TTPE: TEST-TIME POLICY EVOLUTION FOR AGENTIC TEXT-TO-VIDEO REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

A single text-to-video (T2V) generation rarely satisfies every semantic requirement of a challenging prompt, so recent work spends additional video generations at test time on iterative generate–evaluate–refine loops. Existing methods refine each prompt in isolation under fixed strategies, repeatedly paying for video synthesis while discarding the experience accumulated from previous refinement episodes. We instead cast refinement as a budget-constrained sequential decision problem: each action shapes both the next video and the budget left for later correction. Conventional reinforcement learning is ill-suited here, as it requires costly video-generation rollouts and yields policies that must be retrained for each new generator. We introduce **TTPE** (Test-Time Policy Evolution), a training-free framework that accumulates completed refinement trajectories in an episodic memory and reuses them to decide whether another generation is worth spending and which refinement action to take. From similar past trajectories, TTPE estimates two decoupled long-horizon advantages per action family: a quality advantage capturing expected future quality gain, and a cost advantage capturing expected future generation cost. Rather than collapsing them into a single objective, TTPE resolves their trade-off only at decision time, pruning Pareto-dominated actions in the quality–cost space and selecting among the survivors under the remaining budget. Across VBench-2.0 and StoryEval with three video generators, TTPE consistently improves the quality–cost frontier over prior iterative refinement, achieving the best aggregate quality in every evaluated generator–benchmark setting, with gains of up to 5.38 points and generation-cost reductions of up to 65.5% relative to fixed-round refinement. Code and implementation details are available at `https://anonymous.4open.science/r/ttpe-C245/`.

1 INTRODUCTION

Recent text-to-video (T2V) models have achieved remarkable progress in visual quality, temporal consistency, and text alignment. Nevertheless, single-shot generation remains unreliable for challenging prompts involving complex entity relations, fine-grained actions, and multi-stage temporal structures. Recent methods therefore introduce additional test-time computation through iterative *generate–evaluate–refine* procedures, including resampling, prompt rewriting, and conditioning-space optimization (Xue et al., 2025; Kim et al., 2026; Long et al., 2026). Although these methods demonstrate that additional inference-time generation can improve video quality, they typically allocate compute through a predetermined number of refinement rounds. Such a fixed schedule ignores a fundamental property of test-time video generation: the marginal utility of an additional generation is highly state-dependent. The refinement process must therefore decide how to allocate a limited generation budget based on the current state and the expected value of subsequent actions. This naturally casts iterative T2V refinement as a budget-constrained sequential decision problem.

A second limitation of existing refinement procedures is that they typically treat each video independently. Each refinement episode reveals how different interventions affect particular residual failure patterns, but this information is discarded once the episode ends (Lee et al., 2026; Huang et al., 2026a). Yet similar failures recur across prompts: an intervention that successfully repairs a missing entity, incorrect interaction, or temporal inconsistency in one episode may also be informative in another. Rather than repeatedly solving every video from scratch, a refinement system should reuse previous trajectories to improve its future compute-allocation decisions.

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

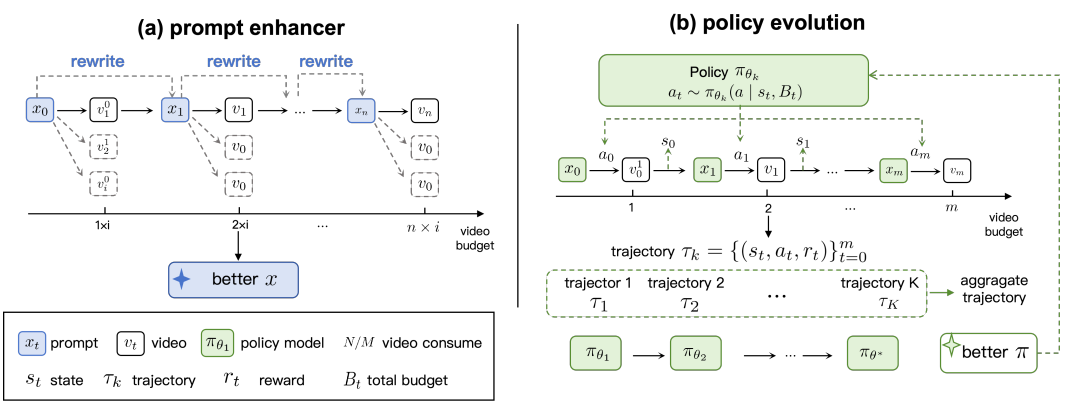

Figure 1: Comparison between conventional iterative refinement and TTPE. Conventional methods repeatedly improve individual outputs using a fixed refinement schedule. TTPE instead reuses cross-episode trajectories to improve the policy that allocates test-time generation compute.

Learning such a policy through conventional reinforcement learning, however, is prohibitively expensive. Existing approaches collect video-generation outcomes to train prompt optimizers, requiring costly synthesis and evaluation to obtain training signals (Gao et al., 2025b; Cheng et al., 2025). The learned optimizers may also be sensitive to changes in the generator, evaluator, or failure distribution at deployment. Extending such learning to sequential refinement would require dedicated generation rollouts to train a policy or critic before any inference-time savings could be realized. These limitations motivate a training-free alternative that improves decisions by reusing trajectories already collected during ordinary refinement, without gradient updates or additional policy-training rollouts.

Based on this insight, we introduce **TTPE**, a training-free, memory-based method for adaptive test-time compute allocation in T2V refinement. Casting refinement as a sequential decision problem, TTPE retrieves relevant cross-episode trajectories based on the current alignment state and residual failure pattern to estimate each action family's expected discounted quality improvement and future generation consumption. These estimates enable Pareto filtering in the quality–budget space and budget-conditioned scoring of non-dominated actions. Our main contributions are:

- We formulate iterative T2V refinement as an adaptive, budget-constrained sequential decision problem at test time, accounting for how each generation affects both immediate quality and future opportunities for correction.
- We use cross-episode retrieval to estimate long-term quality gains and generation costs, guiding refinement and stopping through Pareto filtering and budget-conditioned scoring without policy training.
- We evaluate TTPE on VBench-2.0 and StoryEval with Wan-2.1-T2V-1.3B, HunyuanVideo-1.5, and MiniMax-H3. In our online evaluation setting, TTPE achieves the highest aggregate quality in every evaluated generator–benchmark setting while using fewer video generations than fixed-round refinement.

2 Method

2.1 Preliminaries

We abstract the video optimization process as a finite-horizon Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, T)$. Here, $\mathcal{S}$ denotes the set of all possible states in the video optimization process. Each state is represented by information including the current prompt, quality evaluation, and remaining budget. The action space $\mathcal{A}$ contains interventions for optimizing video generation. The transition probability $P(s_{t+1} \mid s_t, a_t)$ is induced jointly by the video generator and the evaluation process after an action is executed. The immediate reward $r(s_t, a_t, s_{t+1})$ is obtained from the newly

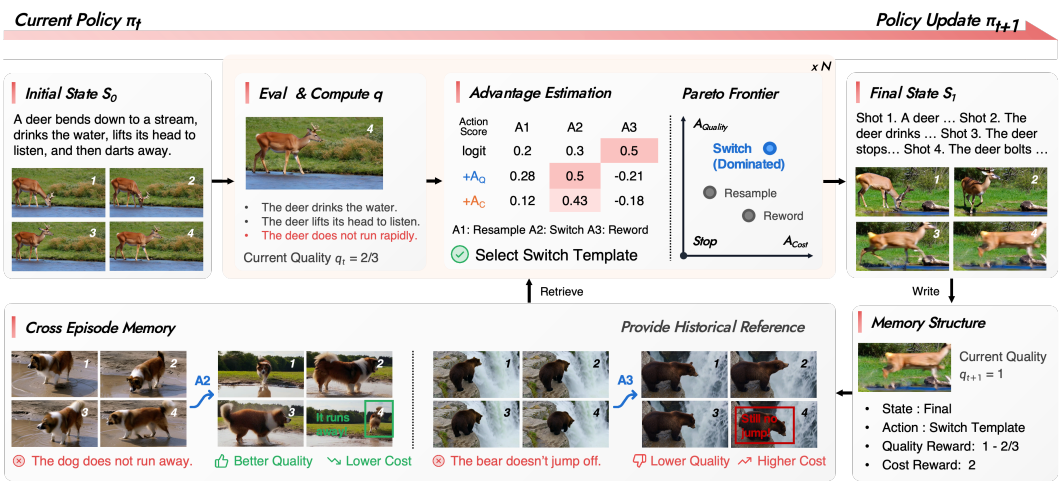

Figure 2: Overview of TTPE. Iterative T2V refinement is formulated as a budget-constrained decision process, where each refinement action yields immediate quality feedback while consuming one unit of generation budget. TTPE separately estimates long-term quality return and future generation cost, filters dominated actions via Pareto optimality, and performs budget-conditioned selection with endogenous stopping.

generated result and its evaluation feedback. The discount factor satisfies $\gamma \in [0, 1]$, and T denotes the maximum episode horizon. At time t, the discounted return is defined as

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k, \qquad r_k = r(s_k, a_k, s_{k+1}). \tag{1}$$

We use the following standard definitions of the state-action value function Q^π, the state-value function V^π, and the advantage function A^π:

$$\begin{aligned} Q^\pi(s_t, a_t) &= \mathbb{E}[G_t \mid s_t, a_t], \\ V^\pi(s_t) &= \mathbb{E}[G_t \mid s_t], \\ A^\pi(s_t, a_t) &= Q^\pi(s_t, a_t) - V^\pi(s_t), \end{aligned} \tag{2}$$

where the expectations are taken over trajectories induced by policy π and transition dynamics P.

Here $V^\pi(s)$ is the expected return from state s under π, $Q^\pi(s, a)$ is that return after taking action a and then following π, and $A^\pi(s, a)$ measures the value of a relative to the average behavior of π at s.

Building on these standard definitions, TTPE instantiates this generic formulation for iterative T2V refinement. Specifically, Sec. 2.2 specifies the state, action space, and trajectory structure, Sec. 2.3 defines element-level quality evaluation, the reward, and the two return-to-go axes, Sec. 2.4 estimates the corresponding advantages from retrieved trajectories, Sec. 2.5 selects actions by budget-conditioned Pareto filtering, and Sec. 2.6 assembles them into the test-time loop.

2.2 Decision Process

We instantiate the MDP horizon T of Sec. 2.1 as the maximum number of rounds, and denote the total generation budget as B. An episode begins with an initial video generation and then makes up to $B - 1$ refinement decisions, indexed $t = 0, \ldots, B - 2$, consuming at most B video generations in total. The remaining-budget fraction is $\rho_t = (B - 1 - t)/(B - 1) \in (0, 1]$, where $B - 1 - t$ is the number of refinement decisions still available at step t. The state at step t is

$$s_t = \left(x_t,\, q_t,\, q_t^{\max},\, \rho_t\right) \tag{3}$$

where x_t is the current prompt, q_t is the current video quality, detailed in Sec. 2.3, $q_t^{\max} = \max_{i=0}^{t} q_i$ is the running best quality, and ρ_t is the remaining-budget fraction. At $t = 0$, before any decision,

we have the initial video v_0 with quality q_0, and $q_0^{\max} = q_0$, as illustrated in Fig. 2. A trajectory is a sequence

$$\tau = (s_t, a_t, r_t)_{t=0}^{L} \tag{4}$$

where $L \leq B - 2$ is the step of the last action in the trajectory, and r_t is the immediate reward, detailed in Sec. 2.3.

Actions come from a closed vocabulary: `resample`, the prompt edits `reword`, `split`, `reorder`, `prune`, and `switch_repr` over five caption representations, and the `stop` actions. Every non-stop action synthesizes exactly one video and consumes one unit of budget, while `stop` actions consume none. We focus our cost accounting in the main text on video generation, as it constitutes the dominant computational expense. The costs associated with text editing and other non-generation operations are detailed in the appendix. Detailed action examples are provided in the appendix.

2.3 Element-Level Reward and Two Return-to-Go Axes

Every action yields a complete video, so feedback is dense. In line with widely used VQA-based evaluation protocols in multimodal research Lin et al. (2024); Singh & Zheng (2023), we decompose the original prompt once, before any generation, into m frozen atomic requirements spanning object presence, properties, spatial arrangement, object motion, camera movement, scene transitions, and temporal ordering. A strict judge returns an independent binary verdict per element, and video quality is the number satisfied, $q_t \in \{0, \ldots, m\}$. Freezing the decomposition prevents the evaluator from moving the target mid-episode and exposes *which* requirement failed, which is what makes states comparable across prompts in Sec. 2.4.

The immediate reward for a generating action is defined as the improvement over the running best quality:

$$r_t = q_{t+1} - q_t^{\max}, \tag{5}$$

followed by the update $q_{t+1}^{\max} = \max(q_t^{\max}, q_{t+1})$. Measuring against the running best $q_t^{\max}$ prevents an action from being credited for merely undoing a previous regression, and r_t may be negative. This reward therefore favors trajectories that reach higher-quality outputs efficiently, while penalizing intermediate regressions that do not surpass the running best.

Because an action changes both the state that follows and the number of decisions still affordable, we propagate two return-to-go axes by backward recursion after the episode terminates at step $L \leq B-2$:

$$G_t^Q = r_t + \gamma\, G_{t+1}^Q, \qquad G_t^C = \mathbf{1}_{a_t \text{ generates}} + G_{t+1}^C, \tag{6}$$

with $\gamma = 0.95$ and boundary conditions $G_{L+1}^Q = G_{L+1}^C = 0$, since no further reward or generation cost is incurred after the episode terminates. Quality follows the standard discounted return, whereas generation cost is undiscounted since it directly counts the number of video syntheses consumed.

2.4 Retrieval-Based Dual Advantage Estimation

After each episode, memory stores its transitions $\mathcal{H} = \{(d_i, a_i, r_i, G_i^Q, G_i^C)\}$, where d_i denotes the textual state representation used for retrieval. It is populated purely online: the reused trajectories are exactly those deployment would otherwise discard, and there is no offline corpus. Each textual state representation d_i is embedded using a word 1–2-gram hashing vectorizer, and similarity is measured by cosine similarity. We take the $K = 10$ most similar entries with positive similarity w_i.

State baselines and action values are similarity-weighted means of the stored return-to-go axes, with

$$V_*(s) = \frac{\sum_{i \in \mathcal{N}} w_i\, G_i^*}{W}, \qquad Q_*(s,a) = \frac{\sum_{i \in \mathcal{N}_a} w_i\, G_i^*}{W_a}, \qquad * \in \{Q, C\}, \tag{7}$$

where $W = \sum_{i \in \mathcal{N}} w_i$ sums over all retrieved neighbors and $W_a = \sum_{i \in \mathcal{N}_a} w_i$ sums over those that took action a. When the neighborhood is empty or no neighbor has positive similarity, we set all advantages to zero. These estimates pool trajectories from different historical episodes and represent empirical predictions rather than values under a fixed policy π. We define the two task-specific advantages as follows:

$$A_Q(s,a) = Q_Q(s,a) - V_Q(s), \qquad A_C(s,a) = V_C(s) - Q_C(s,a). \tag{8}$$

Algorithm 1 TTPE Test-Time Refinement

Require: Prompt x_0, total budget B, frozen generator and agent π_{base}, memory $\mathcal{H}$, strength β, discount γ
Ensure: Delivered video $v^\star$
Decompose x_0 once into m frozen elements; generate initial video v_0, evaluate to get q_0; set $q_0^{\max} \leftarrow q_0$
for $t = 0$ **to** $B - 2$ **do**
  Build s_t with $\rho_t \leftarrow (B - 1 - t)/(B - 1)$ and its document d_t
  Freeze a memory snapshot excluding this episode and prompt; retrieve $\mathcal{N}_t$
  Get candidates $\mathcal{C}_t$ with confidences from π_{base}
  Estimate $\widehat{A}_Q, \widehat{A}_C$ per action via Eq. 9
  Discard Pareto-dominated actions; $a_t \leftarrow \arg\max_c \text{score}(c)$ by Eq. 11
  if a_t is stop **then**
    Record $(s_t, a_t, 0)$; **break**
  end if
  Execute a_t to get v_{t+1}; evaluate to get q_{t+1}
  $r_t \leftarrow q_{t+1} - q_t^{\max}$; update $q_{t+1}^{\max} \leftarrow \max(q_t^{\max}, q_{t+1})$; record (s_t, a_t, r_t)
end for
$v^\star \leftarrow$ video with the largest q, earliest on ties
Compute $\{G_t^Q, G_t^C\}_{t=0}^{L}$ by Eq. 6; commit to $\mathcal{H}$ atomically
return $v^\star$

The retrieval-conditioned quality and cost advantages serve as empirical signals for action selection, with larger values favoring quality improvement and lower generation cost, respectively. Before selection, we normalize each advantage by the scale of the retrieved returns, clip it to $[-1, 1]$, and reduce its magnitude when the retrieved evidence for action a is weak:

$$\widehat{A}_* = \underbrace{\frac{W_a}{W_a + \nu_0}}_{\eta(s,a)} \cdot \text{clip}\left(\frac{A_*(s,a)}{\max\{0.01,\ \max_{i \in \mathcal{N}} |G_i^* - V_*(s)|\}}, -1, 1 \right), \qquad \nu_0 = 1. \tag{9}$$

The gate $\eta \in [0, 1)$ discounts an action supported by a single weak neighbor and trusts one backed by many similar states. When no neighbor took a we set $\widehat{A}_Q = \widehat{A}_C = 0$ and defer to the frozen agent.

2.5 Pareto Filtering and Budget-Conditioned Selection

The frozen agent may produce duplicate or near-duplicate candidate prompts. We merge candidates with the same normalized prompt and retain the one with the highest confidence. The confidences of the remaining candidates are then renormalized to sum to one, yielding the base policy $\pi_{\text{base}}(c \mid s)$. We discard action families Pareto-dominated in normalized quality–cost advantage space:

$$a' \succ_s a \iff \widehat{A}_Q(s, a') \geq \widehat{A}_Q(s, a), \qquad \widehat{A}_C(s, a') \geq \widehat{A}_C(s, a), \tag{10}$$

with at least one strict inequality.

Among the remaining candidates, we condition selection on the remaining-budget fraction ρ_t defined in Sec. 2.2:

$$\text{score}(c) = \log \pi_{\text{base}}(c \mid s) + \left[\rho_t \widehat{A}_Q(s, a) + (1 - \rho_t) \widehat{A}_C(s, a) \right]. \tag{11}$$

2.6 Test-Time Loop

Algorithm 1 summarizes the complete test-time refinement procedure. An episode refines until a `stop` action is chosen or the budget is exhausted, then delivers the video with the largest aligned-element count, earliest on ties so no generation is spent on an improvement already available. Only afterwards are the transitions finalized into the two axes of Eq. 6 and committed, at which point they inform every subsequent prompt. Per step, TTPE adds only one vectorizer transform, a top-K cosine scan, and a dominance test over a handful of actions, all negligible beside a single synthesis. That is the point: the policy is improved with compute that is free relative to the compute it allocates.

Table 1: **Quantitative comparisons on VBench-2.0.** Bold indicates the best result for each dimension. Gen. Cost reports the total number of video generations.

Category	Video Model	Method	Dynamic Spatial Relationship	Dynamic Attribute	Motion Order Understanding	Human Interaction	Composition	Average	Total Video Generations
Controllability & Creativity	Wan-2.1	Baseline	22.22	18.32	18.86	55.00	41.78	31.23	412
		VPO	23.19	51.65	35.35	70.00	42.20	44.48	412
		VQQA	34.78	55.49	34.34	72.00	50.50	49.42	2,562
		PRIS	**46.38**	59.34	35.35	**82.00**	52.20	55.05	8,240
		VISTA	17.39	54.95	32.32	64.00	46.98	43.13	8,240
		Ours	43.48	**75.82**	**38.38**	68.00	**67.17**	**58.57**	2,845
	HunyuanVideo	Baseline	24.15	23.08	17.17	55.67	42.44	32.50	412
		VPO	34.78	48.35	26.26	77.00	56.64	48.61	412
		VQQA	34.00	54.95	28.79	74.00	58.60	50.07	2,240
		PRIS	31.88	59.34	30.30	**79.00**	60.88	52.28	4,120
		VISTA	34.30	56.04	26.87	70.71	56.43	48.87	4,120
		Ours	**42.03**	**64.84**	**36.36**	75.00	**62.99**	**56.24**	1,843
	MiniMax-H3	Baseline	46.38	54.95	27.27	59.00	54.21	48.36	412
		VPO	42.03	67.03	32.32	77.00	64.40	56.56	412
		VQQA	46.38	67.58	35.35	76.00	61.95	57.45	710
		PRIS	49.28	65.93	32.32	**82.00**	60.35	57.98	1,648
		VISTA	46.86	68.86	39.39	70.67	63.81	57.92	1,648
		Ours	**52.28**	**72.53**	**41.28**	79.00	**71.04**	**63.23**	869

Category	Video Model	Method	Camera Motion	Motion Rationality	Mechanics	Material	Thermotics	Average	Gen. Cost
Commonsense & Physics	Wan-2.1	Baseline	17.28	32.18	57.85	34.94	56.78	39.81	323
		VPO	31.48	37.93	61.36	**83.33**	60.00	54.82	323
		VQQA	34.72	39.26	64.18	79.42	61.08	55.73	2,240
		PRIS	**50.00**	39.66	69.05	65.79	56.25	56.15	6,460
		VISTA	41.67	**41.38**	64.29	81.82	61.70	58.17	6,460
		Ours	46.30	**41.38**	71.74	**83.33**	**75.00**	**63.55**	3,261
	HunyuanVideo	Baseline	55.86	32.76	60.50	37.08	38.13	44.87	323
		VPO	63.89	44.83	63.64	69.23	60.87	60.49	323
		VQQA	64.35	45.91	69.26	72.44	60.83	62.56	1,464
		PRIS	60.19	48.28	**79.55**	62.16	**71.74**	64.38	3,230
		VISTA	67.28	38.10	72.36	81.13	68.84	65.54	3,230
		Ours	**70.37**	**50.00**	68.00	**76.92**	68.89	**66.84**	1,482
	MiniMax-H3	Baseline	62.04	53.45	67.50	48.39	45.83	55.44	323
		VPO	69.44	58.62	68.89	83.33	64.58	68.97	323
		VQQA	68.10	58.14	76.38	81.72	67.95	70.46	597
		PRIS	**70.37**	55.17	82.61	73.81	**79.59**	72.31	1,292
		VISTA	65.74	59.20	77.04	83.02	77.03	72.41	1,292
		Ours	65.74	**62.07**	**84.56**	**86.25**	77.55	**75.23**	799

3 EXPERIMENTS

Experimental setup. Following PRIS (Kim et al., 2026), we use a VBench-2.0 subset of 735 prompts across ten dimensions. We also evaluate on 423 StoryEval prompts for multi-event completion. Experiments use three frozen video generators: Wan2.1-T2V-1.3B, HunyuanVideo-1.5, and MiniMax-H3. All iterative methods use the same maximum per-prompt generation budget for each video generator. Considering model capabilities and generation costs, we set the maximum generation budget per prompt to 20 for Wan-2.1, 10 for HunyuanVideo-1.5, and 4 for MiniMax-H3. TTPE operates online at test time, with memory updated sequentially and retained across prompts within each evaluation stream; each prompt can access only trajectories collected from preceding prompts. For VBench-2.0, memory is reset between evaluation dimensions and does not accumulate across dimensions. For the ablation study, we use HunyuanVideo-1.5 and evaluate all variants on a 60% subset of StoryEval, randomly sampled in proportion to the original difficulty-level distribution.

3.1 MAIN RESULT

Results on VBench-2.0. As shown in Table 1, TTPE achieves the highest group-average scores across all three generators. On Controllability & Creativity, it scores 58.57, 56.24, and 63.23 on Wan-2.1, HunyuanVideo, and MiniMax-H3, respectively; on Commonsense & Physics, the corresponding scores are 63.55, 66.84, and 75.23. Compared with the strongest competing method in each group, TTPE improves the group-average score by 1.30–5.38. Meanwhile, TTPE reduces the total number of video generations relative to PRIS and VISTA by 58.5% on Wan-2.1, 54.8% on HunyuanVideo, and 43.3% on MiniMax-H3.

Results on StoryEval. Table 2 shows that TTPE achieves the highest overall Completion Score and Full Story rate across all three generators. Completion Scores reach 37.47, 48.60, and 68.81 on Wan-2.1, HunyuanVideo, and MiniMax-H3, respectively, exceeding the strongest competing results by 0.85, 2.54, and 2.32 points. Full Story rates improve to 10.17, 17.49, and 39.72, indicating better

Table 2: **Results on StoryEval.** Completion scores (%) across event counts, semantic categories, and difficulty levels, with overall event and full-story completion rates. Gen. Cost reports the total number of video generations. Bold indicates the best results.

Video Model	Method	Event Count			Semantic Category					Difficulty			Overall		Total Video Generations
		2 events	3 events	4 events	Human	Animal	Object	Retrieval	Creative	Easy	Medium	Hard	Comp. Score	Full Story	
Wan-2.1	Baseline	19.92	18.96	21.43	20.24	22.02	14.96	25.00	14.39	33.42	17.94	8.75	19.42	1.65	423
	VPO	28.95	25.83	25.89	26.81	30.25	23.80	31.85	26.59	40.75	25.22	16.42	26.81	3.55	423
	VQQA	37.59	**36.90**	29.46	36.44	**38.79**	36.14	37.10	38.11	50.42	35.65	25.00	36.62	8.98	3,728
	PRIS	33.08	30.53	31.25	32.09	34.49	29.33	40.86	28.79	46.58	29.78	19.75	31.38	7.57	8,460
	VISTA	37.79	33.46	37.50	34.60	35.53	32.55	38.84	30.43	48.50	35.01	21.72	35.09	6.15	8,460
	Ours	**37.88**	36.82	**41.67**	**38.52**	35.37	**39.89**	**41.80**	**39.60**	**53.09**	**35.67**	**26.18**	**37.47**	**10.17**	4,221
HunyuanVideo	Baseline	31.58	27.74	26.79	29.55	32.22	25.09	31.45	20.76	41.25	29.41	15.33	28.88	4.02	423
	VPO	36.09	31.42	33.04	32.30	35.96	30.80	33.20	28.26	44.33	34.08	19.25	33.00	6.62	423
	VQQA	43.61	**46.56**	36.61	41.46	49.75	43.92	38.98	43.48	56.83	42.60	**38.42**	44.98	12.53	2,260
	PRIS	36.47	37.02	41.07	33.95	40.86	34.58	47.18	26.14	54.08	37.22	19.92	37.12	9.46	4,230
	VISTA	52.63	43.33	40.18	47.43	47.73	43.42	42.20	42.88	**60.42**	45.02	34.00	46.06	15.13	4,210
	Ours	**54.51**	45.93	**45.54**	**49.69**	**50.86**	**45.44**	**49.19**	**44.24**	60.33	**49.93**	33.92	**48.60**	**17.49**	2,410
MiniMax-H3	Baseline	52.26	47.33	55.36	50.36	48.23	49.91	49.19	45.15	65.67	49.74	32.42	49.41	19.62	423
	VPO	57.89	54.20	43.75	59.01	55.20	52.62	51.88	57.80	65.92	54.67	43.42	54.67	23.64	423
	VQQA	65.04	63.99	54.46	64.54	64.65	64.46	60.48	67.20	73.33	62.14	57.50	63.69	33.33	805
	PRIS	65.79	59.03	**65.18**	65.01	60.81	61.51	**65.19**	60.15	70.92	62.86	49.33	61.56	32.39	1,692
	VISTA	70.68	64.50	59.82	66.87	**70.05**	63.81	64.65	67.58	75.92	68.20	53.25	66.49	37.59	1,692
	Ours	**73.68**	**67.30**	**65.18**	**73.40**	67.02	**68.92**	63.84	**72.95**	**76.83**	**69.10**	**60.17**	**68.81**	**39.72**	975

completion of entire event sequences. These gains require only 10.0, 5.7, and 2.3 generations per prompt, compared with 20, 10, and 4 for PRIS and VISTA. Overall, TTPE improves multi-event completion while requiring substantially fewer generation calls.

3.2 Ablations

Table 3: Ablation study of the quality advantage $\mathcal{A}_Q$, budget advantage $\mathcal{A}_C$, and Pareto pruning.

$\mathcal{A}_Q$	$\mathcal{A}_C$	Pareto	Completion ↑	Avg. Generations ↓	Candidate Conflict	Selected Conflict ↓	Pareto Pruned
✓	✗	✗	**47.58**	6.30	19.76%	23.73%	-
✗	✓	✗	41.17	**3.97**	27.84%	28.27%	-
✓	✓	✗	46.92	5.77	26.22%	32.18%	-
✓	✓	✓	47.42	4.47	25.00%	**17.99%**	31.80%

Ablation dual advantage and pareto. Table 3 examines the contribution of $\mathcal{A}_Q$, $\mathcal{A}_B$, and Pareto pruning. Using $\mathcal{A}_Q$ alone achieves the highest completion score of 47.58, but requires 6.30 generations per prompt, whereas $\mathcal{A}_B$ alone substantially reduces the generation cost to 3.97 at the expense of a marked performance drop to 41.17. Combining $\mathcal{A}_Q$ and $\mathcal{A}_B$ recovers most of the completion performance (46.92), but introduces a high selected-conflict rate of 32.18%. Incorporating Pareto pruning effectively resolves this trade-off: the full model maintains a comparable completion score of 47.42 while reducing the average generation cost to 4.47 and the selected-conflict rate to 17.99%. In addition, 31.80% of candidates are removed through Pareto pruning. These results suggest that $\mathcal{A}_Q$ and $\mathcal{A}_B$ play complementary roles in balancing generation quality and efficiency, while Pareto pruning is critical for filtering conflicting candidates and achieving a better overall trade-off.

Table 4: Ablation study on different ways of utilizing retrieved memory.

Memory Utilization	Performance (%) ↑	Event Comp. (%) ↑	Avg. Video Generations / Prompt ↓	Effective Exploration (%) ↑	Repeated Failure (%) ↓
No Memory	41.83	13.00	6.55	19.10	70.25
Reward-based Memory	45.35	17.00	5.52	23.18	64.37
Advantage-based Memory (Ours)	**47.42**	**21.00**	**4.47**	**27.08**	**59.65**

Ablation on Memory Utilization. Table 4 studies how retrieved memory is used for decision making. No Memory removes retrieved experience, Reward-based Memory directly uses the rewards

of retrieved trajectories, while our full design uses advantage estimates to capture the utility of actions beyond their raw rewards. Reward-based Memory improves performance from 41.83% to 45.35% and reduces the average number of video generations per prompt from 6.55 to 5.52. Using advantage-based memory further improves performance to 47.42% and event completion to 21.00%, while reducing the generation cost to 4.47 per prompt. It also achieves the highest effective exploration rate of 27.08% and the lowest repeated-failure rate of 59.65%. These results show that historical experience is more effectively exploited when actions are evaluated through advantage estimates rather than raw rewards, leading to better quality–efficiency trade-offs.

3.3 Analysis

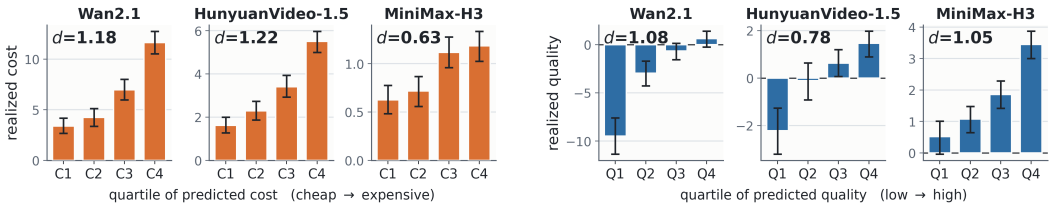

Figure 3: Predicted axes are reflected in realized outcomes across generators. Decisions are binned into quartiles by predicted cost (left; C1–C4, low → high) or predicted quality (right; Q1–Q4, low → high). Bold numbers denote the standardized highest-to-lowest quartile gap.

Prediction–outcome alignment. We test whether predictions made at the initial decision step anticipate downstream outcomes. Prompts are ranked by their predicted cost and quality scores and divided into quartiles. As shown in Fig. 3, higher predicted-cost quartiles incur larger realized generation costs, while higher predicted-quality quartiles achieve better realized quality across all three generators. The separation between the lowest and highest quartiles shows that the initial predictions contain meaningful information about later outcomes.

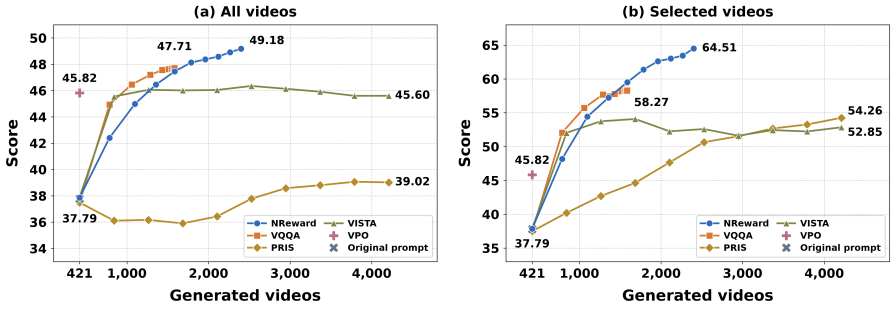

Figure 4: Quality–cost trade-off of different methods. We report the evaluation score as a function of the number of generated videos for (a) all generated videos and (b) selected videos. TTPE achieves a better quality–cost trade-off, reaching higher scores with fewer video generations.

Quality–cost trade-off. Figure 4 further compares the methods under different generation budgets. TTPE achieves a more favorable quality–cost trade-off, reaching higher evaluation scores with substantially fewer generated videos. This trend is consistent for both all generated videos and the selected subset, demonstrating that the performance gains of TTPE are not obtained simply by increasing the number of generation calls.

Action Selection across Optimization Phases. Figure 7 examines how memory utilization affects action selection across optimization phases. With advantage-based memory, *Switch repr.* decreases from 52.5% to 31.6%, while *Reword* increases from 35.9% to 62.3% from early to late optimization. By comparison, no memory continues to favor representation switching, while reward-based memory

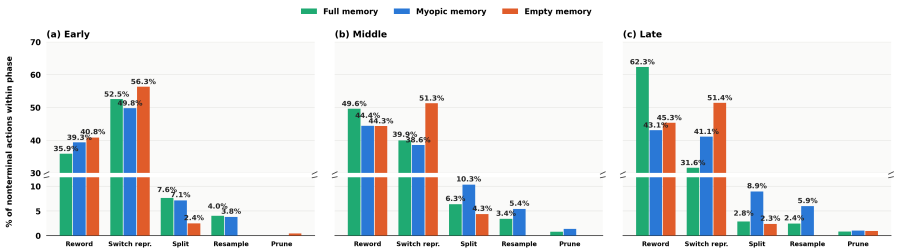

Figure 5: Action distributions across early, middle, and late optimization phases under different memory utilization strategies. Advantage-based memory shows a clearer shift from representation-level exploration to rewording-based refinement.

uses more *Split* and *Resample*. These trends align with the improved exploration efficiency and fewer repeated failures in Table 4.

4 Related Work

Iterative and Agentic Video Refinement. Recent work improves text-to-video generation through iterative prompting, feedback, and correction. PhyT2V (Xue et al., 2025), PRIS (Kim et al., 2026), and VideoRepair (Lee et al., 2026) refine generations using prompt rewriting, aggregated feedback, or selective regeneration. Other methods learn prompt optimization from reward or preference signals, including Prompt-A-Video (Ji et al., 2025), VPO (Cheng et al., 2025), PhyPrompt (Wu et al., 2026), RAPO (Gao et al., 2025a), and RAPO++ (Gao et al., 2025b).

Agentic approaches further coordinate planning, generation, verification, and correction, as in VideoDirectorGPT (Lin et al., 2023), MovieAgent (Wu et al., 2025), ViMax (Huang et al., 2026b), SCMAPR (Yang et al., 2026), GenMAC (Huang et al., 2026a), and VISTA (Long et al., 2026). While these methods support iterative refinement, their inference-time refinement strategies are generally predefined. TTPE instead formulates refinement as adaptive decision making, selecting actions online according to generation feedback and their expected quality gains and generation costs.

Test-Time Learning. Test-time learning enables models or agents to adapt using experience acquired during deployment. Early work performs self-supervised updates on unlabeled test samples to adapt to distribution shifts (Sun et al., 2020), with subsequent approaches improving the stability and robustness of online adaptation (Liu et al., 2021). More recent work extends this paradigm to large language models. TLM (Hu et al., 2025) adapts lightweight model parameters by minimizing perplexity on unlabeled test data, while TTRL (Zuo et al., 2026) performs reinforcement learning at test time using self-generated reward signals. TTTDiscover (Yuksekgonul et al., 2026) further performs problem-specific reinforcement learning, continually optimizing model parameters using experience collected for a particular target problem. TTPE follows the same general principle of improving behavior from deployment-time experience, but does not update the parameters of the underlying models. Instead, it adapts the action-selection policy online by accumulating refinement experience and using timely quality feedback to estimate the long-term quality gains and generation costs of candidate actions.

5 Conclusion

We presented **TTPE**, a training-free method for adaptive test-time compute allocation in text-to-video refinement. By formulating refinement as a budget-constrained sequential decision problem, TTPE estimates the long-term quality gain and generation cost of candidate actions from historical trajectories, and combines Pareto filtering with budget-aware selection to determine both how to refine and when to stop. Experiments across multiple video generators show that TTPE consistently improves the quality–cost trade-off, achieving stronger aggregate quality while requiring fewer generation calls. These results suggest that effective test-time scaling depends not simply on increasing computation, but on allocating additional generations according to their expected long-term quality gains and generation costs.

6 AI USE STATEMENT

Generative AI tools were used during the preparation of this manuscript to assist with language polishing, grammatical correction, and improving the clarity and readability of the text. All AI-assisted suggestions were reviewed and revised by the authors.

REFERENCES

Jiale Cheng, Ruiliang Lyu, Xiaotao Gu, Xiao Liu, Jiazheng Xu, Yida Lu, Jiayan Teng, Zhuoyi Yang, Yuxiao Dong, Jie Tang, et al. Vpo: Aligning text-to-video generation models with prompt optimization. *arXiv preprint arXiv:2503.20491*, 2025.

Bingjie Gao, Xinyu Gao, Xiaoxue Wu, Yujie Zhou, Yu Qiao, Li Niu, Xinyuan Chen, and Yaohui Wang. The devil is in the prompts: Retrieval-augmented prompt optimization for text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3173–3183, June 2025a.

Bingjie Gao, Qianli Ma, Xiaoxue Wu, Shuai Yang, Guanzhou Lan, Haonan Zhao, Jiaxuan Chen, Qingyang Liu, Yu Qiao, Xinyuan Chen, Yaohui Wang, and Li Niu. Rapo++: Cross-stage prompt optimization for text-to-video generation via data alignment and test-time scaling. *arXiv preprint arXiv:2510.20206*, 2025b.

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*, 2025.

Kaiyi Huang, Yukun Huang, Xuefei Ning, Zinan Lin, Yu Wang, and Xihui Liu. Genmac: compositional text-to-video generation with multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 5049–5057, 2026a.

Lingxuan Huang, Sizhe He, Hengji Zhou, Liqiang Nie, Lianghao Xia, and Chao Huang. Vimax: Agentic video generation. *arXiv preprint arXiv:2606.07649*, 2026b.

Yatai Ji, Jiacheng Zhang, Jie Wu, Shilong Zhang, Shoufa Chen, Chongjian Ge, Peize Sun, Weifeng Chen, Wenqi Shao, Xuefeng Xiao, et al. Prompt-a-video: Prompt your video diffusion model via preference-aligned llm. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18725–18735. IEEE, 2025.

Subin Kim, Sangwoo Mo, Mamshad Nayeem Rizve, Yiran Xu, Difan Liu, Jinwoo Shin, and Tobias Hinz. Rethinking prompt design for inference-time scaling in text-to-visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22090–22099, 2026.

Daeun Lee, Jaehong Yoon, Jaemin Cho, and Mohit Bansal. Self-correcting text-to-video generation with misalignment detection and localized refinement. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36464–36489, 2026.

Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. Videodirectorgpt: Consistent multi-scene video generation via llm-guided planning. *arXiv preprint arXiv:2309.15091*, 2023.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pp. 366–384. Springer, 2024.

Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö Arik. Vista: A test-time self-improving video generation agent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6021–6032, 2026.

Jaskirat Singh and Liang Zheng. Divide, evaluate, and refine: Evaluating and improving text-to-image alignment with iterative vqa feedback. *Advances in Neural Information Processing Systems*, 36: 70799–70811, 2023.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Yiping Wang, Xuehai He, Kuan Wang, Luyao Ma, Jianwei Yang, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Is your world simulator a good story presenter? a consecutive events-based benchmark for future long video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13629–13638. IEEE, 2025.

Shang Wu, Chenwei Xu, Zhuofan Xia, Weijian Li, Lie Lu, Pranav Maneriker, Fan Du, Manling Li, and Han Liu. Phyprompt: Rl-based prompt refinement for physically plausible text-to-video generation. *arXiv preprint arXiv:2603.03505*, 2026.

Weijia Wu, Zeyu Zhu, and Mike Zheng Shou. Automated movie generation via multi-agent cot planning. *arXiv preprint arXiv:2503.07314*, 2025.

Qiyao Xue, Xiangyu Yin, Boyuan Yang, and Wei Gao. Phyt2v: Llm-guided iterative self-refinement for physics-grounded text-to-video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18826–18836. IEEE, 2025.

Chengyi Yang, Pengzhen Li, Jiayin Qi, Aimin Zhou, Ji Wu, and Ji Liu. Scmapr: Self-correcting multi-agent prompt refinement for complex-scenario text-to-video generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6942–6973, 2026.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, et al. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

A Action vocabulary

At every decision the policy observes the current video and proposes exactly three candidates drawn from a closed vocabulary; each candidate is validated against the current prompt before it enters the set, so an action that does not apply is discarded rather than executed. Four actions edit or re-generate the prompt and one terminates the episode.

`reword` repairs a single unmet element in place, leaving the rest of the prompt untouched. `split` expands one event into k ordered events and is used when the generator has collapsed a temporal progression into a single moment; because it addresses one event, the surrounding prompt is carried over unchanged. `switch_repr` changes only the presentation, re-expressing the same content in one of six caption styles named after formats used in the video-captioning literature (`miradata`, `instancecap`, `vc4vg`) alongside three generic ones (`anchored`, `prose`, `shotlist`); the style is realised from the policy's own prior rather than from a fixed specification. `resample` changes nothing and spends a generation on variance alone. `stop` ends the episode and keeps the current video.

B Computational Resources

Our experimental infrastructure included a cluster of 32 NVIDIA H100 GPUs. To ensure a fair comparison across methods, we consistently used Gemini-3.1 Pro as the language-model backbone for all primary experiments. We summarize the Gemini API workload of the primary comparisons

Table 5: One example per action; **bold** marks what the action changed.

Action	Parameters	Example
`reword`	`event`	*The wooden door handle transformed into a plastic door handle.* → *The wooden door handle transformed into a **bright red, glossy** plastic door handle.*
`split`	`event, k`	*A horse is standing still, then it suddenly starts trotting around in a circle.* → *A horse is standing still, then it suddenly starts trotting. **As it trots, the horse leans and continually turns leftward, tracing a clear circular path in the dirt.***
`switch_repr`	`template`	*A balloon changes from large to small.* → ***Shot 1:** A large balloon fills the center of the frame. **Shot 2:** The balloon's size actively changes, shrinking from large to small. **Shot 3:** The balloon is very small at the end.*
`resample`	—	*A rock changes from red to grey.* → identical prompt, new seed
`stop`	—	*A car changes from red to white.* → episode ends

across Wan, HunyuanVideo-1.5, and MiniMax-H3 on StoryEval (Wang et al., 2025)and VBench-2.0 (Zheng et al., 2025). Across the full primary-run inventory, including retained partial trajectories, the reconstructed workload amounts to approximately 631,000 core Gemini calls. Figure 6 compares the average number of calls per original prompt, using the shared set of prompts completed by all five methods for each generator–benchmark combination.

TTPE requires fewer core Gemini calls than PRIS and VISTA in all six settings. Compared with VQQA, TTPE uses fewer calls on HunyuanVideo-1.5 and MiniMax-H3, and more on Wan. API call counts depend on both the number of generated candidates and the granularity of the verification and refinement procedures.

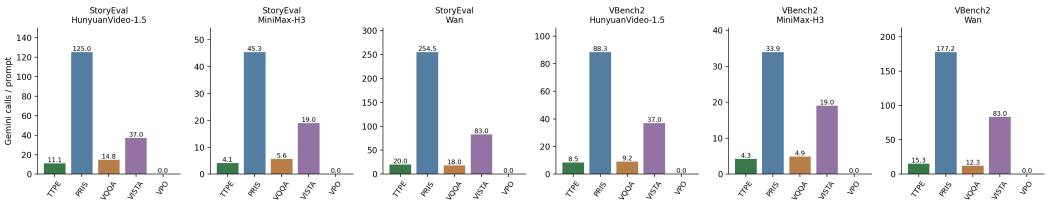

Figure 6: Average number of core Gemini calls per prompt across five refinement methods, three video generators, and two benchmarks. Each comparison uses prompts completed by all five methods. Call counts are reconstructed from retained execution records and exclude input decomposition, retries, and external benchmark evaluation. VPO uses a local prompt optimizer and makes no Gemini calls during refinement.

C Case Study

Qualitative Outcome Figure 7 presents a qualitative case study on the prompt "A dog is sitting in the yard, then it suddenly starts running in circles," which requires both a clear temporal transition in action and sustained trajectory-level motion. This prompt is challenging because a successful refinement must preserve the initial sitting state while introducing a distinct change to dynamic, non-linear movement in the later frames.

Across all three video generators, TTPE more consistently produces videos that exhibit the intended transition from a relatively static initial pose to active running with noticeable changes in direction. In contrast, several baselines capture only part of the instruction: some generate plausible running motion but largely follow a one-directional trajectory, while others show pose or viewpoint

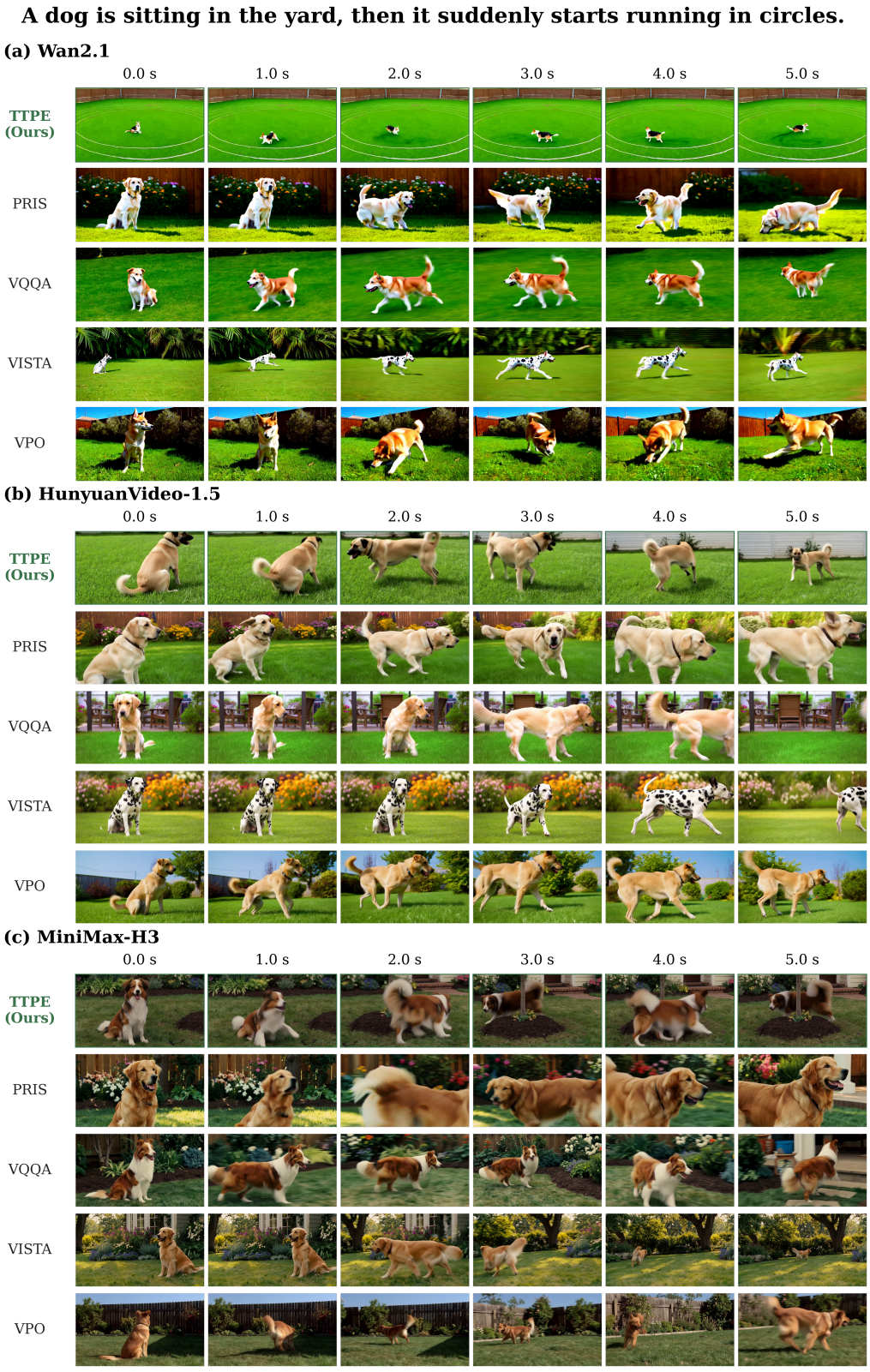

Figure 7: Qualitative comparison of five test-time refinement methods across three T2V generators on a multi-stage motion prompt.

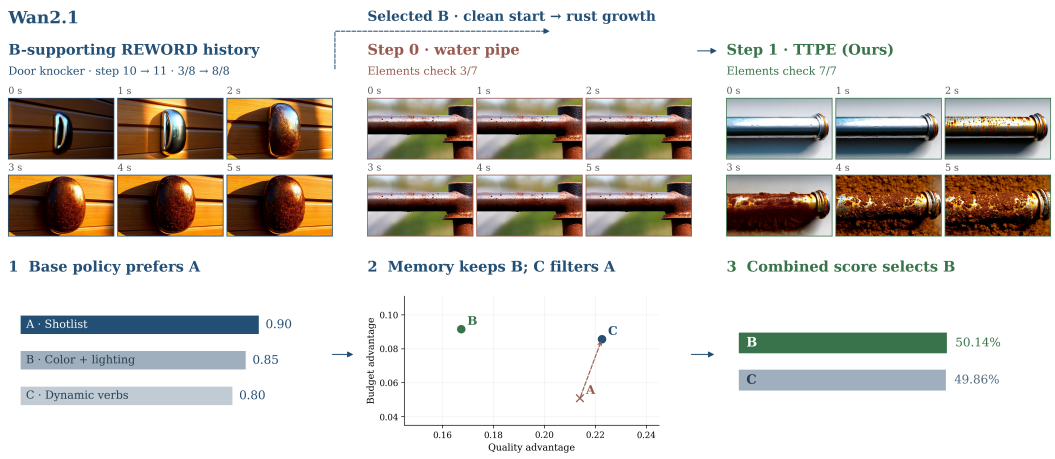

Figure 8: Memory-guided rewrite selection on Wan2.1. Top: a retrieved door-knocker rusting example, the initial water-pipe video, and the result after applying rewrite B, each represented by six frames sampled from 0 to 5 seconds. Bottom: the base policy prefers A; memory-based quality and budget estimates eliminate A through Pareto filtering; the final ranking selects B from the remaining candidates. The resulting video shows a progression from clean metal to heavy rust, improving the Elements check from 3/7 to 7/7.

changes without a clear transition to circular motion. The difference is particularly visible over the later timestamps, where TTPE tends to maintain stronger temporal progression instead of collapsing to a single repeated motion pattern.

This example illustrates the benefit of treating refinement as a sequential test-time decision process rather than applying a fixed number of revisions. By leveraging long-horizon estimates of quality gain and generation cost, TTPE can favor refinement actions that improve the unresolved temporal and motion attributes of the current video, while avoiding additional generations that provide limited improvement.

Decision Mechanism Figure 8 presents a refinement step for a prompt describing a metal water pipe rusting over several years. The initial video shows an already rusty pipe with little visible change, satisfying 3 of 7 checked elements. The retrieved history contains a related door-knocker rusting trajectory, where a `reword` action specifies a clean silver initial surface, changing illumination, and expanding orange-brown rust. That historical transition improves the Elements check from 3/8 to 8/8. The selected rewrite B similarly emphasizes color progression and temporal lighting cues. Its generated video begins with a clean pipe and develops extensive rust, satisfying all seven checked elements. The lower panels show how memory affects the choice among three fixed candidates: A (shotlist), B (color and lighting), and C (dynamic verbs). Their base confidence scores are 0.90, 0.85, and 0.80, respectively, and an empty-memory replay selects A. With memory, C has higher estimated quality and budget advantages than A, causing A to be removed by Pareto filtering. B and C remain eligible: B has the higher budget advantage, while C has the higher quality advantage. Combining the base policy with the estimated quality advantage then yields selection probabilities of 50.14% for B and 49.86% for C.

D Effect of Language-Model Backbones

Table 6 compares TTPE with different language-model backbones on VBench-2.0 using Wan2.1. Overall, the choice of backbone affects refinement quality, especially on dimensions that require more complex motion and physical reasoning. Gemini-3.1 Pro achieves the highest score on most evaluated dimensions, including Camera Motion, Dynamic Attribute, Dynamic Spatial Relationship, Material, Mechanics, Motion Order Understanding, and Thermotics. Claude-Sonnet 4.6 performs best on Human Interaction, while Qwen3.5-9B and Qwen3.8-27B obtain the best results on Composition and Motion Rationality, respectively. Despite these differences, TTPE remains applicable across all four

Table 6: Comparison of Language Model Backbones on VBench 2.0.

Dimension	Qwen3.5-9B	Qwen3.8-27B	Claude-Sonnet 4.6	Gemini-3.1 Pro
Camera Motion	32.10	37.37	43.52	**46.30**
Composition	**68.06**	58.63	62.18	67.17
Dynamic Attribute	64.00	66.67	61.54	**75.82**
Dynamic Spatial Relationship	31.48	40.30	42.19	**43.48**
Human Interaction	61.80	64.95	71.88	68.00
Material	53.33	56.00	80.00	**83.33**
Mechanics	56.25	68.89	70.73	**71.74**
Motion Order Understanding	28.00	30.93	30.61	**38.38**
Motion Rationality	41.67	**55.17**	50.00	41.38
Thermotics	70.0	59.46	65.85	**75.00**

backbones, indicating that the framework is not tied to a specific proprietary model, while stronger reasoning backbones can further improve refinement decisions.

E Memory Step: Cross-Benchmark and Cross-Model Experience Transfer

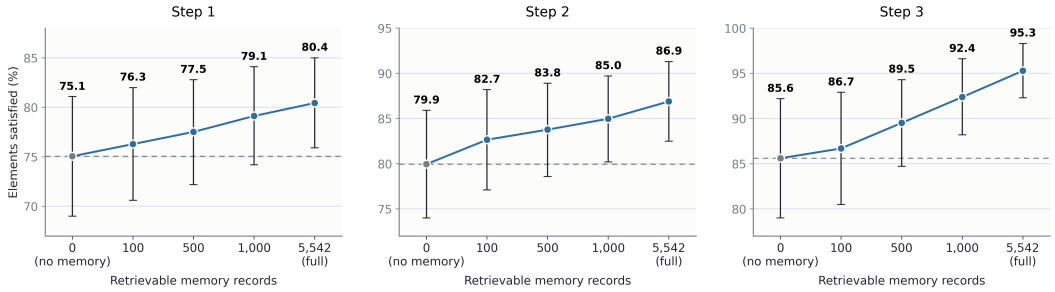

Figure 9: **Cross-benchmark and cross-model transfer of accumulated experience.** We transfer experience records accumulated on VBench 2.0 with Wan to Consist on T2VCompBench, while limiting the execution budget to three steps. The x-axis denotes the number of retrievable experience records, and the y-axis reports the element satisfaction rate.

To further examine whether accumulated experience can generalize beyond the benchmark and model on which it was collected, we conduct a cross-benchmark and cross-model transfer experiment. Specifically, we reuse the experience records accumulated on VBench 2.0 with Wan and transfer them to Consist evaluated on T2VCompBench. We restrict the execution budget to three steps and vary the number of retrievable memory records from no memory to the full memory bank containing 5,542 records. Figure 9 reports the element satisfaction rate at each execution step. Across all three steps, performance consistently improves as more experience records become available for retrieval. With the full memory bank, the element satisfaction rate increases from 75.1% to 80.4% at Step 1, from 79.9% to 86.9% at Step 2, and from 85.6% to 95.3% at Step 3. These correspond to absolute gains of 5.3, 7.0, and 9.7 percentage points over the corresponding no-memory baselines. The improvement becomes larger at later execution steps, suggesting that retrieved experience is not only useful for the initial generation decision, but can also provide increasingly effective guidance throughout iterative refinement. More importantly, the experience records are collected under a different benchmark–model combination from the one used at evaluation time.