal-value curve comparison on a real-world trajectory. Higher values indicate closer proximity to task completion. The highlighted interval marks a period of task regression where the robot moves away from a productive state; TimeValue responds with a sharp potential drop, whereas Robometer remains relatively flat.

C.5 Scaling Analysis

Having isolated the contribution of each model component, we next ask whether the heterogeneous data recipe itself drives generalization. To disentangle *quantity* from *diversity*, we construct two families of training subsets from the full mixture. In the **episode-volume** family (orange), we fix the full set of training tasks and randomly subsample $\{1\%, 10\%, 25\%, 50\%, 75\%\}$ of episodes within each task. In the **task-diversity** family (blue), we fix per-task episode counts at the full-scale level and randomly subsample the same fractions of tasks. At each fraction, both protocols train on a comparable total number of episodes, ensuring that performance differences reflect diversity rather than data volume. Both families converge to the same full training set at 100%. We train each variant from scratch under identical optimization settings and report the mean absolute temporal-distance error on a held-out validation set whose tasks do not overlap with any training subset.

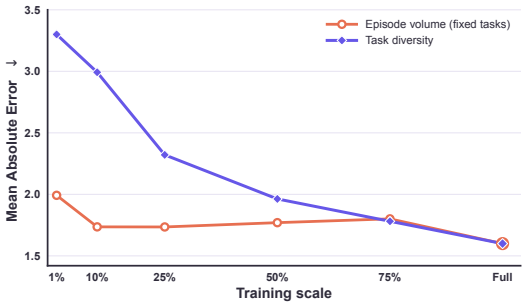

Figure 6: Scaling episode volume vs. task diversity. Mean absolute temporal-distance error on a held-out validation set of unseen tasks. We independently scale two aspects of the training set: (orange) episode count with the full task set fixed, and (blue) task count with per-task episode counts fixed. Both curves converge to the same full-scale training set at 100%. Task diversity yields a consistently steeper error reduction than episode volume, which saturates early.

As shown in Figure 6, episode-volume scaling saturates almost immediately: beyond a small fraction of episodes the error plateaus and does not improve further as more within-task data are added. Task-diversity scaling behaves qualitatively differently: increasing the number of training tasks reduces error monotonically across the entire range, with substantial gains still observable well past the midpoint. This contrast demonstrates that the heterogeneous data recipe contributes not merely additional samples, but the diversity required for general-purpose temporal-value learning: broader task coverage introduces diverse goal structures, visual configurations, and execution patterns that transfer to unseen tasks, whereas repeated observation of the same tasks quickly stops adding signal.

D Real-World Experimental Settings

This section complements Section 5 with the real-world protocol on a dual-arm Franka platform. We specify four manipulation tasks, mixed-expertise datasets, the camera and control stack, and an evaluation that reports success rate and execution length under fixed offline resets and randomized online initializations. Offline IQL and online DSRL use the language-conditioned value for potential-based reward shaping, while multi-task data filtering and inference-time value guidance use it to select training data and action proposals, respectively. Together, these settings test whether



Figure 7: **Representative demonstrations for real-world evaluation.** Each row shows a sequence of observations from one manipulation task. These tasks cover object grasping, spatial manipulation, and articulated-object interaction.

temporal-distance estimates transfer to unseen tasks, objects, and scenes without target-domain fine-tuning.

D.1 Experimental Settings

Robot platform. Our experiments are conducted on a dual-arm Franka robot system with wrist-mounted cameras on both arms and two additional third-person cameras capturing the workspace from the left and right sides.

Tasks and datasets. We evaluate on the following four real-world robotic manipulation tasks as shown in Figure 7:

- **Bread Basket Placement**: The robot places two pieces of bread into a basket.
- **Steak Serving with a Spatula**: The robot uses a spatula to transfer a steak from a pan to a plate.
- **Box-in-Drawer Placement**: The robot places a box into a drawer and then closes the drawer.
- **Bimanual Box Transfer**: Two robot arms collaboratively move a box to a target location.

For each task, we collect approximately 100 successful trajectories and retain every unsuccessful attempt encountered during the same collection process, yielding a mixed-expertise dataset. Table 10 reports the task instructions and dataset statistics.

Table 10: **Real-world task instructions and dataset statistics.** Success rate is computed over all collected trajectories for each task.

Task	Language instruction	Trajectories			Success rate
		Success	Failure	Total	
Bread Basket Placement	"Put the two pieces of bread in the basket."	99	4	103	96.1%
Steak Serving with a Spatula	"Move the steak from the pan to the plate."	98	4	102	96.1%
Box-in-Drawer Placement	"Put the box in the drawer and close it."	101	3	104	97.1%
Bimanual Box Transfer	"Move the box from the right side to the left side."	100	1	101	99.0%
Total	–	**398**	**12**	**410**	**97.1%**

Baselines and reward shaping. We compare TimeValue against Robometer, the strongest baseline in our benchmark evaluations, and Sparse Reward, which assigns -1 before task completion and

0 upon successful completion. We additionally include two scalable post-training baselines, multi-task SFT and Filter BC, detailed in Section 5. Plain SFT serves as the per-task imitation-learning reference. To ensure a fair comparison, all reward models are used through the same potential-based reward shaping interface:

$$r'_t = \kappa\big(\gamma\Phi_{t+1} - \Phi_t\big) + \begin{cases} 0, & \text{if } t = T \text{ and the trajectory is successful,} \\ -1, & \text{otherwise.} \end{cases} \tag{14}$$

where Φ_t denotes the potential predicted by the reward model at step t, γ is the discount factor, and κ controls the strength of the shaping term. We use $\kappa = 0.1$ for TimeValue and $\kappa = 1.0$ for Robometer in both the offline and online experiments. We retain the sparse completion term because reward-model predictions may be noisy, whereas the success label provides a clean and reliable signal for the final task objective. These success labels are manually annotated by human operators during both data collection and online policy training. For Sparse Reward, we set $\Phi_t = 0$ for all t, so the reward is -1 before completion and 0 on the transition that successfully completes the task.

D.2 Evaluation Protocols

We evaluate each policy over 20 trials per tested task and report both the success rate and the average number of action chunks over successful episodes. All offline evaluations use fixed initial configurations. During online evaluation, we randomize the initial object configurations for *Bread Basket Placement*, *Steak Serving with a Spatula*, and *Bimanual Box Transfer* to assess robustness under varying initial conditions, while *Box-in-Drawer Placement* uses a fixed reset because the task requires precise gripper control and box-drawer alignment.

Reset, success, and termination. Environment resets are performed by an operator, who repositions the robot and objects and confirms that the next episode may begin. All offline-policy evaluations use fixed initial configurations. During online evaluation, Bread Basket Placement, Steak Serving with a Spatula, and Bimanual Box Transfer use randomized initial-object configurations, whereas Box-in-Drawer Placement uses a fixed reset because the task requires precise gripper control and box–drawer alignment. The operator records a binary task-success label. An episode terminates when success or failure is confirmed or when it reaches the maximum horizon of 600 low-level control steps. Reward models provide only the shaping potential; they do not determine success or termination.

Offline RL with mixed-expertise datasets. First, we evaluate whether TimeValue can effectively recover high-performing policies from mixed-expertise datasets. Specifically, using $\pi_{0.5}$ as the base policy, we apply Implicit Q-Learning (IQL) [29] to train policies on these datasets with reward signals provided by either TimeValue or the baselines.

Online RL with diffusion steering. Next, we investigate whether TimeValue can accelerate online policy learning by providing real-time reward specification. We adopt Diffusion Steering via Reinforcement Learning (DSRL) [53] with task-specific policy initialization. Bread Basket Placement and Steak Serving with a Spatula start from their SFT checkpoints, whereas Box-in-Drawer Placement and Bimanual Box Transfer start from the corresponding Robometer-based offline-RL checkpoints. This initialization is held fixed across the online reward variants, whose collected trajectories are specified by either TimeValue or the baselines.

Scalable post-training via data filtering. Scalable post-training should improve a generalist policy across *all* tasks at once, rather than requiring a separate adaptation run for each task. We therefore test whether TimeValue can curate multi-task behavior-cloning (BC) data without target-task fine-tuning. Concretely, we pool the demonstrations from all four tasks, score every trajectory segment with TimeValue, and discard the low-quality ones before fine-tuning the pretrained policy.

Inference-time value guidance. We evaluate inference-time value guidance on Bread Basket Placement and Steak Serving with a Spatula using a frozen multi-task policy. At each decision step, Cosmos3-Edge samples four candidate action chunks together with their imagined futures, and TimeValue selects the candidate with the largest predicted progress. The sampling and scoring procedure is detailed in Section E.7.

E Policy Learning Algorithms

This section complements Section 5 with the policy backbones used on the real-world platform and the offline IQL and online DSRL procedures that consume them.

E.1 Policy Backbones

Real-world policy learning uses two flow-matching backbones of different types: a vision-language-action (VLA) model and a world-action model (WAM). Offline IQL, online DSRL, per-task SFT, and multi-task data filtering all fine-tune or decode from the VLA, $\pi_{0.5}$. Inference-time value guidance instead uses a Cosmos3-Edge WAM jointly fine-tuned on all four real-world tasks to generate future video and actions; its value-guidance evaluation is restricted to Bread Basket Placement and Steak Serving with a Spatula. Both backbones are conditioned on multi-view RGB and a language instruction; neither receives proprioception. They differ in how they consume vision and emit actions: $\pi_{0.5}$ ingests per-camera 224×224 views, predicts a 16-step chunk padded to width 32, and uses relative gripper commands at 10 Hz, whereas Cosmos3-Edge packs cameras into a single mosaic, jointly generates 16 future frames and 16 absolute-joint actions padded to width 64, and is trained at 30 FPS.

$\pi_{0.5}$ Backbone. SFT, offline IQL, and online DSRL all use the same flow-matching vision-language-action policy, $\pi_{0.5}$. Sharing this backbone isolates the effect of the learning algorithm and reward specification. The VLA consumes camera images resized to 224×224, image-validity masks, the task instruction, and the flow-matching timestep. It outputs an action chunk with horizon $H = 16$ and padded action dimension $d_a = 32$. The low-level controller executes absolute joint-position commands and relative gripper commands at 10 Hz. A single-arm action contains seven absolute joint-position values and one relative gripper value; a bimanual action contains 14 absolute joint-position values and 2 relative gripper values. These task-specific actions are padded to $d_a = 32$. Each VLA prediction contains an action chunk with horizon $H = 16$.

Cosmos3-Edge Backbone. The inference-time value-guidance experiments in Section 5 use a Cosmos3-Edge world-action model (WAM) jointly fine-tuned on all four real-world tasks, rather than the $\pi_{0.5}$ backbone of the IQL and DSRL runs. We initialize from Cosmos3-Edge-Policy-DROID, retain the Nemotron-2B reasoner and a mixture-of-transformers (MoT) generation branch, and jointly synthesize future video and action chunks by flow matching. At inference the policy is queried in world-action mode: given the current multi-view observation and the language instruction, it generates $N = 4$ candidate future videos together with the corresponding 16-step action chunks, which TimeValue then ranks. We record the interfaces and training recipe of that policy below.

Tasks and data mixture. We jointly fine-tune on the same four real-world tasks as in Section D.1, sampling them in equal proportion 1:1:1:1 so that each task occupies approximately 25% of every training batch. Failed episodes are excluded. Table 11 lists the embodiment, action dimension, and camera streams of each task. Bread Basket Placement, Steak Serving with a Spatula, and Box-in-Drawer Placement are single-arm skills; Bimanual Box Transfer is a dual-arm skill.

Table 11: **Four-task Cosmos3-Edge SFT mixture.** Tasks are sampled in equal proportion. Failed episodes are not used.

Task	Embodiment	Action dim.	Visual input
Bread Basket Placement	single-arm	8	`left_side`, `left_wrist`
Steak Serving with a Spatula	single-arm	8	`left_side`, `left_wrist`
Box-in-Drawer Placement	single-arm	8	`left_side`, `left_wrist`
Bimanual Box Transfer	dual-arm	16	`left_side`, `left_wrist`, `right_wrist`

Language, vision, and temporal window. The policy is vision-language only: proprioception is not provided. The language input is the natural-language instruction of the current task, for example "pick up the bread" or "close the drawer". When a task has several phrasings, they are stored as a

Table 12: **Cosmos3-Edge four-task SFT hyperparameters.** The recipe fine-tunes a DROID-initialized Cosmos3-Edge world-action model jointly on the four real-world tasks.

Hyperparameter	Value
Initialization	Cosmos3-Edge-Policy-DROID
Reasoner	Nemotron-2B
Training algorithm	SFT with flow matching
Action representation	absolute joint position
Maximum iterations	40,000
GPUs / per-GPU batch / accumulation	8 / 32 / 1 (global batch 256)
Trainer / episode-shuffle seed	42 / 42
Precision	BF16 activations; FP32 FSDP master weights
Gradient clipping	1.0
Checkpoint interval	1,000 iterations
Optimizer	FusedAdam with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$, weight decay 0.05
Base learning rate	5×10^{-5}
Action-adapter learning rate	2.5×10^{-4} ($5\times$ the base rate) on `action2llm`, `llm2action`, `action_modality_embed`
Scheduler	LambdaLinear; 500-step warm-up; 40,000-step cycle; final learning-rate multiplier 0
Independent action schedule	yes; logit-normal action time; action-time shift 1.0; uniform train-time weighting
Dynamic shift / discrete rectified flow	off / off
Action / vision flow-matching loss weights	10.0 / 10.0
Action σ	one scalar shared across the 16-step action sequence
EMA	Power EMA with rate 0.1
Text-conditioning dropout	0.1
Proprioceptive state	not used
Action horizon / padded width	$H = 16$ / 64
Data frame rate	30 FPS

|-delimited list and one phrasing is sampled per training example. A text-conditioning dropout of 0.1 is applied for classifier-free guidance.

Single-arm observations concatenate `left_side` and `left_wrist` horizontally. The raw mosaic is 224×448 and is resized by the model image pipeline to approximately 192×320. Bimanual observations are packed into a 2×2 mosaic of resolution 256×256, with `left_side` and `right_wrist` on the top row, `left_wrist` on the bottom left, and a black placeholder on the bottom right.

Demonstrations are recorded at 30 FPS. Each training example contains 17 video frames and the corresponding 16-step future action chunk. At inference, frame 0 is the visual condition and the model predicts the subsequent 16 video frames together with 16 actions, for a horizon of $16/30 \approx 0.53$ s.

Action representation. Actions are absolute joint positions rather than joint deltas. Internally the model uses a padded action tensor of shape $[B, 16, 64]$; only the first d_a dimensions are valid, and the remaining coordinates are zero-filled and masked in the loss, the injected noise, and the predicted-velocity computation. For single-arm tasks, $d_a = 8$ stores seven absolute arm joints (radians) followed by an absolute gripper command. For the bimanual task, $d_a = 16$ stores fourteen absolute arm joints followed by the left and right absolute gripper commands. Each dimension is linearly normalized from its 1st/99th-percentile statistics onto $[-1, 1]$: the three single-arm tasks share one 8-dimensional absolute-joint statistic, while the bimanual task uses a separate 16-dimensional statistic. Predicted actions are denormalized with the corresponding statistic before execution.

Optimization. Table 12 reports the supervised fine-tuning configuration. We train for 40,000 iterations in BF16 (FSDP master weights in FP32) on 8 GPUs with a per-GPU batch size of 32 and no gradient accumulation, giving a global batch size of 256. The optimizer is FusedAdam with a base learning rate of 5×10^{-5}; the action adapters `action2llm`, `llm2action`, and `action_modality_embed` use a $5\times$ learning rate of 2.5×10^{-4}. A LambdaLinear schedule warms up for 500 steps over a 40,000-step cycle and decays to a final learning-rate multiplier of 0. Flow matching uses an independent action-time schedule with a logit-normal action-time distribution, so that the action and video branches do not share a flow-matching timestep; a single scalar action σ is shared across the 16-step action sequence of each example. Training uses Power EMA with rate 0.1.

Trainable parameters. The four-task SFT stage is a selected-module fine-tuning procedure rather than a full-parameter update: only the generation tower and the action pathway are trained, while the vision-language understanding tower, the VAE, and all remaining parameters stay frozen. All parameters of the selected modules are updated densely. Table 13 lists the selected modules.

Table 13: **Selected modules.** The recipe uses selected-module fine-tuning: only the generation tower and the action pathway are optimized.

Status	Modules
Updated	generation tower `moe_gen`; time embedder `time_embedder`; VAE–LLM adapters `vae2llm`, `llm2vae`; action pathway `action2llm`, `llm2action`, `action_modality_embed`
Frozen	vision-language understanding tower; VAE; all parameters outside the selected keys

E.2 Implementation Inputs and Outputs

The $\pi_{0.5}$ VLA interface is specified in Section E.1. The remaining policy-learning interfaces are the IQL critic and the DSRL actor–critic. Throughout this section, o_t denotes the multi-camera visual observation at policy-decision step t, and a_t denotes the corresponding executable action chunk. No component receives a proprioceptive or robot-state vector.

Control and action representation. Low-level control runs at 10 Hz. During online DSRL, the latent action has shape $H \times d_z$ with $d_z = 32$, and the frozen $\pi_{0.5}$ decoder is queried after every $q = 10$ executed low-level control steps. The $\pi_{0.5}$ action layout is given in Section E.1.

Camera configuration. The dual-arm Franka platform is instrumented with 4 Intel RealSense cameras: 2 D435 third-person cameras, denoted `left_side` and `right_side`, and 2 D405 wrist-mounted cameras, denoted `left_wrist` and `right_wrist`. Images are resized with aspect-preserving padding to 224×224 for $\pi_{0.5}$ and the IQL critic; the DSRL visual encoder receives inputs downsampled to 64×64. Table 14 lists the exact streams used by each pathway.

Table 14: **Camera streams used by the policy and RL encoders.** Entries list the exact streams provided to each pathway.

Pathway	Single-arm tasks	Bimanual Box Transfer
$\pi_{0.5}$	`left_side`, `left_wrist`	`left_side`, `left_wrist`, `right_wrist`
IQL critic	`left_side`, `left_wrist`	`left_side`, `left_wrist`, `right_wrist`
DSRL actor–critic	`left_side`, `left_wrist`	`left_side`, `right_side`, `left_wrist`, `right_wrist`

IQL critic and value. The offline RL networks consume the current and next multi-camera observations together with the demonstrated action chunk. They estimate $Q(o, a)$ and $V(o)$, from which we compute the advantage weight used to refine $\pi_{0.5}$.

DSRL actor–critic. The online RL networks consume only multi-camera observations at 64×64. The actor predicts a latent-noise chunk $z \in [-1, 1]^{16\times 32}$, the critics estimate its Q-values, and automatic entropy tuning optimizes the temperature α.

Reward-model inference. TimeValue and Robometer are deployed through the same scoring interface. For each collected trajectory, the reward service receives the task instruction and the RGB sequence from the `right_side` third-person camera. Both models operate in a causal, history-conditioned scoring mode. Given a trajectory containing T RGB observations, for each step $t = 1, \ldots, T$, the service scores the observation history $o_{1:t}$, uniformly subsampled to 4 frames (rather than the 8 used during training, to match Robometer's 4-frame inference protocol) before inference. Robometer decodes its native normalized progress $p_t \in [0, 1]$ as the expectation over ten progress bins and uses $\Phi_t^{\mathrm{Robo}} = p_t$. During reward relabeling and online RL, TimeValue uses only its absolute temporal-distance head and does not invoke the language-generation branch. Following

the notation in the main text, we convert its predicted temporal distance v_t into a higher-is-better potential using $\Phi_t = -v_t$. Each complete trajectory is scored once after collection, and the resulting potential sequence is sampled at policy-decision boundaries to construct the shaping rewards. Reward relabeling therefore operates at the action-chunk granularity used by the RL algorithm, rather than at every low-level control step.

E.3 Policy and RL Initialization

RL networks. The auxiliary RL networks are small relative to the VLA. Offline IQL uses a ResNet-18 visual encoder with GroupNorm, spatial softmax, and a 50-dimensional bottleneck, followed by MLP heads with hidden dimensions $(256, 256)$. It maintains 2 Q-functions and a separate value function. Online DSRL uses a 4-layer convolutional visual encoder with 32 channels per layer, GroupNorm, spatial softmax, and a 50-dimensional bottleneck. Its actor and critic MLPs have hidden dimensions $(128, 128, 128)$; the critic ensemble contains ten Q-functions, and the actor is a $\tanh$-squashed Gaussian distribution over latent noise.

Initialization and training stages. SFT and IQL are independent, single-stage fine-tuning procedures initialized from the same pretrained $\pi_{0.5}$ checkpoint; IQL is not initialized from SFT. SFT optimizes the standard flow-matching behavior-cloning objective, whereas IQL uses advantage-weighted flow matching. Online DSRL uses task-specific initialization: Bread Basket Placement and Steak Serving with a Spatula start from their SFT checkpoints, while Box-in-Drawer Placement and Bimanual Box Transfer start from the corresponding Robometer-based offline-RL checkpoints. The initial checkpoint for each task is held fixed across all online reward variants. During DSRL, the initialized VLA remains frozen and serves only as a noise-conditioned action decoder; the latent policy, critic ensemble, and entropy temperature are optimized.

E.4 Offline RL with IQL

Mixed-expertise dataset construction. For each task j, we construct a mixed-expertise offline dataset by combining all successful and unsuccessful trajectories:

$$\mathcal{D}_j = \mathcal{D}_{j,\text{succ}} \cup \mathcal{D}_{j,\text{fail}}. \tag{15}$$

Each trajectory is segmented at policy-decision boundaries into transitions (o_h, a_h, o_{h+1}, m_h), where a_h is an action chunk and $m_h \in \{0, 1\}$ is the bootstrap mask. We set $m_h = 0$ for terminal transitions and $m_h = 1$ otherwise.

All reward variants share the same transitions, task-outcome labels, policy initialization, and optimization configuration. We relabel this common dataset with the potential produced by TimeValue or the corresponding baseline reward model; only the potential source and shaping coefficient differ across variants.

Offline reward construction. For a transition observation o_h, we write $\Phi_h = -v_h$ for its shaping potential, where v_h is TimeValue's decoded absolute temporal-distance prediction for o_h (Equation 12); the instruction ℓ and metadata m are omitted for brevity. The sparse task reward is

$$r_h^{\text{sparse}} = \begin{cases} 0, & \text{if the task is completed after executing } a_h, \\ -1, & \text{otherwise.} \end{cases} \tag{16}$$

Thus, every transition before completion receives -1, while the transition that completes the task receives 0. All transitions in an unsuccessful trajectory receive -1.

The offline potential-based shaping reward is

$$r_h^{\text{shape}} = \gamma_{\text{off}} \Phi_{h+1} - \Phi_h, \tag{17}$$

where γ_{off} is the offline RL discount factor. The final reward used to train IQL is

$$r_h^{\text{off}} = r_h^{\text{sparse}} + \kappa r_h^{\text{shape}}, \tag{18}$$

where $\kappa \geq 0$ is the shaping coefficient used in the main text. We set $\kappa = 0.1$ for TimeValue, $\kappa = 1.0$ for Robometer, and $\kappa = 0$ for the sparse-reward baseline. These values are fixed across tasks.

Table 15: **Offline IQL and SFT hyperparameters.** All IQL reward variants share the same mixed-expertise dataset, sparse task reward, policy initialization, and optimization configuration. SFT uses the same policy optimizer and learning-rate schedule and is also trained for $10{,}000$ steps per task.

Hyperparameter	Value
Base policy	$\pi_{0.5}$ with flow matching; action dimension 32
Action horizon H	16
Batch size	64
Policy optimizer	AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}, and gradient-norm clipping at 1.0
Learning-rate schedule	Cosine decay with 2,000 linear warm-up steps
Peak / final policy learning rate	3×10^{-5} / 3×10^{-6}
Policy EMA decay	0.99
Training steps per task	10,000
Critic / value optimizer	Adam with a learning rate of 3×10^{-4}
Offline discount γ_{off}	0.99
Target update rate ρ_{off}	0.005
Expectile parameter τ_e	0.8
Advantage temperature β	10.0
Maximum advantage weight $w_{\max}$	100
Number of Q-functions K_{IQL}	2; minimum aggregation
Critic and value encoder	ResNet-18 with GroupNorm and spatial softmax; 50-dimensional bottleneck
Critic and value hidden dimensions	$(256, 256)$
Number of critic cameras	2 for single-arm tasks and 3 for the bimanual task
Critic input resolution	224×224 using the VLA preprocessing pipeline
Policy warm-up N_{warm}	200 optimization steps with $w(o, a) = 1$
Sparse task reward	-1 before task completion and 0 upon task completion
Potential-based shaping reward	$r_h^{\text{shape}} = \gamma_{\text{off}} \Phi_{h+1} - \Phi_h$
Shaping coefficient κ	0.1 for TimeValue and 1.0 for Robometer; fixed across tasks
Sparse-reward baseline	$\kappa = 0$
Image augmentation	Random cropping applied to both current and next observations; no color jitter

IQL optimization and VLA refinement. We use the standard IQL objective [29]: the value function is trained by expectile regression, each Q-function uses a one-step temporal-difference target, and the 2 target Q-functions are aggregated by their minimum. Target Q-networks are updated by Polyak averaging. Rather than introducing a separate IQL policy, we retain the flow-matching VLA and weight its training loss by

$$w(o, a) = \min \left\{ \exp \left(\beta \left[\min_k Q_{\theta_k}(o, a) - V_\psi(o) \right] \right), w_{\max} \right\}. \tag{19}$$

For the first N_{warm} optimization steps, we set $w(o, a) = 1$; thereafter, demonstrated actions with larger estimated advantages receive greater weight. Table 15 reports the complete architecture and optimization configuration.

E.5 Online RL with DSRL

Latent policy optimization. For online RL, we use Diffusion Steering via Reinforcement Learning (DSRL) [53]. The task-specific VLA checkpoint described above is frozen and used as a noise-conditioned action decoder:

$$a_t = \pi_{0.5}(o_t, z_t). \tag{20}$$

A Soft Actor-Critic (SAC) policy $\mu_\varphi(z_t \mid o_t)$ is trained over the latent noise z_t, rather than directly over executable robot actions. The latent action space is

$$z_t \in [-1, 1]^{H \times d_z}, \tag{21}$$

where $H = 16$ is the action horizon and $d_z = 32$ is the per-step latent dimension.

SAC optimization. We use the standard entropy-regularized SAC objective in the latent action space. Critic targets use the mean of ten target Q-functions, the actor trades off the ensemble-mean Q-value against policy entropy, and the temperature is optimized automatically toward target entropy $\overline{\mathcal{H}} = -\dim(z)$. Target critics are updated by Polyak averaging. Table 16 reports the complete architecture and optimization configuration.

Online reward construction. Online DSRL uses the same sparse task-reward convention as Equation 16: each transition receives -1 until task completion, and the completing transition receives 0.

Let γ_s denote the per-environment-step shaping discount. Because each policy decision executes q low-level control steps, the corresponding chunk-level discount is

$$\gamma_c = \gamma_s^q, \tag{22}$$

and the online shaping reward is

$$r_t^{\text{shape}} = \begin{cases} \gamma_c \Phi_{t+1} - \Phi_t, & \text{if } o_{t+1} \text{ is nonterminal,} \\ -\Phi_t, & \text{if } o_{t+1} \text{ is terminal.} \end{cases} \tag{23}$$

We use the shaping coefficient κ defined in the main text, setting $\kappa = 0.1$ for TimeValue, $\kappa = 1.0$ for Robometer, and $\kappa = 0$ for the sparse-reward baseline. These values are fixed across tasks, and all variants otherwise share the same reward definition and online optimization configuration. The final reward used by SAC is

$$r_t^{\text{on}} = r_t^{\text{sparse}} + \kappa r_t^{\text{shape}}. \tag{24}$$

Table 16: **Online DSRL hyperparameters.** SAC operates in the latent space of the frozen VLA, which decodes latent variables into executable action chunks. All reward variants share the same optimization configuration and differ only in the potential source and shaping coefficient.

Hyperparameter	Value
Base policy	Frozen SFT checkpoint for Bread Basket Placement and Steak Serving with a Spatula; frozen Robometer offline-RL checkpoint for Box-in-Drawer Placement and Bimanual Box Transfer
Latent action space	$z \in [-1, 1]^{H \times d_z}$, with $H = 16$ and $d_z = 32$
RL algorithm	SAC with automatic entropy tuning and initial temperature $\alpha_0 = 1.0$
Actor optimizer	Adam with a learning rate of 1×10^{-4}
Critic optimizer	Adam with a learning rate of 3×10^{-4}
Temperature optimizer	Adam with a learning rate of 3×10^{-4}
Gradient clipping	None
Target entropy $\overline{\mathcal{H}}$	$-\dim(z)$
Online SAC discount γ_{on}	0.999
Target update rate ρ_{on}	0.005
Number of Q-functions K_{SAC}	10; mean aggregation
Actor and critic hidden dimensions	$(128, 128, 128)$
Image encoder	Four-layer CNN with 32 channels per layer, strides $(2, 1, 1, 1)$, VALID padding, GroupNorm, spatial softmax, and a 50-dimensional bottleneck
SAC input resolution	64×64
Batch size	256
Update-to-data ratio	100
Training length	6,000 training steps
Online rollout trajectories	60 per task
Replay-buffer capacity	$\max(\text{training steps}/\text{UTD}, 10^4) = 10^4$
Update frequency	After each episode
Exploration warm-up	$N_{\text{noise}} = 2$ episodes with Gaussian noise standard deviation $\sigma = 0.1$
Minimum replay size N_{start}	200 transitions
Maximum episode length	600 environment steps
Policy-decision frequency	Low-level control at 10 Hz, with one policy decision every $q = 10$ environment steps
Number of DSRL cameras	2 for single-arm tasks and 4 for the bimanual task
Sparse task reward	-1 before task completion and 0 upon task completion
Shaping coefficient κ	0.1 for TimeValue, 1.0 for Robometer, and 0 for the sparse-reward baseline; fixed across tasks
Per-step shaping discount γ_s	0.999

E.6 Scalable Post-Training with Data Filtering

This section describes the data-filtering procedure used by the Filtered BC baseline in Section 5. Filtered BC identifies and excludes low-quality action chunks based on task progress, then trains a standard behavior-cloning policy on the retained demonstration frames. Filtering is performed offline at the frame level; subsequent policy training requires neither an additional critic or value function nor policy-side conditioning.

Progress-based filtering strategy. Given a demonstration trajectory $\{(o_t, a_t)\}_{t=0}^{T-1}$, TimeValue assigns a scalar task-progress score p_t to each observation. The score reflects task completion and is expected to increase as the task progresses. To identify frames associated with forward progress, we define a windowed advantage using only the resulting score sequence. Specifically, the per-frame reward is the progress increment

$$r_t \;=\; p_{t+1} - p_t, \qquad r_{T-1} \;=\; 0, \tag{25}$$

where the final-frame reward is set to zero because no subsequent frame is available. The windowed advantage at frame t is the λ-discounted sum of these increments over a window of length W:

$$A_t \;=\; \sum_{k=0}^{W-1} \lambda^k \, r_{t+k}, \qquad \text{with } r_{t+k} \equiv 0 \text{ for } t+k \geq T. \tag{26}$$

Rewards beyond the episode boundary are set to zero, yielding a truncated return for frames near the end. Advantages are computed independently for each episode, so no window extends into another episode.

The windowed advantage therefore measures near-term progress toward task completion. All reported runs use $\lambda = 0.98$, which places greater weight on progress made earlier in the window. A frame is retained if and only if its windowed advantage is strictly positive:

$$\mathrm{keep}(t) \iff A_t > 0. \tag{27}$$

This rule retains frames followed by net forward progress within the window and discards frames whose windowed progress return is zero or negative.

Filtering and training hyperparameters. The evaluated Filtered BC setup uses $\lambda = 0.98$ and retains exactly those frames with strictly positive windowed progress returns. Table 17 summarizes its filtering and standard BC optimization settings.

Table 17: **Hyperparameters.** Filtering and optimization settings for the evaluated Franka setup, which retains frames with $A_t > 0$ and uses $\lambda = 0.98$.

Hyperparameter	Value
Tasks	4 (drawer, box, bread, steak)
Retention criterion	$A_t > 0$
Window W	50 steps
Discount λ	0.98
Action horizon H	16
Action dimension	32
Batch size	64
Peak LR / schedule	3×10^{-5}, cosine (2k-step warmup, 20k-step decay $\rightarrow 3 \times 10^{-6}$)
EMA decay	0.99
Training steps	20,000
Initialization	$\pi_{0.5}$ base

Progress-source ablation. For the Franka setup, we replace TimeValue-derived progress scores with Robometer success scores while keeping all filtering and training hyperparameters fixed. This comparison isolates the effect of the progress signal on data selection and subsequent policy learning.

E.7 Inference-time Value Guidance

We evaluate value guidance on Bread Basket Placement and Steak Serving with a Spatula (Section 5), using the Cosmos3-Edge policy jointly fine-tuned on all four tasks (Section E.1). The policy and reward model remain frozen throughout evaluation. At each decision step, we sample several candidate actions and use TimeValue to select the one whose predicted outcome makes the most progress toward the goal.

Value-guided action selection. Given the current observation o_t and instruction ℓ, the policy samples $N = 4$ candidate action chunks $\{a_t^{(i)}\}_{i=1}^{N}$ with different noise seeds. Each candidate contains $H = 16$ actions and is jointly generated with a video of $T = 17$ frames: the current conditioning frame followed by 16 imagined future frames. Since TimeValue takes visual observations rather than actions as input, we evaluate each action chunk through its associated video.

Let $\Phi_j^{(i)} = -v_j^{(i)}$ denote the potential assigned to frame j of candidate i, where $v_j^{(i)}$ is the absolute temporal-distance prediction conditioned on ℓ (Equation 12). We score a candidate by the increase

in potential from the current frame to the final predicted frame:

$$\widehat{Q}_i = \Phi_{T-1}^{(i)} - \Phi_0^{(i)}, \qquad i^\star = \arg\max_{i \in \{1,\dots,N\}} \widehat{Q}_i. \tag{28}$$

The score measures the predicted reduction in remaining task time over the action horizon. Taking a difference also cancels additive offsets shared by the current and future frame estimates. We execute the selected chunk $a_t^{(i^\star)}$ and repeat the procedure at the next policy query. This gives a model-based best-of-N selection rule without training an action-conditioned critic.

Scoring imagined trajectories. For these two single-arm tasks, we remove reflection padding after VAE decoding and extract the `left_side` third-person view from the left half of the camera mosaic. We score $K = 5$ frames per candidate, retaining frame 0 and uniformly sampling four future frames with rounded indices, which gives $\{0, 1, 6, 11, 16\}$. All candidate clips are scored in one batch with the original task instruction. Each clip is processed in a single forward pass, rather than repeatedly evaluating its successive prefixes. For the Robometer comparison, we substitute its native progress predictions for the TimeValue potentials while keeping the policy, candidate budget, and selection rule unchanged.

Inference hyperparameters. We use the multi-task SFT checkpoint at $40{,}000$ training steps and the sampling configuration in Table 18. Both guided variants use the same configuration; only the model used to score the imagined trajectories differs.

Table 18: **Value-guidance hyperparameters.** Sampling and action-selection settings for evaluating the frozen multi-task Cosmos3-Edge policy on Bread and Steak.

Hyperparameter	Value
Policy checkpoint	Multi-task SFT, $40{,}000$ steps
Candidate count N	4
Action horizon H	16
Executable action dimension	8 (single-arm)
Action representation	Absolute joint and gripper commands
Video length T	17 (one current and 16 future frames)
Scored frames K	5, at indices $\{0, 1, 6, 11, 16\}$
Sampler / denoising steps	UniPC / 4
Classifier-free guidance scale	3.0
Sampler shift	5.0
Conditioning frame rate	30 FPS
Score aggregation / selection	Potential gain / $\arg\max$
Candidate seeds (deterministic mode)	$\{0, 1, 2, 3\}$
Executed steps / control frequency	16 / 30 Hz

Inference cost. In the single-arm setup with $N = 4$ and $K = 5$, candidate generation and VAE decoding take approximately 1.10 s, and value scoring takes 0.33 s, giving a total latency of about 1.44 s per policy query. The unguided policy takes approximately 0.15 s. Reducing N to 2 lowers guided inference to about 0.8 s. A 16-step chunk takes 0.53 s to execute at 30 Hz, so the synchronous implementation incurs waiting time between chunks despite any reduction in the number of chunks needed to complete a task.

F Compute Resources

Offline IQL and SFT each use 2 GPUs with 80GB of memory each. IQL assigns one FSDP device to each replica, whereas SFT uses two-way model sharding. Training one task requires approximately 16 hours for offline IQL and 6 hours for SFT. Online DSRL runs on a single x86 server equipped with two Intel Xeon Platinum 8575C processors (48 physical cores per socket with hyper-threading enabled; 192 logical cores in total), 1.5 TiB of system memory, and eight NVIDIA GeForce RTX 5090 GPUs with 32 GB of memory each. The server has a dual-socket NUMA topology and runs NVIDIA driver 570.211.01 with CUDA 12.8. Each task uses $6{,}000$ training steps and 60 online

rollout trajectories, requiring approximately 1.5 hours. All online DSRL experiments use single-node execution.