VIDEO MODELS REASON EARLY: EXPLOITING PLAN COMMITMENT FOR MAZE SOLVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Video diffusion models have recently been shown to solve reasoning tasks such as mazes and puzzles, yet little is understood about *how* such solutions are created during generation. We study this question using 2D maze-solving as a controlled testbed, examining how solutions form during the denoising process. We find a phenomenon we call **early plan commitment**, where video diffusion models commit to a high-level motion plan within the first few denoising steps, and these early plans are highly predictive of final task success. However, single generations struggle on long mazes, as success drops sharply once the solution requires more steps than one short video can show. Building on these findings, we introduce **Ch**aining with **Ea**rly **P**lanning, or ChEaP, an inference-time strategy which screens seeds by their early plans, completes only the promising ones, and chains short generations to reach solutions beyond a single video's horizon. On Frozen Lake and VR-Bench with Wan2.2-14B and HunyuanVideo-1.5, early screening matches best-of-N with up to $3.3\times$ fewer denoising steps, and chaining raises Wan2.2's success on long mazes from 9.1% (best-of-N) to 72.7%. Finally, we show evidence that early screening transfers beyond grid mazes, improving over best-of-N on irregular mazes and robot manipulation in LIBERO. Understanding *when* video models decide on their answers thus offers a new lens on *how* they solve tasks during generation.

1 INTRODUCTION

Video generation models produce high-fidelity, temporally coherent videos of complex scenes (Wan et al., 2025; Wu et al., 2025). Beyond creative applications, they increasingly serve as models of the world, both as synthetic data engines (Agarwal et al., 2025; Jang et al., 2025) and as world action models that plan a task by generating a video demonstration (Du et al., 2023; Hu et al., 2025; Ye et al., 2026; Kim et al., 2026; Li et al., 2026a). Recent work has also found that these models can solve visual reasoning tasks such as mazes and symmetry puzzles directly in pixel space (Wiedemer et al., 2025; Wang et al., 2026a). Unlike VLMs, which express intermediate reasoning in language, video models express task-relevant decisions directly by evolving visual states. However, success varies widely across samples, especially for harder tasks where correct behaviors are rare tail events. On maze benchmarks, for example, prior work relies on sampling many videos in order to find a successful solution (Yang et al., 2025). Without a deeper understanding of how a video's solution takes shape during generation, it is unclear if this inference-time compute can be better spent.

We use these mazes to study how a solution takes shape. Each maze has a well-defined solution, and correctness can be checked in an automated manner. We investigate the intermediate samples of the denoising process to understand how the solutions take shape. At each denoising step, we decode the model's *intermediate prediction*, its current estimate of the clean video, and extract the agent's path from it. We find that these paths converge to their final plan within the first few denoising steps, while later steps primarily refine visual details (Fig. 1). We call this phenomenon **early plan commitment**, as these early predictions are highly predictive of the final plan, long before generation finishes.

This observation has immediate practical consequences. If the plan is visible early, then standard best-of-N sampling—which fully denoises every seed—wastes most of its compute polishing unsuccessful trajectories. Instead, compute should be spent exploring more candidate plans rather than refining each one. We apply this insight with *Early Planning Beam Search* (EPBS), which partially denoises

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.

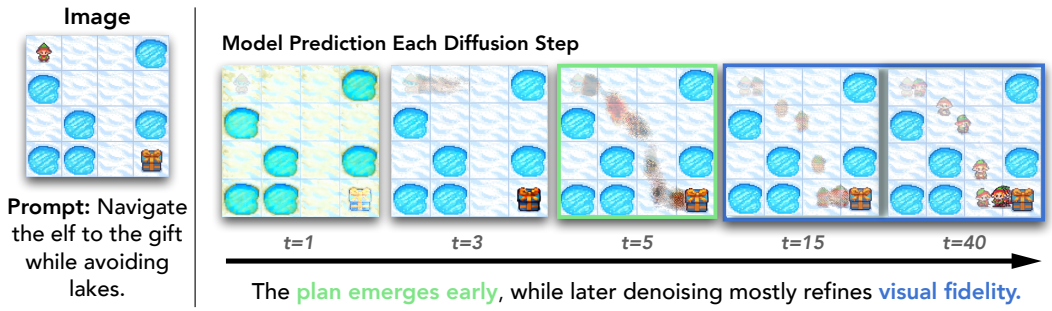


Figure 1: **Video diffusion models plan early.** Intermediate agent trajectories converge within the first few denoising steps (green box); later steps primarily refine visual details (blue box). Each image is a video averaged over all of its frames in the decoded prediction.

a pool of independent seeds, scores their intermediate predictions with a lightweight verifier, and completes only the most promising candidates. By terminating candidates early, EPBS re-allocates that compute toward exploring additional trajectories within the same budget. However, when the candidate pool contains no solution, selection alone is insufficient. We find that single-generation success deteriorates as required maze path length increases, motivating the sequential composition of valid agent-trajectory segments. We achieve this by chaining together short generations using the previous segment's final frame as the initial state for the next segment. Together, these strategies form **Ch**aining with **Ea**rly **P**lanning, or ChEaP (Fig. 5).

We evaluate on Frozen Lake (Towers et al., 2025) and VR-Bench (Yang et al., 2025) using Wan2.2-14B (Wan et al., 2025) and HunyuanVideo-1.5 (Wu et al., 2025). ChEaP matches best-of-N accuracy with $3\times$ fewer wall clock time, and improves accuracy by about 10 points on average on the same NFE budget, outperforming previous techniques in both accuracy and efficiency. On Frozen Lake mazes that exceed the single generation horizon, EPBS succeeds only 25.5% of the time. Chaining lifts this to 72.7%, compared with 9.1% for best-of-N, by letting each generation solve only part of the maze and merging them together after. Finally, we test EPBS on irregular mazes and LIBERO manipulation tasks, where our grid-based analysis no longer applies. EPBS still generalizes and improves over best-of-N in both, and on LIBERO it surpasses best-of-N with $N = 10$ samples using only 40% of the NFEs.

Together, these results show that we can measure when a video model settles on its answer, and that this timing is useful. It explains why late denoising rarely changes a solution, and it tells us where to spend inference-time compute. Without any task-specific training, our strategies turn capabilities expressed inconsistently across samples into more reliable task performance.

2 Related Work

Visual reasoning in video models. Reasoning through video generation provides a setting in which a solution can be expressed in changes to a visual scene. Recent works have shown video model capabilities beyond content generation in tasks such as maze solving, puzzles, and physical reasoning (Wiedemer et al., 2025; Guo et al., 2025). This has motivated new benchmarks and datasets (Yang et al., 2025; Wang et al., 2026a; Chen et al., 2026) and process-aware evaluation beyond final-frame accuracy (Li et al., 2026b). Despite these capabilities, the internal mechanisms of how these models plan and the nature in which they execute sequential reasoning tasks remain underexplored. Concurrent work by Wang et al. (2026b) examines reasoning along the denoising trajectory, proposing a "Chain-of-Steps" view in which early hypotheses progressively converge during generation. We complement this view with a quantitative measurement of when the task-relevant trajectory becomes fixed, which we use to optimize inference-time scaling.

Phase transitions in diffusion models. The reverse diffusion process exhibits a coarse-to-fine hierarchy: global semantics and spatial layout emerge early, while later denoising refines lower-level details (Choi et al., 2022; Sclocchi et al., 2024). This behavior has been studied through cross-attention and internal features (Hertz et al., 2023; Kwon et al., 2023), motivates noise-level-

specialized denoisers (Balaji et al., 2022), and further explored through phase-transition (Raya & Ambrogioni, 2023; Biroli et al., 2024; Li & Chen, 2024), geometric (Yaguchi et al., 2024), spectral (Ventura et al., 2025), and information-theoretic (Ramachandran et al., 2025) analyses. These works primarily characterize visual semantics and layout in image generation. We instead examine when intermediate trajectories converge to their corresponding final trajectories, and whether that timing can guide inference-time compute allocation.

Inference-time scaling for diffusion models. Test-time scaling has explored noise-seed search with verifier feedback for diffusion models (Ma et al., 2025), stochastic generation and adaptive budget allocation for flow models (Kim et al., 2025), Feynman–Kac particle resampling (Singhal et al., 2025), and tree search with MCMC refinement (Zhang et al., 2025). For video, prior work explores noise-trajectory optimization (Liu et al., 2025) and beam search over diffusion latents (Oshima et al., 2025), inspired by analogous scaling results in language models (Snell et al., 2024). Our EPBS instead exploits early convergence of intermediate trajectories, pruning unpromising candidates after only a few denoising steps.

Generative models as planners. Generative models have also been used for sequential decision-making. Diffuser (Janner et al., 2022) denoises state–action trajectories, whereas Decision Diffuser (Ajay et al., 2023) generates state trajectories and recovers actions with an inverse-dynamics model. CompDiffuser (Luo et al., 2025) stitches trajectories through diffusion composition. Related offline reinforcement learning methods, including IQL (Kostrikov et al., 2022) and HIQL (Park et al., 2023), learn policies from previously collected trajectories. At the pixel level, UniPi (Du et al., 2023) and SuSIE (Black et al., 2024) generate future frames or subgoals but delegate actuation to separately trained policies. These methods assume target-domain trajectories, rewards, or low-level control; our reasoning setting instead uses an off-the-shelf video model that must express the plan directly in generated pixels, and extends its horizon by reconditioning on its own rendered outputs.

3 Early Plan Commitment in Video Diffusion Models

To understand how video models solve reasoning tasks, we first ask when during generation they decide on a solution. Since a diffusion model refines its prediction at every denoising step, we can observe how its chosen solution changes over time. We treat the agent's route as the model's *plan*, and we say the model has *committed* to its plan once its intermediate route matches its final route.

Mazes provide a controlled setting to investigate when in the denoising process this commitment happens. They are useful because the environment is wholly specified by the conditioning image, and the solution is expressed as a 2D agent trajectory from the start state to the goal. In this setting, we compare trajectories extracted from clean-sample predictions at intermediate denoising steps to the final trajectory produced by the same seed. We run Wan2.2-14B on Frozen Lake (Towers et al., 2025) across grid sizes 4×4 to 10×10, totaling 747 trajectories.

3.1 Observing intermediate trajectories

All video models we study are built on *flow matching* (Lipman et al., 2023; Liu et al., 2023b), a generative framework that learns a velocity field transporting noise ϵ to data x_0 along straight paths. During training, an interpolant constructs noisy samples $x_t = (1-t)\,x_0 + t\,\epsilon$ for $t \in [0,1]$ and $\epsilon \sim \mathcal{N}(0, I)$, and a network v_θ is trained to predict the velocity $\frac{dx_t}{dt} = \epsilon - x_0$. At inference, one integrates v_θ from $t = 1$ (pure noise) to $t = 0$ (clean video) using a discrete schedule of T steps.

In particular, at any intermediate step t, the velocity prediction can be rearranged to recover a *clean-sample estimate*:

$$\hat{x}_0^{(t)} = x_t - t \cdot v_\theta(x_t, t). \tag{1}$$

While the current latent x_t may still be highly noisy, the intermediate prediction ($\hat{x}_0^{(t)}$) can be decoded into pixel space without completing the remaining denoising steps. These decoded predictions let us directly examine how the generated video evolves throughout inference.

Throughout, we index denoising steps by $\tau \in \{1, \ldots, T\}$ and write $\hat{x}_0^{(\tau)}$ for the clean-sample estimate at step τ. From each $\hat{x}_0^{(\tau)}$, we extract an *intermediate trajectory*: the ordered sequence of grid cells

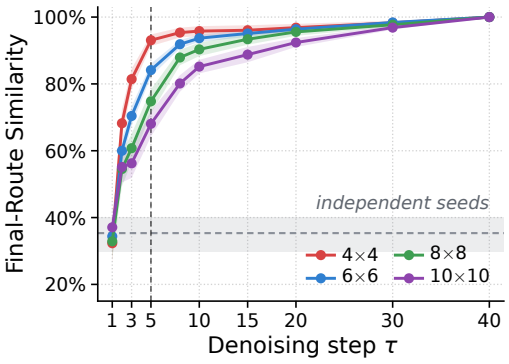


Figure 2: **Trajectories converge to final routes early in denoising.** We calculate order-aware similarity between trajectories at denoising step τ and final trajectory. The gray band denotes chance similarity between independent seeds.

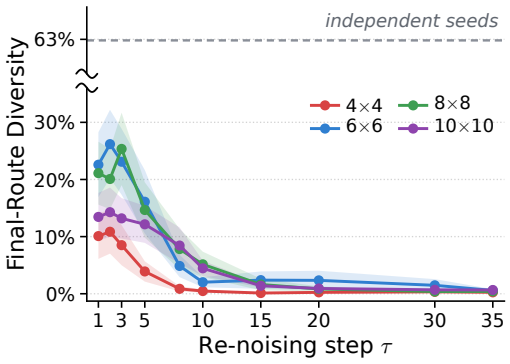


Figure 3: **Re-noising explores less than resampling new seeds.** Shown is route diversity between five re-noised branches and its parent's final video, across branch steps τ. The dashed line denotes diversity across independent seeds.

visited by the agent. We refer to the trajectory extracted from the fully denoised video as the *final trajectory*. With this setup we ask: *how early does an intermediate trajectory converge to the final trajectory generated by the same seed?*

3.2 Final motion paths emerge early

We quantify this convergence by comparing each intermediate trajectory to the final trajectory from the same seed. To do so, we estimate the agent's position in every frame using background subtraction and then map each position to a grid cell, which forms the route taken. Afterwards, we compute *final-route similarity* using dynamic time warping (DTW) (Müller, 2007), reporting $1 - \mathrm{DTW}_{\mathrm{norm}}$ so that larger values indicate greater similarity. Because DTW matches cells in order, it separates routes that cover the same cells in a different sequence.

Across 747 trajectories and all maze sizes, intermediate trajectories reach an average final-route similarity of 80.3% by denoising step 5 and 91.4% by step 10 (Fig. 2), compared with 35.3% between the final routes of independent seeds on the same maze. Measured from this chance level, trajectories close roughly 70% of the gap to their final route by step 5 and 87% by step 10, indicating that routes are largely decided within the first quarter of denoising and remain largely unchanged thereafter. Larger mazes, with more possible routes, take longer to commit. At step 5, $10{\times}10$ mazes reach 68.1% similarity compared with 93.1% for $4{\times}4$ mazes. We observe similar convergence dynamics in HunyuanVideo-1.5 (see Appendix D.1). This is closely related to the idea of *phase transitions* in diffusion models (Choi et al., 2022), where diffusion models generate image details in a coarse-to-fine hierarchy. In our case, the high level plan constitutes a "coarse" detail of the video. Here, the early stabilization of this plan offers an opportunity to assess task progress before generation is complete, enabling more efficient search for successful solutions (Sec. 4). Qualitative examples across seeds are provided in Appendix E.1.

3.3 New seeds explore more routes than refinement

If the model fixes its route within the first few steps, the remaining steps leave little room to consider alternative solutions. We therefore ask whether route diversity is better obtained by branching from an existing generation or by sampling a new seed. Prior work branches from intermediate states through search over denoising paths and local refinement to obtain higher quality samples (Ma et al., 2025; Zhang et al., 2025). We test the simplest version of this on a different set of mazes: re-noising an intermediate prediction and continuing from it.

For a fixed seed, we run the sampler normally until denoising step τ and cache both its latent x_τ and clean prediction $\hat{x}_0^{(\tau)}$. The noise level assigned to step τ is denoted by σ_τ, with larger σ_τ corresponding to noisier, earlier states. We then re-noise the clean prediction back to the *same* noise

level with independent Gaussian noise, creating a new step τ prediction

$$\tilde{x}_\tau = (1 - \sigma_\tau)\hat{x}_0^{(\tau)} + \sigma_\tau \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I). \tag{2}$$

We repeat this for five branches, then follow the same remaining denoising schedule as its parent. An unperturbed parent completion resumes from x_τ, providing the trajectory that the original seed would have generated without intervention. We measure *route diversity* as $\mathrm{DTW}_{\mathrm{norm}}$ between the routes extracted from each branch's fully denoised video and its parent's, computed exactly as in Section 3.2. For comparison, we compute the same quantity between final videos generated from independent seeds on the same maze, $4{,}635$ branches in total.

Route diversity peaks at the earliest branch points ($\tau \leq 3$) and decays to near zero by $\tau = 15$ as seen in Figure 3. A re-noised branch differs from its parent by only 26.2% at most (over maze sizes and branch step) against 62.9% average diversity between independent seeds. The decay in route diversity mirrors the convergence in Section 3.2 on a similar timescale. Crucially, beyond $\tau = 15$, branches stay within 2.4% of their parent's route, nearly indistinguishable from the original trajectory. These results demonstrate that re-noising individual seeds yields very little route exploration compared to sampling independent seeds, especially after the early commitment threshold.

4 Trajectory Screening for Efficient Sampling

In Section 3, we found that a seed's route is largely decided within the first few denoising steps, and that new routes come from new seeds rather than from refining existing ones. This suggests a different way to spend inference-time compute. We can probe many seeds briefly and fully denoise only those whose early routes are promising. We develop this idea into a sampling strategy for generating successful demonstrations of a query task (Ma et al., 2025; Wiedemer et al., 2025).

4.1 Early Planning Beam Search

Early Planning Beam Search, or EPBS, reallocates inference-time compute from full denoising to early candidate exploration. Rather than fully denoising every seed, we first partially denoise many candidates for τ steps, decode their intermediate predictions, score them with a lightweight verifier, and reserve complete denoising only for the top-K seeds. Under a fixed number of function evaluations (NFEs), EPBS explores substantially more candidates than standard best-of-N sampling.

Given a compute budget B, probe step τ out of T total denoising steps, and K final candidates, EPBS probes $N = \lfloor (B - K(T - \tau))/\tau \rfloor$ independent seeds, ranks their intermediate predictions, and only continues denoising the top-K predictions for the remaining $T - \tau$ denoising steps. By reliably skipping unpromising seeds based on early plan commitment, we save on compute which did not meaningfully help us find better solutions.

Lightweight trajectory verifier. Our goal is to score intermediate predictions with as little task knowledge as possible. For our tasks, the verifier only takes the agent's starting position and the goal's position into consideration. We take an intermediate prediction $\hat{x}_0^{(\tau)}$ and decode its final coordinate p_{end} using a motion tracker to assess its progress from the start p_{start} towards the goal pixel p_{goal}:

$$\text{score}\left(\hat{x}_0^{(\tau)}\right) = 1 - \frac{d(p_{\text{end}}, p_{\text{goal}})}{d(p_{\text{start}}, p_{\text{goal}})} \quad \text{where} \quad d(x, y) := \|x - y\|_2^2, \quad x, y \in \mathbb{R}^2 \tag{3}$$

Only the top-K seeds ranked by score are fully denoised and evaluated for task success. Motion tracker and scoring details are in Appendices A.1 and A.3.

Evaluation. We evaluate on 302 Frozen Lake mazes across four grid sizes and all 180 VR-Bench mazes, 482 mazes in total, using Wan2.2-14B and HunyuanVideo-1.5 (Wan et al., 2025; Wu et al., 2025). We evaluate *task success* rather than exact agreement with a reference solution path, as done in prior work (Yang et al., 2025). Exact agreement is overly strict, as many mazes admit multiple valid solutions, and small trajectory-extraction errors can invalidate an otherwise correct video. Instead, we extract the agent trajectory from the generated video using SAM 2 (Ravi et al., 2024) and determine success if the extracted trajectory reaches the goal without violating task constraints. Full extraction details and success criteria are provided in Appendices A.1 and A.4.

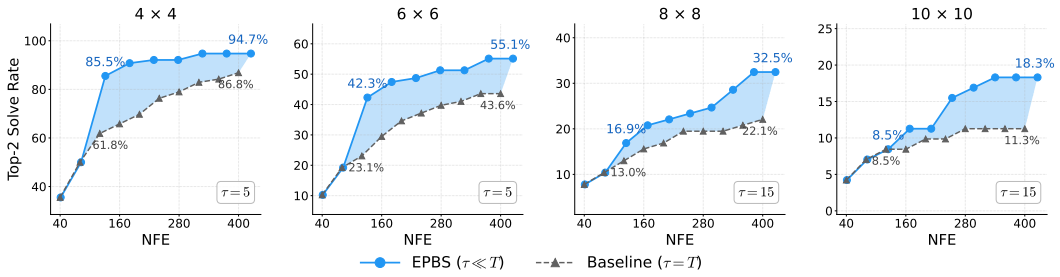


Figure 4: **EPBS finds solutions much more efficiently than best-of-N.** Accuracy vs Function Evaluations (NFEs) on Frozen Lake mazes across four sizes with Wan2.2-14B. NFEs here are adjusted for the extra VAE calls and other extra time incurred during EPBS. EPBS improves the compute–accuracy trade-off, with the largest wall-clock speedups on smaller mazes.

4.2 Results

We compare EPBS to best-of-N (equivalent to $\tau = T$) with $N = 1, \ldots, 10$, using beam size $K = 2$ and top-2 solve rate, on Wan2.2-14B ($T = 40$) and HunyuanVideo-1.5 Step-Distilled ($T = 8$). For Wan2.2, this implies $N = 66$ samples when probing at $\tau = 5$ and $N = 23$ samples at $\tau = 15$. For HunyuanVideo-1.5, we probe at $\tau = 2$ of $T = 8$ steps, and at $\tau = 3$ on VR-Bench Hard.

As shown in Fig. 4, EPBS outperforms best-of-N by around 10% on average for Wan2.2-14B across all maze sizes. It matches best-of-N accuracy with up to $3.3\times$ fewer NFEs, with the largest gains on mazes where standard sampling plateaus. On HunyuanVideo-1.5, EPBS improves top-2 solve rate on 4×4 mazes by 7.9 and 14.4 points at 40 and 80 NFEs, with smaller or no gains on larger mazes and VR-Bench (Fig. 6a). Counting NFEs alone understates the cost of EPBS, since each probe also decodes $\hat{x}_0^{(\tau)}$ with the VAE. Using wall-clock timings of 11.2s per NFE and 18s per VAE decode, we estimate that one decode costs roughly 1.6 NFEs, and we charge EPBS for these decodes in Fig. 4. EPBS still improves over best-of-N under this accounting, and it reaches the same accuracy up to $3.0\times$ faster in end-to-end wall-clock time (Appendix C.3).

Hyperparameter selection. We explore different values of τ and K on Frozen Lake mazes in Appendices C.1 and C.2, but summarize our results here. Probing at τ=5 yields the best trade-off for smaller sizes, while $\tau = 10, 15$ is better for larger mazes, indicating that trajectories converge later. Beam size K=2 is the most robust choice, as K=1 plateaus when its single top-ranked seed fails.

Verifier reliability of early predictions. Early predictions are not only stable but also informative about correctness. On Frozen Lake, our verifier's scores at the probe step predict final task success with ROC AUC above 0.85 at every maze size, and above 0.93 on large mazes, where successful seeds are rare (Tab. 1). When we pick two seeds from the pool, our verifier comes close to the oracle on large mazes and falls roughly halfway between random and oracle selection on small ones. Our verifier sees only the start and goal positions, with no access to obstacles. A privileged verifier that also penalizes crossing a lake raises top-2 precision on small mazes ($+21.3$ points on 4×4), where many early predictions shortcut through lakes, but matches ours on 10×10, where selection is hardest and few goal-reaching predictions are valid to begin with.

Table 1: **End-goal distance is enough for our verifier.** Precision of selecting two seeds per maze based on their $\hat{x}_0$ predictions. *Random* selects two seeds uniformly; *Verifier* is our end-goal verifier used in EPBS; *Privileged* additionally penalizes obstacle interactions; and *Oracle* selects the two best possible seeds. AUC scores early predictions against final task success over all seeds.

Size	4×4	6×6	8×8	10×10
Random	40.7	19.7	11.1	6.2
Verifier top-2	72.7	33.8	19.5	10.1
Privileged top-2	94.0	37.7	20.8	10.1
Oracle	98.7	53.9	22.8	12.9
AUC	0.859	0.887	0.956	0.939

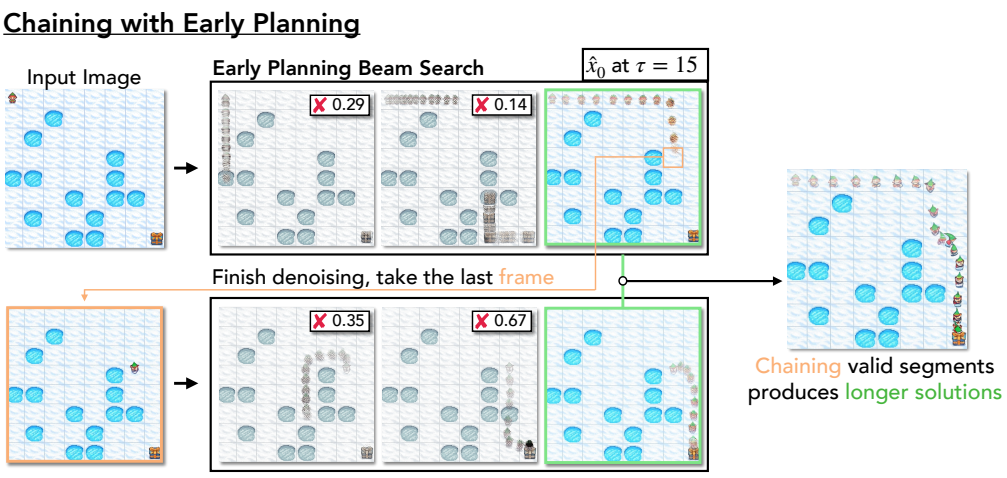


Figure 5: **Overview of Chaining with Early Planning (ChEaP).** *Early Planning Beam Search* scores intermediate predictions and selects the most promising candidates for full generation. *Chaining* reconditions on the last frame of a valid agent-trajectory segment to condition the next generation.

5 Chaining Generations for Long-Horizon Reasoning

EPBS can only surface a solution that one of its seeds already contains. When we break down success by the length of the optimal path (Fig. 6b), we find that this becomes a hard limit on long mazes. On Frozen Lake, EPBS solves about 90% of mazes that require at most 9 moves, but only 25.5% of those requiring 10 to 13. Path length dominates this failure, with per-maze success strongly anti-correlated with it (r from -0.58 to -0.78, on mazes 6×6 and above), while obstacle density shows almost no correlation ($|r| \leq 0.06$; Tab. 4). We hypothesize that the limit is the model's *generation horizon*, since a single 5-second video is too short for the agent to traverse a long path. If so, the model should be able to solve these mazes when we split them across several shorter generations.

5.1 Chaining

We test this with *chaining*, which decomposes a long-horizon task into a sequence of shorter sub-problems, each solvable within a single generation. After each generation, we pick a *pivot* frame as the conditioning image for the next, extending the model's planning horizon across multiple generations. Together with EPBS, this forms ChEaP: **Ch**aining with **Ea**rly **P**lanning.

Given an input task, we again use EPBS to screen N candidate seeds by their intermediate predictions. Unlike EPBS alone, chaining must hand a valid partial solution to the next segment, so here we additionally use the maze layout. We discard seeds whose predicted trajectories violate the maze constraints, and fully generate the top-K remaining seeds according to our verifier. The last valid frame of the best completed candidate becomes the *pivot frame*, which then conditions the next generation. Chaining stops once a candidate solves the maze according to our success checker (Appendix A.4), when no candidate provides a usable pivot, or after $D_{\max}$ pivots. Crucially, no latent or hidden model state is carried between segments, and each segment sees only the pivot frame.

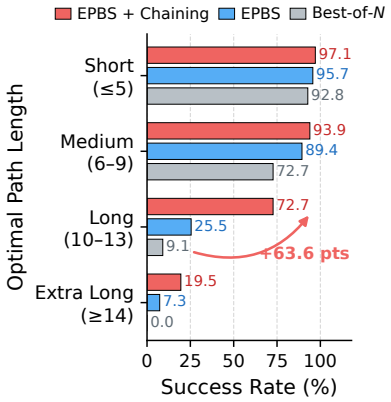
Compute budget. ChEaP screens candidates at the probe step τ and completes only candidates the verifier ranks for verification, so a chain of D segments costs $\sum_{d=0}^{D-1} (N_d\,\tau + K_d\,(T - \tau))$ NFEs, where N_d and K_d are the numbers of seeds screened and completed at depth d. Since N_d and K_d depend on how quickly a valid pivot or solution is found, we report the mean NFE across chaining runs. On Frozen Lake we use $\tau = 15$, screen $N = 20$ from each pivot, and allow $D_{\max} = 3$ pivots.

5.2 Results

Figure 6a details the performance of ChEaP on Frozen Lake and VR-Bench mazes. In particular for Wan2.2 on 6×6 mazes, ChEaP reaches 88.5%, a 32.1-point gain over EPBS alone and more than

Method	NFEs	Frozen Lake 4×4	Frozen Lake 6×6	Frozen Lake 8×8	Frozen Lake 10×10	VR-Bench Easy	VR-Bench Medium	VR-Bench Hard
Wan2.2-14B (40 NFEs / full gen)								
BoN	120	61.8	23.1	13.0	8.5	40.0	8.0	3.3
EPBS	120	85.5	42.3	16.9	8.5	54.7	18.7	6.7
BoN	400	86.8	43.6	22.1	11.3	58.7	20.0	6.7
EPBS	400	96.1	56.4	32.5	18.3	72.0	36.0	23.3
ChEaP	∼630	**98.7**	**88.5**	**46.8**	**22.5**	**74.7**	**50.7**	**26.7**
HunyuanVideo-1.5 Step Distilled (8 NFEs / full gen)								
BoN	40	27.6	15.4	11.7	1.4	25.3	1.3	0.0
EPBS	40	35.5	19.2	11.7	1.4	26.7	2.7	0.0
BoN	80	38.2	20.5	11.7	2.8	26.7	2.7	**3.3**
EPBS	80	52.6	23.1	11.7	2.8	32.0	2.7	**3.3**
ChEaP	∼95	**60.5**	**29.4**	**15.5**	**5.6**	**49.3**	**4.0**	**3.3**

(a) Top-2 solve rate (%) on Frozen Lake and VR-Bench.



(b) Frozen Lake success by path length.

Figure 6: **ChEaP substantially improves maze performance, especially beyond the single-generation horizon.** (a) For Wan2.2 on 4×4 and 6×6 Frozen Lake mazes, EPBS at 120 NFEs achieves similar accuracy to best-of-N at 400 NFEs. At a matched budget of 400 NFEs, EPBS outperforms best-of-N across all reported Wan2.2 splits. (b) When we break down success rate by optimal path length, we clearly see that chaining helps most when the solution exceeds the generation window (≥ 10 steps), raising long maze success rate from 9% to 73%.

double the best-of-N rate of 43.6%. The gains concentrate precisely where EPBS is bottlenecked by the horizon (Fig. 6b); on mazes requiring 10–13 moves, ChEaP reaches 72.7%, up from 25.5% with EPBS and 9.1% with best-of-N. VR-Bench shows a similar trend. For Wan2.2, the largest gains come on Medium puzzles, while the weaker HunyuanVideo benefits most on Easy puzzles. In short, chaining successfully extends the horizon of both models.

On paths of 14+ steps, the gain from ChEaP shrinks (7.3% to 19.5%), which we primarily attribute to compounding errors across chain segments. Particularly, pivot frame quality degrades with chain depth, making it difficult to continuously extend generations. One notable failure mode (illustrated in Figure 5, middle of top row) is when the model attempts to satisfy the prompt by any means possible. This usually manifests either as teleporting the elf to the gift in the last second in early generations or as moving the gift towards the elf instead of keeping it static. We include more analysis on chaining and path length related failures in Appendix B.

6 Beyond Gridworlds: Irregular Mazes and Robot Manipulation

The mazes we have studied so far are gridworlds, whose cells let us extract consistent routes and measure early plan commitment (Sec. 3). EPBS itself does not need a grid, since our verifier only scores how much progress a prediction makes toward the goal. Therefore, we test EPBS on two domains without a grid: irregular mazes and tabletop robot manipulation. Qualitatively, the model settles on its plan within the first few denoising steps in both domains as well (Fig. 7).

Irregular mazes. VR-Bench has a separate split for irregular mazes, which replace the grid with walls drawn at arbitrary angles, so there are no cells or discrete moves to check a trajectory against. We condition on the rendered maze and task instruction and evaluate 60 puzzles across four visual styles. We count successes when the agent reaches the goal without crossing any walls, the continuous analogue of our grid success criterion (Appendix A.4). We use the verifier from Sec. 4 unchanged.

Simulated robot manipulation. LIBERO (Liu et al., 2023a) specifies tabletop tasks through natural-language instructions and goal predicates over objects. We condition on a rendered initial observation and the task instruction, and evaluate on the 116 tasks with `On`/`In` goal predicates. A video succeeds if every required object is placed so that it rests on or in its target, judged on the final video by Gemini 3 Flash (full details in Appendix A.4). Here, success is a relation between two objects rather than an agent reaching a location. Since the robot arm repeatedly occludes the object it carries, we

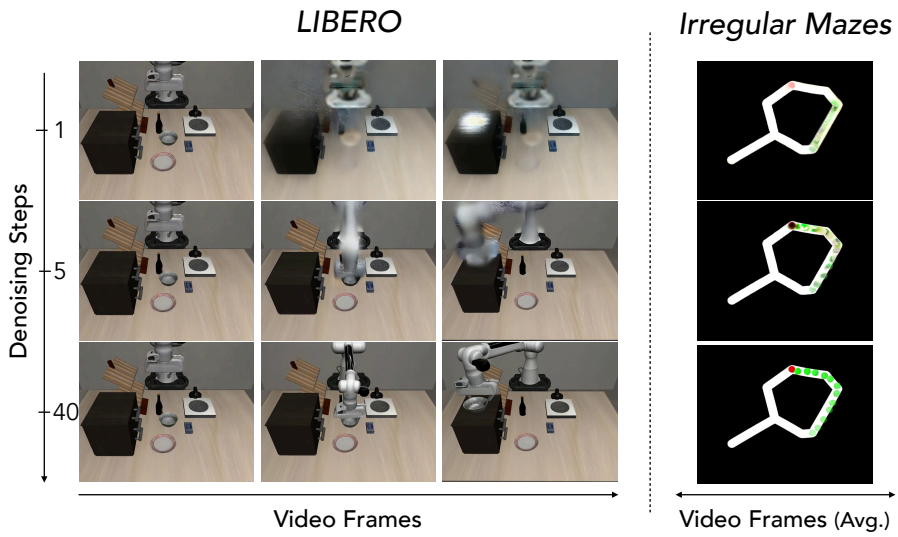


Figure 7: **Early plan commitment beyond gridworlds.** (Left) We visualize the intermediate predictions of Wan2.2 at $\tau = 1, 5, 40$ over three frames for robot manipulation. Qualitatively, early plan commitment can be seen to come about at step 5. (Right) The same is true for irregular mazes, where the model sketches its plan over the first $\tau = 5$ denoising steps, after which its plan does not deviate from the final plan at $\tau = 40$.

track both objects with SAM 2, prompted with boxes from LIBERO's instance segmentation of the initial scene. We score an intermediate prediction by whether the two masks come to intersect and stay intersected over the final frames of the video. As with mazes, the verifier needs only to know which object should move and where it should end up; here, this information comes from LIBERO's instance segmentation of the initial scene.

Results. Table 2 compares EPBS to best-of-N at matched NFE budgets. On LIBERO, EPBS improves success from 77.6% to 87.1% at 160 NFEs, already exceeding best-of-N at 400 NFEs (85.3%), similar to what we observe on Frozen Lake (Fig. 6a). At 400 NFEs the gap narrows to 1.8 points, as best-of-N also finds a successful candidate for most tasks. On irregular mazes, EPBS reaches 40.0% on easy and 23.3% on medium puzzles at 400 NFEs, compared with 26.7% and 16.7% for best-of-N, though it trails on medium puzzles at 160 NFEs (10.0% vs. 13.3%). These results suggest that trajectory screening is not necessarily tied to grid structure. A verifier that knows only where the moving object starts and where it should end up improves selection on both irregular mazes and LIBERO.

Table 2: **EPBS transfers beyond gridworlds.** Top-2 solve rate (%) with Wan2.2-14B at matched NFE budgets, τ=5, K=2. Irregular mazes: 60 puzzles across four visual styles, judged by SAM 2 tracking. LIBERO: 116 tasks with `On`/`In` goal predicates, judged by Gemini 3 Flash.

		Best-of-N		EPBS	
Domain	Split	160	400	160	400
Irregular Mazes	Easy	26.7	26.7	30.0	**40.0**
	Medium	13.3	16.7	10.0	**23.3**
LIBERO	–	77.6	85.3	**87.1**	**87.1**

7 Conclusion

Our findings identify two properties of how current video models solve tasks during generation. The task-relevant trajectory becomes fixed early in denoising, while single-generation performance deteriorates sharply as the required trajectory length grows. The first property allows us to screen candidates long before they are fully generated, and the second motivates composing partial solutions across generations. Our analysis is agnostic to how a model arrives at its plan, whether by generalizing or by drawing on patterns seen in pretraining. We characterize when that plan becomes fixed, and we observe early commitment across both models and all domains we study. More broadly, standard full-generation sampling leaves useful task-relevant signal unexploited, and exploiting the temporal structure of denoising can substantially improve performance without retraining.

AI USE STATEMENT

We used generative AI tools to help write code, to give feedback on research methodology, and critique the writing of this paper. We have not used generative AI tools for proposing foundational research ideas used in the paper, assisting in the writing of proofs, or the use of AI for generating synthetic data. Assisting with translation is not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957, November 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-54281-3. URL https://www.nature.com/articles/s41467-024-54281-3.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Y. charles, Yiping Bao, Yuantao Fan, Guopeng Li, Haiyang Shen, Xuanzhong Chen, Wendong Xu, Shuzheng Si, Zefan Cai, Wenhao Chai, Ziqi Huang, Fangfu Liu, Tianyu Liu, Baobao Chang, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. BabyVision: Visual Reasoning Beyond Language, January 2026. URL http://arxiv.org/abs/2601.06521. arXiv:2601.06521 [cs].

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, Dongzhi Jiang, Xiangtai Li, Manyuan Zhang, Hongsheng Li, and Pheng-Ann Heng. Are video models ready as zero-shot reasoners? an empirical study with the mme-cof benchmark. *arXiv preprint arXiv:2510.26802*, 2025.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. In *International Conference on Learning Representations (ICLR)*, 2023.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *International Conference on Machine Learning (ICML)*, 2025.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, pp. 9902–9915, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Jaihoon Kim, Taehoon Yoon, Jisung Hwang, and Minhyuk Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2503.19385.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion models already have a semantic latent space. In *International Conference on Learning Representations (ICLR)*, 2023.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a.

Marvin Li and Sitan Chen. Critical windows: Non-asymptotic theory for feature emergence in diffusion models. In *International Conference on Machine Learning (ICML)*, 2024.

Yifan Li, Yukai Gu, Yingqian Min, Zikang Liu, Yifan Du, Kun Zhou, Min Yang, Wayne Xin Zhao, and Minghui Qiu. Beyond the Last Frame: Process-aware Evaluation for Generative Video Reasoning, January 2026b. URL http://arxiv.org/abs/2512.24952. arXiv:2512.24952 [cs] version: 2.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023a.

Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-t1: Test-time scaling for video generation. *arXiv preprint arXiv:2503.18942*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023b.

Yunhao Luo, Utkarsh A. Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *arXiv preprint arXiv:2503.05153*, 2025.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps, January 2025. URL http://arxiv.org/abs/2501.09732. arXiv:2501.09732 [cs].

Meinard Müller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pp. 69–84. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-74048-3_4.

Yuta Oshima, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Inference-time text-to-video alignment with diffusion latent beam search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sai Niranjan Ramachandran, Manish Krishan Lal, and Suvrit Sra. Cross-fluctuation phase transitions reveal sampling dynamics in diffusion models, October 2025. URL http://arxiv.org/abs/2511.00124. arXiv:2511.00124 [cs].

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos, October 2024. URL http://arxiv.org/abs/2408.00714. arXiv:2408.00714 [cs].

Gabriel Raya and Luca Ambrogioni. Spontaneous Symmetry Breaking in Generative Diffusion Models, October 2023. URL http://arxiv.org/abs/2305.19693. arXiv:2305.19693 [cs].

Antonio Sclocchi, Alessandro Favero, and Matthieu Wyart. A Phase Transition in Diffusion Models Reveals the Hierarchical Nature of Data, December 2024. URL http://arxiv.org/abs/2402.16991. arXiv:2402.16991 [stat].

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *International Conference on Machine Learning (ICML)*, pp. 55810–55827, 2025. URL https://proceedings.mlr.press/v267/singhal25b.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments, November 2025. URL http://arxiv.org/abs/2407.17032. arXiv:2407.17032 [cs].

Enrico Ventura, Beatrice Achilli, Gianluigi Silvestri, Carlo Lucibello, and Luca Ambrogioni. Manifolds, Random Matrices and Spectral Gaps: The geometric phases of generative diffusion, April 2025. URL http://arxiv.org/abs/2410.05898. arXiv:2410.05898 [stat].

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and Advanced Large-Scale Video Generative Models, April 2025. URL http://arxiv.org/abs/2503.20314. arXiv:2503.20314 [cs].

Maijunxian Wang, Ruisi Wang, Juyi Lin, Ran Ji, Thaddäus Wiedemer, Qingying Gao, Dezhi Luo, Yaoyao Qian, Lianyu Huang, Zelong Hong, Jiahui Ge, Qianli Ma, Hang He, Yifan Zhou, Lingzi Guo, Lantao Mei, Jiachen Li, Hanwen Xing, Tianqi Zhao, Fengyuan Yu, Weihang Xiao, Yizheng Jiao, Jianheng Hou, Danyang Zhang, Pengcheng Xu, Boyang Zhong, Zehong Zhao, Gaoyun Fang, John Kitaoka, Yile Xu, Hua Xu, Kenton Blacutt, Tin Nguyen, Siyuan Song, Haoran Sun, Shaoyue Wen, Linyang He, Runming Wang, Yanzhi Wang, Mengyue Yang, Ziqiao Ma, Raphaël Millière, Freda Shi, Nuno Vasconcelos, Daniel Khashabi, Alan Yuille, Yilun Du, Ziming Liu, Bo Li, Dahua Lin, Ziwei Liu, Vikash Kumar, Yijiang Li, Lei Yang, Zhongang Cai, and Hokin Deng. A Very Big Video Reasoning Suite, February 2026a. URL http://arxiv.org/abs/2602.20159. arXiv:2602.20159 [cs].

Ruisi Wang, Zhongang Cai, Fanyi Pu, Junxiang Xu, Wanqi Yin, Maijunxian Wang, Ran Ji, Chenyang Gu, Bo Li, Ziqi Huang, Hokin Deng, Dahua Lin, Ziwei Liu, and Lei Yang. Demystifying video reasoning. *arXiv preprint arXiv:2603.16870*, 2026b.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, Linus, Patrol, Peizhen Zhang, Peng Chen, Penghao Zhao, Qi Tian, Songtao Liu, Weijie Kong, Weiyan Wang, Xiao He, Xin Li, Xinchi Deng, Xuefei Zhe, Yang Li, Yanxin Long, Yuanbo Peng, Yue Wu, Yuhong Liu, Zhenyu Wang, Zuozhuo Dai, Bo Peng, Coopers Li, Gu Gong, Guojian Xiao, Jiahe Tian, Jiaxin Lin, Jie Liu, Jihong Zhang, Jiesong Lian, Kaihang Pan, Lei Wang, Lin Niu, Mingtao Chen, Mingyang Chen, Mingzhe Zheng, Miles Yang, Qiangqiang Hu, Qi Yang, Qiuyong Xiao, Runzhou Wu, Ryan Xu, Rui Yuan, Shanshan Sang, Shisheng Huang, Siruis Gong, Shuo Huang, Weiting Guo, Xiang Yuan, Xiaojia Chen, Xiawei Hu, Wenzhi Sun, Xiele Wu, Xianshun Ren, Xiaoyan Yuan, Xiaoyue Mi, Yepeng Zhang, Yifu Sun, Yiting Lu, Yitong Li, You Huang, Yu Tang, Yixuan Li, Yuhang Deng, Yuan Zhou, Zhichao Hu, Zhiguang Liu, Zhihe Yang, Zilin Yang, Zhenzhi Lu, Zixiang Zhou, and Zhao Zhong. HunyuanVideo 1.5 Technical Report, November 2025. URL http://arxiv.org/abs/2511.18870. arXiv:2511.18870 [cs].

Manato Yaguchi, Kotaro Sakamoto, Ryosuke Sakamoto, Masato Tanabe, Masatomo Akagawa, Yusuke Hayashi, Masahiro Suzuki, and Yutaka Matsuo. The Geometry of Phase Transitions in Diffusion Models: Tubular Neighbourhoods and Singularities. October 2024. URL https://openreview.net/forum?id=YYMd6zsP2e.

Cheng Yang, Haiyuan Wan, Yiran Peng, Xin Cheng, Zhaoyang Yu, Jiayi Zhang, Junchi Yu, Xinlei Yu, Xiawu Zheng, Dongzhan Zhou, and Chenglin Wu. Reasoning via Video: The First Evaluation of Video Models' Reasoning Abilities through Maze-Solving Tasks, November 2025. arXiv:2511.15065 [cs].

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time Scaling of Diffusion Models through Classical Search, October 2025. URL http://arxiv.org/abs/2505.23614. arXiv:2505.23614 [cs].

A Implementation Details

A.1 Trajectory Extraction Pipeline

We extract trajectories from generated videos in our analysis as well as the methods built from it. We have two methods of extraction:

Trajectory Extraction from Intermediate Predictions. On intermediate predictions, the agent trajectory is typically blurry and flickers, causing naive motion detection methods to fail. From the video's frames, we estimate a static background as the per-pixel median of every fifth frame. A pixel is marked as moving when its largest per-channel absolute difference from the background exceeds 60 (on a 0–255 scale). The agent's position in a frame is the centroid of the moving pixels (c_x, c_y). If fewer than 50 pixels move, the detection is treated as missing and the previous position is carried forward. In our analysis only, each position is mapped to a grid cell, resulting in one cell per frame.

Tracking with SAM 2 for Evaluation. For evaluation, we extract trajectories from fully decoded video, which do not incur the same visual clarity issues as extracting from decoded intermediate predictions. In this case, we use SAM 2.1 (Hiera-Tiny) to track the elf sprite across video frames (Ravi et al., 2024). The tracker is initialized in the first frame with a bounding box derived from the known start-cell pixel region (from maze metadata). For each subsequent frame, SAM 2 produces a segmentation mask from which we extract the centroid (c_x, c_y). We similarly track the goal (*e.g.*, the gift in Frozen Lake) to check for goal drift during generation.

Centroid-to-cell mapping. In our analysis and during evaluation, we need to map pixel coordinates to grid cells. Given the dimensions of the image $(x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}})$ and grid size G, we compute cell dimensions $w_{\text{cell}} = (x_{\text{max}} - x_{\text{min}})/G$ and $h_{\text{cell}} = (y_{\text{max}} - y_{\text{min}})/G$. Each centroid is mapped to a grid cell:

$$\text{col} = \left\lfloor \frac{c_x - x_{\text{min}}}{w_{\text{cell}}} \right\rfloor, \quad \text{row} = \left\lfloor \frac{c_y - y_{\text{min}}}{h_{\text{cell}}} \right\rfloor \tag{4}$$

with clamping to $[0, G-1]$. The trajectory is the ordered sequence of unique cells visited.

A.2 Trajectory Similarity Metrics

The similarity measures of Sec. 3 compare the route in each intermediate prediction $\hat{x}_0^{(\tau)}$ with the same seed's final video. They use a lightweight tracker based on background-difference, rather than the SAM 2 tracker of Appendix A.1. This is the same tracker we use in our trajectory verifier in Sec. 4.

Route Similarity. For DTW, we align the per-frame sequences of each video and use the Manhattan distance between cells as the cost. Let D be the total cost of the optimal alignment and L the length of its warping path. We define $\text{DTW}_{\text{norm}} = D/\big(L \cdot \bar{d}_N\big)$, where $\bar{d}_N = 2(N^2-1)/(3N)$ is the expected Manhattan distance between two cells drawn uniformly at random on an $N \times N$ grid, so that 0 denotes identical routes and 1 the separation of two random positions, and report $1 - \text{DTW}_{\text{norm}}$. We choose $\bar{d}_N$ because distances grow with the board, so dividing by the typical separation on an $N \times N$ grid places all grid sizes on a common scale.

A.3 Verifier Details

The verifier ranks candidates from their intermediate predictions $\hat{x}_0$ using only two image-space points supplied by the task instance: p_{start}, where the object that should move begins, and p_{goal}, where it should end up. On Frozen Lake and VR-Bench these are the centers of the start and goal cells, taken from the instance metadata. The verifier receives no obstacle map or grid structure.

Trajectory Extraction. The tracker, as described in Appendix A.1 starts at p_{start} and, in each frame, takes the centroid of the moving pixels inside a disc centered on its current position. This window is wide enough to follow a cell-to-cell move but keeps the tracker on the agent rather than on other motion in the frame, such as flickering obstacle tiles. The final position of the track is p_{end}.

Score. A candidate is scored by how much closer to the goal the tracked agent ends than it began:

$$\text{score}\left(\hat{x}_0^{(\tau)}\right) = 1 - \frac{d(p_{\text{end}}, p_{\text{goal}})}{d(p_{\text{start}}, p_{\text{goal}})} \quad \text{where} \quad d(x, y) := \|x - y\|_2^2, \quad x, y \in \mathbb{R}^2 \tag{5}$$

with distances in pixels. The score is 1 when the agent ends on the goal, 0 when it ends as far from the goal as it started, and negative when it ends farther away. For a specific maze, the denominator is constant, so the ranking depends only on how close p_{end} lands to the goal; the normalization serves to make scores comparable across instances. EPBS completes the K highest-scoring candidates. For our best-of-N comparison, it uses this verifier to rank fully denoised videos by the same score.

A.4 Evaluation and Success Criterion.

Maze Evaluation. When evaluating mazes for success we use SAM 2 tracking described above. A generated video is considered successful if the extracted trajectory satisfies: (1) the trajectory ends at the goal cell; (2) no intermediate cell falls on a hole or maze border (constraint violation). We use this permissive criterion to accept all valid solutions rather than comparing against ground-truth optimal paths. A common behavior on easier mazes is constraint violations after solving the maze: because these models are trained to produce smooth video throughout the 5-second window, they continue producing motion after the agent has already reached the goal. We handle this by truncating the trajectory at the first valid goal visit. Additionally, cases where the elf remains stationary or oscillates between two cells are classified as degenerate failures.

LIBERO Evaluation. LIBERO scenes have no grid, so we cannot decide success from a tracked trajectory. We instead judge each final video using Gemini 3 Flash (thinking level *low*, temperature 0). The judge receives the task instruction, the list of objects the task requires to move (read from the goal predicates of the task's BDDL file), and 32 frames: 24 spaced uniformly over the video and the final 8 consecutive frames. In addition to these frames, for each object that needs to move, the LLM receives a close-up panel of three crops taken from the simulator's object boxes: the object's starting location in the first frame, the same location in the last frame, and its goal in the last frame. The rest of the prompt is specified in Appendix A.6. A video succeeds only if every required object has left its starting location and rests on or in its goal position, no new objects are created, and the arm itself carries out the task. The judge evaluates semantic task completion rather than exact physical fidelity: the arm must be used to complete the task but its motion need not match the robot's kinematics or be executable in the simulator.

To validate the judge, we hand-labeled 55 videos from the evaluation pool without seeing the model's verdicts. The judge agreed with the human label on 51 of 55 videos (92.7%, Cohen's $\kappa = 0.81$), with 3 false passes and 1 false fail.

A.5 Generation Hyperparameters

Table 3 summarizes the generation settings. Both models use image+text-to-video conditioning. We pad the condition images to 16:9 aspect ratio (832×480 for Wan, 480p for Hunyuan) using centered black borders. Wan2.2-14B (Wan et al., 2025) uses the UniPC scheduler with shift 5.0. HunyuanVideo-1.5 (Wu et al., 2025) uses its native Euler flow-matching scheduler with 8 step-distilled inference steps. We do not change any of the default hyperparameters for these models.

Table 3: **Generation hyperparameters** for both video diffusion models.

Parameter	Wan2.2-14B	HunyuanVideo-1.5
Resolution	832×480	480p (848×480)
Frame count	81	121
FPS	16	24
Guidance scale	3.5	6.0
Scheduler	UniPC	Euler (flow matching)
Denoising steps (T)	40	8 (step-distilled)
Compute	4× Nvidia L40	4× Nvidia L40

A.6 TEXT PROMPTS

We include the exact prompts we used to query the models for each benchmark.

Frozen Lake Prompt

Animate the elf moving step by step toward the gift while carefully avoiding the icy frozen lake. Highlight the successful path and end with the elf touching the gift. There are no changes to the layout of the maze. No new lakes or characters appear. Static camera. No zoom. No pan. No glitches, noise, or artifacts.

VR-Bench Trapfield Prompt Template

Create a 2D animation based on the provided image of a maze. The {`player`} slides smoothly along the {`floor`} path, stopping perfectly on the {`goal`}. The {`player`} never slides into or crosses the {`trap`} (trap areas). The camera is a static, top-down view showing the entire maze.

Maze: The maze paths are {`floor`}, and the trap areas are {`trap`}. The {`player`} moves to the goal position, represented by the {`goal`}. The {`player`} slides smoothly along the {`floor`} path. The {`player`} never slides into or crosses the {`trap`} of the maze. The {`player`} stops perfectly on the {`goal`}.

Scene: No change in scene composition. No change in the layout of the maze. The {`player`} travels along the {`floor`} path without speeding up or slowing down.

Camera: Static camera. No zoom. No pan. No glitches, noise, or artifacts.

Skins:

1. `player`=blue circle, `goal`=green circle, `trap`=red x, `floor`=white square
2. `player`=blue parka explorer & penguin, `goal`=red flag, `trap`=blue water pool, `floor`=blue ice crystals
3. `player`=blue adventurer, `goal`=golden eagle emblem, `trap`=fiery explosion, `floor`=gray stone bricks
4. `player`=gray robot, `goal`=golden star, `trap`=blue crystal block, `floor`=silver metal plate

VR-Bench Maze Prompt Template

Create a 2D animation based on the provided image of a maze. The {`player`} slides smoothly along the {`floor`} path, stopping perfectly on the {`goal`}. The {`player`} never slides or crosses into the {`wall`} areas of the maze. The camera is a static, top-down view showing the entire maze.

Maze: The maze paths are {`floor`}, the walls are {`wall`}. The {`player`} moves to the goal position, represented by {`goal`}. The {`player`} slides smoothly along the {`floor`} path. The {`player`} never slides or crosses into the {`wall`} areas of the maze. The {`player`} stops perfectly on the {`goal`}.

Scene: No change in scene composition. No change in the layout of the maze. The {`player`} travels along the {`floor`} path without speeding up or slowing down.

Camera: Static camera. No zoom. No pan. No glitches, noise, or artifacts.

Skins:

1. `player`=red circle, `goal`=green square, `wall`=light blue square, `floor`=white square
2. `player`=white rabbit, `goal`=orange carrots, `wall`=gray rock, `floor`=green grass tiles
3. `player`=blue robot head, `goal`=green and yellow tile, `wall`=black brick wall, `floor`=gray decorative tile
4. `player`=anime schoolgirl, `goal`=green square, `wall`=gray stone wall, `floor`=wooden floor tiles
5. `player`=green circle, `goal`=red circle, `wall`=blue potion bottle, `floor`=white square

LIBERO Prompt Template

Single-stage task: Create a video of a robotic arm performing a tabletop task. The robot arm {`stage`}. The arm moves smoothly and deliberately. All other objects remain stationary throughout. The camera is static with no zoom or pan.

Multi-stage task: Create a video of a robotic arm performing {`n`} tabletop tasks in sequence. First, the robot arm {$\texttt{stage}_1$}. Then, the robot arm {$\texttt{stage}_2$}. [Finally, the robot arm {$\texttt{stage}_3$}.] The arm moves smoothly and deliberately. All other objects remain stationary throughout. The camera is static with no zoom or pan.

Stages, one per goal predicate of the task's BDDL file:

1. *Place* (on / in / under): reaches down, grasps the {`object`}, lifts it, moves it {`onto | into | under`} the {`target`}, and releases it
2. *Close*: reaches out, grasps the {`drawer`}, and pushes it closed
3. *Open*: reaches out, grasps the {`drawer`}, and pulls it open
4. *Turn on / off*: reaches out and turns the {`knob`} to switch it {`on | off`}
5. *Push*: reaches out and pushes the {`object`} until it rests {`relation`} the {`target`}

{`object`} and {`target`} are taken from LIBERO's task instruction, keeping any qualifier that tells identical objects apart (e.g., "the right moka pot").

LIBERO Judge Prompt (Gemini 3 Flash)

You are judging whether a robot arm completed a tabletop task in a generated video.

Task: "{`instruction`}"
Objects the task requires to be moved (report one entry for each, in this order): {`1. the object`$_1$`, 2. the object`$_2$`, ...`}

For each object to move you get a close-up panel (in the object order above) with three crops: where it started in the first frame, the same spot in the last frame, and its goal in the last frame. Answer `still_at_start_at_end` from the first two crops and `resting_on_goal_at_end` from the third. Trust the close-ups over your overall impression of the video.

You are given 32 video frames in temporal order: the first 24 evenly spaced across the whole video, then the last 8 consecutive frames from the very end.

The task counts as done only if all of these hold at the end of the video:
1. *Same object.* Each required object is the one already present in the first frame. Simulator names may not match colour; identify objects by their position. If the object is still at its starting spot at the end, it has not been moved – an identical-looking object somewhere else is a different object (invented, duplicated, or produced by the arm) and does not count.
2. *Resting on its goal.* Each object ends supported by its goal – resting on or in it, so it would stay there if the gripper vanished. The gripper still touching it in the last frame is fine. An object held up above its goal, hovering, or being lowered but not yet down when the video ends has not been placed. If the goal is closed at the end (a drawer), confirm from the video frames that the object went in before it closed.
3. *Clearly where the task says.* Spatial words are strict: "to the right of the plate" means plainly beside the plate on its right, not behind it and not barely displaced; "in the basket" means inside it, not on its rim. An object that barely moves has not been placed.
4. Every other part of the task (closing a drawer, turning a knob) is visibly done.

Judge the end state, not how realistic the motion looked. If the video is too corrupted to judge at all, set `unscorable` = true.

Notes. {`instruction`} is LIBERO's instruction with asset names removed and "black bowl" written as "bowl" (the asset renders light gray). Each panel follows the prompt, headed "Close-up panel: the {`object`}"; the 32 frames follow, headed "Video frames, in temporal order:". The reply is a JSON schema: per object `still_at_start_at_end`, `resting_on_goal_at_end`, `ends_where_task_requires`; then `new_object_appeared`, `other_task_parts_done`, `success` (plus location, reason, confidence, unscorable). We count a success only if `success` is true and every other check agrees: each object left its start and rests on its goal as required, no object appeared, and the other task parts are done.

B Failure Analysis

We propose chaining in Section 5 because we find required path length to be the strongest correlate of failure on mazes of size 6×6 and larger. Specifically, we quantify these horizon-limited failures in Wan2.2 by looking directly at the effect of path length on success. Failures are driven by trajectory length rather than obstacle density: from 6×6 upward, per-maze success is strongly anti-correlated with the optimal path length (r from -0.58 to -0.78), while the fraction of lake cells has near-zero correlation at every size ($|r| \leq 0.06$; Tab. 4).

In horizon-limited failures, rather than producing an incomplete but valid prefix, the model often "cheats" to solve the maze by any means possible. The model moves the gift closer to the agent or spawns a second agent near the goal (Figure 8). The model preserves the intent of the task (reaching the goal), but violates the underlying environment constraints to achieve it. With an increased required path length the model faces a conflict between its early plan commitment and its generation horizon.

Table 4: **Maze difficulty.** Solve rate falls with trajectory length, not obstacle density. Success is EPBS at 400 NFE with the goal-only verifier; correlations are Pearson r between per-maze success and, respectively, optimal path length and the fraction of hole cells in the board.

Size	4×4	6×6	8×8	10×10
Success (%)	96.1	56.4	32.5	18.3
Mean path length	4.8	7.6	10.9	14.7
Path corr.	-0.11	-0.58	-0.78	-0.68
Lake corr.	0.05	0.06	-0.04	0.06

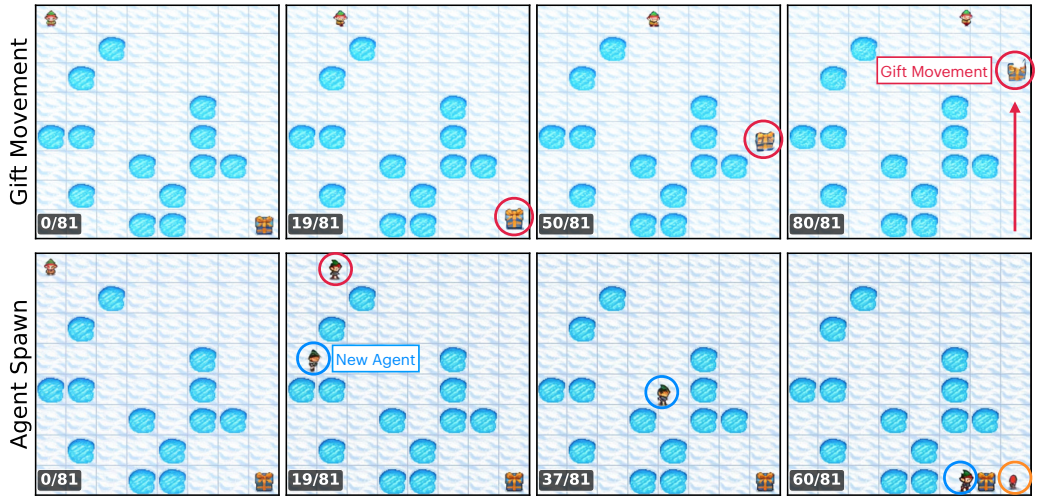


Figure 8: **The model "cheats" on hard mazes.** When the trajectory is too long to complete within the generation window, the model sacrifices structural adherence to fulfill the prompt. *Top:* the gift teleports from the far corner to an adjacent cell, allowing the agent to "solve" the maze without traversing the full path. *Bottom:* a second agent spawns near the goal and reaches it, while the original agent remains stranded.

C EPBS Sensitivity Analysis

We perform more analysis of EPBS on Wan2.2-14B, sweeping our hyperparameters and justifying our method even when taking into account extra NFEs from VAE decode time.

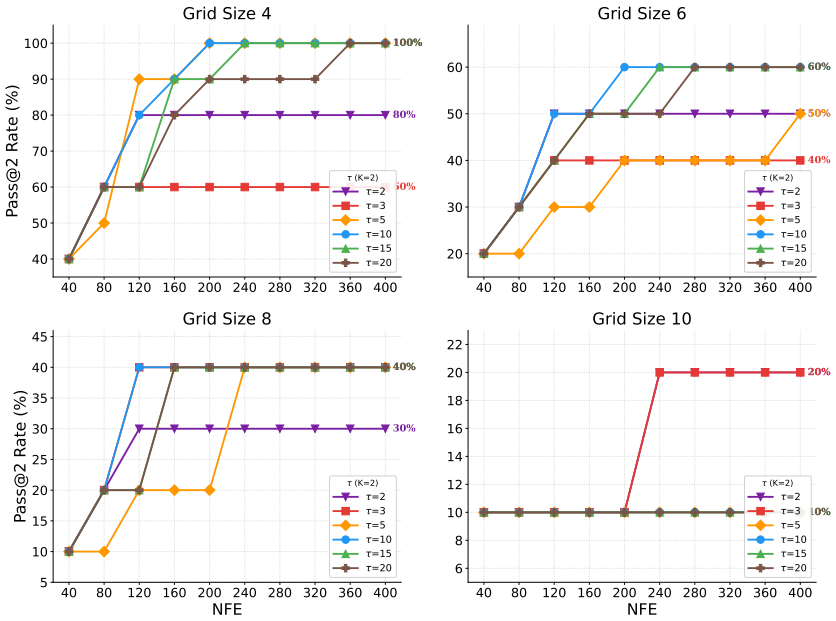


Figure 9: **Probe step sensitivity.** Pass@2 vs. NFE budget for probe steps $\tau \in \{2, 3, 5, 10, 15, 20\}$ at beam size $K = 2$, across four grid sizes. All configurations use shared seed pools for fair comparison. $\tau = 5$ provides the best efficiency trade-off for size 4; larger mazes are less sensitive to τ.

C.1 Sweep on Probe Step τ

Figure 9 shows EPBS accuracy as a function of NFE budget for six probe step values $\tau \in \{2, 3, 5, 10, 15, 20\}$ with beam size $K = 2$ and a fixed random seed generator. We use a randomly selected subset of 10 mazes per size. For size 4, $\tau = 5$ reaches peak accuracy at the lowest NFE; $\tau = 2$ underperforms because the $\hat{x}_0$ prediction at that stage carries insufficient trajectory information. For sizes 6–10, sensitivity to τ decreases, though $\tau = 10$ tends to be a safe choice and very early probing ($\tau = 2$) consistently underperforms. The optimal τ reflects a trade-off: too early, and the prediction lacks discriminative signal; too late, and the probing cost approaches full generation.

C.2 Sweep on Beam Size K

The beam size K controls how many top-scoring seeds are fully denoised after probing. Figure 10 shows the effect of varying K from 1 to 5 while holding τ fixed. At low budgets, higher K values are constrained: since each full completion costs $T-\tau$ steps, a limited budget leaves no room for the extra completions that larger K demands, so all configurations reduce to completing the same small number of seeds and their curves overlap. As budget increases, K=2 is the most robust choice: on size 4, K=2 achieves 100% by NFE 160, while K=1 plateaus at 90%. On size 6, K=2 reaches 60% vs. 40% for K=1. Larger beam sizes ($K \geq 3$) provide no additional benefit and can even reduce accuracy at moderate budgets by consuming NFE on completions rather than probing more seeds. This confirms that K=2 optimally balances exploration (probing diverse seeds) against exploitation (completing promising candidates).

C.3 Wall-Clock Comparison

We compare wall-clock time at matched accuracy levels rather than matched NFE budgets, since EPBS trades additional NFE for better seed selection. All timings are measured on 4×L40 GPUs with FSDP and sequence parallelism.

Each EPBS probe consists of τ denoising steps, VAE decoding of the $\hat{x}_0$ prediction, and background-difference verifier scoring. Each completion runs the remaining $T-\tau$ denoising steps and VAE

Table 5: **Wall-clock comparison at matched accuracy.** For each grid size, we report EPBS accuracy and wall-clock time at NFE = 120, alongside the smallest best-of-N budget that reaches the same accuracy and its wall-clock time. Speedup = best-of-N time / EPBS time. Sizes 4–6 use τ=5; sizes 8–10 use τ=15. Both methods use the goal-only verifier and top-2 selection, and every time includes denoising, VAE decoding, verifier scoring, and SAM 2 evaluation.

		EPBS @ NFE 120		**Best-of-N to match**		
Size	τ	Acc (%)	Time (min)	NFE needed	Time (min)	Speedup
4×4	5	85.5	26.2	400	77.9	2.97×
6×6	5	42.3	26.2	360	70.1	2.68×
8×8	15	16.9	22.5	200	39.0	1.73×
10×10	15	8.5	22.5	120	23.5	1.04×

decodes the final video. One denoising step takes $\sim$11.2 s. VAE decoding takes $\sim$0.3 min per video, for both an $\hat{x}_0$ probe and a final video; scoring a decoded video with the goal-only verifier adds a negligible amount of time. SAM 2 trajectory evaluation (0.1 min per seed) is applied to every completed seed in both EPBS and baseline. A full baseline generation (40 steps plus VAE decode) takes 7.8 min.

Table 5 shows that at NFE = 120, EPBS achieves accuracy that the baseline requires NFE = 120–400 to match, yielding 1.04–2.97× wall-clock speedup. The speed advantage is largest on small mazes where EPBS's screening is most effective.

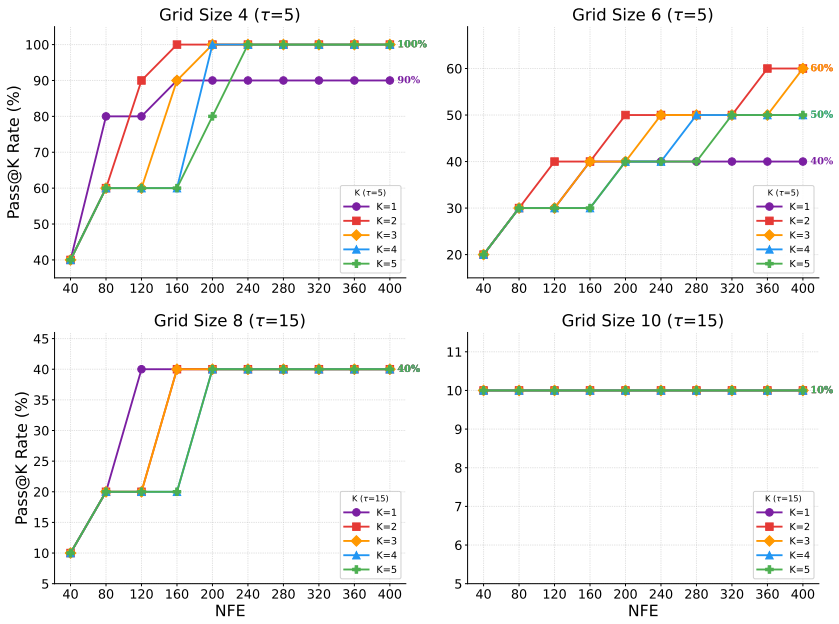


Figure 10: **Beam size ablation.** Pass@K vs. NFE budget for beam sizes $K \in \{1, 2, 3, 4, 5\}$ with fixed probe step ($\tau = 5$ for sizes 4–6, $\tau = 15$ for sizes 8–10). K=2 provides the best trade-off: it reaches peak accuracy at lower budgets than K=1 without the probe-count penalty of larger K values.

D Extended Analysis

D.1 Cross-Model Early Plan Commitment

In Sec. 3 our analysis reveals early plan commitment for Wan2.2 across maze sizes. Here, we provide the same form of measurement for HunyuanVideo-1.5 along with Wan2.2 on the same set of mazes from sizes 4×4–8×8.

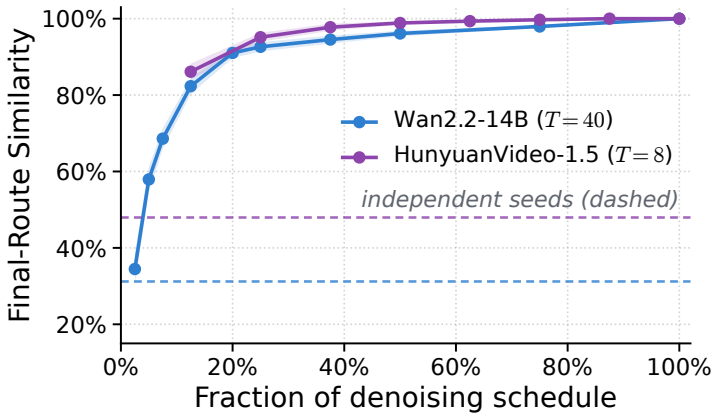


Figure 11: **Cross-model early plan commitment.** Mean final-route similarity between the route in each intermediate $\hat{x}_0$ prediction and the route in the final video, against the fraction of the denoising schedule completed, averaged across Frozen Lake mazes of sizes 4×4–8×8 (shaded: 95% CI).

On 565 trajectories, Fig. 11 shows the same pattern for both models: similarity rises sharply in the first part of the schedule and then plateaus. After 12.5% of the schedule (step 5 of 40 for Wan2.2, step 1 of 8 for HunyuanVideo-1.5), similarity is 84.0% and 86.1%; after 25% it is 93.3% and 95.1%. Both are far above the similarity between independent seeds on the same maze (34.8% and 48.0%; the Wan2.2 value differs slightly from the 35.3% in Sec. 3 because this comparison uses the 565-trajectory subset shared by both models). The two models reach nearly the same similarity at the same fraction of their schedules, even though HunyuanVideo-1.5 takes one fifth as many steps. Here we see early plan commitment is not specific to one architecture, scheduler, or step count.

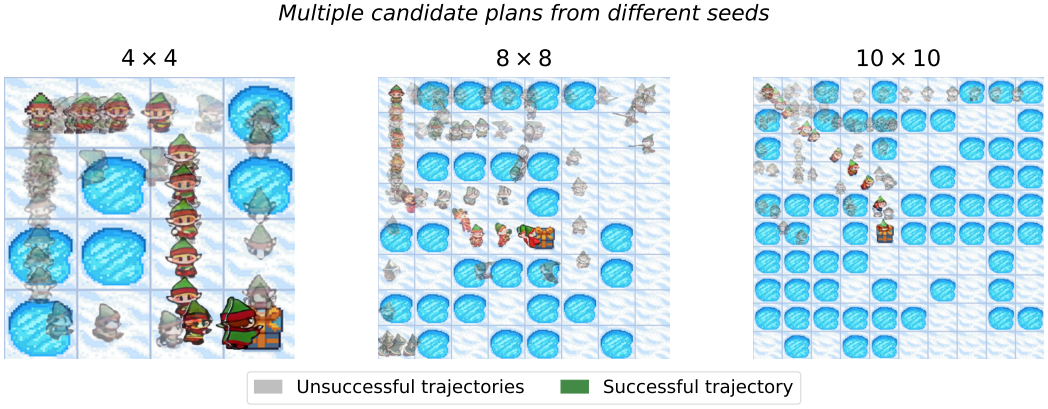


Figure 12: **Diversity of candidate plans under different noise seeds.** We overlay trajectories extracted from multiple samples of the same maze. Incorrect trajectories are shown in gray with reduced opacity, while a successful trajectory is shown in color. Despite sharing identical conditioning, different seeds produce diverse motion plans. Importantly, these plans are distinguishable early in the denoising process, suggesting that inference-time compute should be allocated toward selecting promising trajectories rather than refining all candidates.

E ADDITIONAL QUALITATIVE EXAMPLES

We provide qualitative examples of early commitment, successful chaining, and failure modes. The early commitment gallery compares decoded $\hat{x}_0$ predictions at intermediate denoising steps with the final generated video. The chaining gallery shows successive generation segments and their stitched outputs, while the failure gallery shows representative unsuccessful trajectories. Videos are visualized using ghost trails.

E.1 EARLY COMMITMENT GALLERY

Each row below shows a ghost-trail visualization of a single seed's decoded $\hat{x}_0$ prediction at denoising steps $\tau \in \{2, 5, 15, 20\}$ and the final generated video. The trajectory is visible by τ=5 and remains stable through later steps, confirming early plan commitment. We distinguish *norm* mazes (goal at the far corner, maximizing path length) from *vary* mazes (randomly placed goal, often admitting shorter solutions). HunyuanVideo-1.5 uses a step-distilled schedule with T=8 total steps, so its rows show $\hat{x}_0$ at $\tau \in \{1, 3, 5, 7\}$ instead.

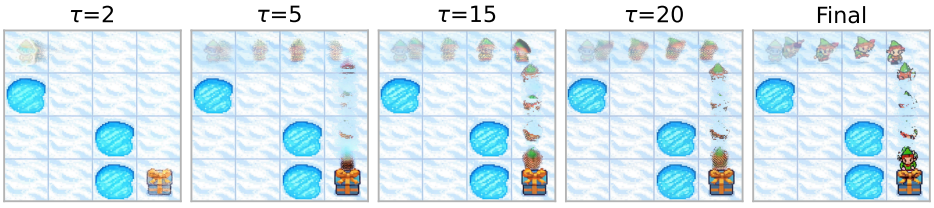


Figure 13: **Wan2.2, size 4 (norm).** The full solution path is already visible in the τ=5 prediction; subsequent steps only sharpen the rendering without altering the planned route.

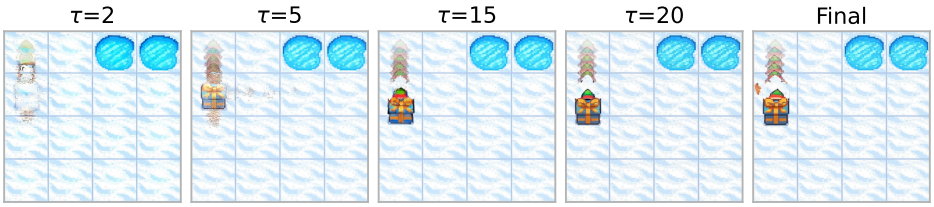


Figure 14: **Wan2.2, size 4 (vary).** The model commits to the route by τ=5 and maintains it through the final video.

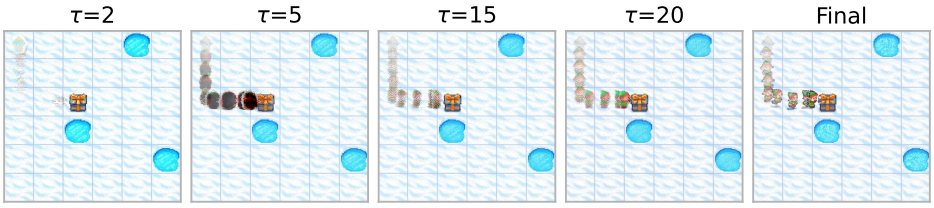


Figure 15: **Wan2.2, size 6 (vary).** A longer maze requiring more steps. Despite the increased path complexity, the overall route direction is locked in by τ=5.

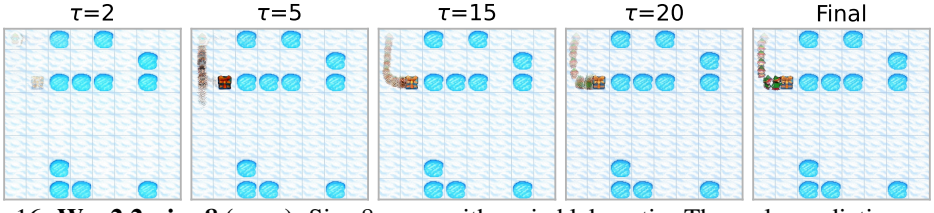


Figure 16: **Wan2.2, size 8 (vary).** Size 8 maze with varied lake ratio. The early prediction captures the general trajectory shape, though fine-grained cell-level details continue to refine through later steps.

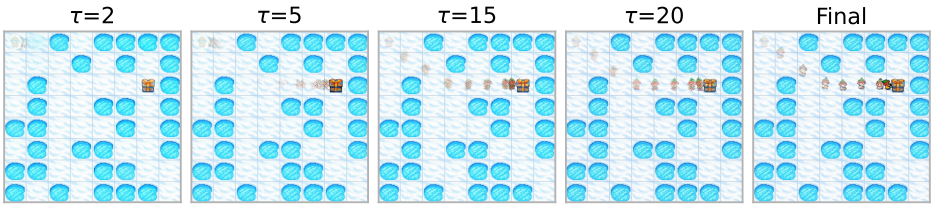


Figure 17: **Wan2.2, size 8 (vary), second example.** The ghost trail at τ=5 already traces a path consistent with the final video, demonstrating that early commitment holds across different maze instances.

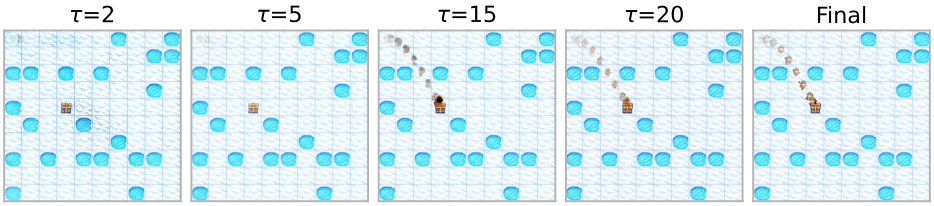


Figure 18: **Wan2.2, size 10 (vary).** The largest maze size. Even for this 10×10 grid, the model's planned trajectory is apparent by step 5 of denoising.

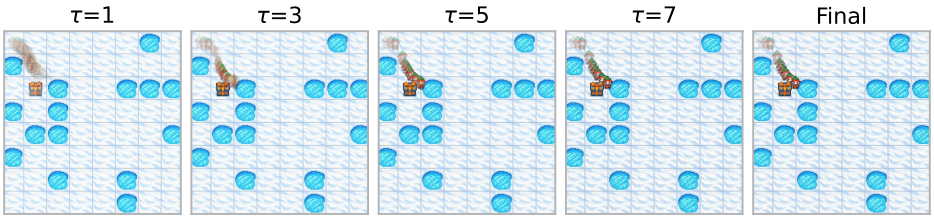


Figure 19: **HunyuanVideo-1.5, size 8 (vary).** HunyuanVideo uses a step-distilled schedule (T=8). Despite the shorter schedule, the trajectory is committed by step 3 and remains stable, mirroring the early commitment seen in Wan2.2.

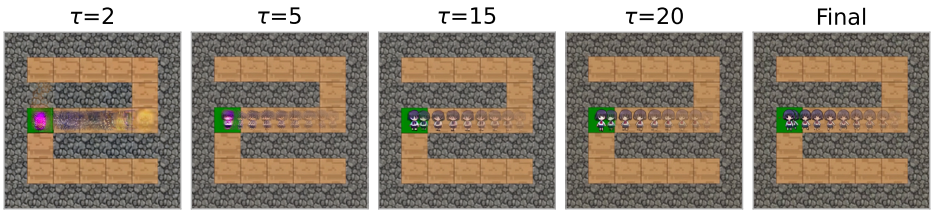


Figure 20: **VR-Bench, maze_4 (easy).** Early plan commitment on a VR-Bench maze with a distinct brown/tan texture. The trajectory direction is already visible at τ=5 and refines through later steps, showing that commitment generalizes beyond the Frozen Lake visual style to procedurally generated maze textures.

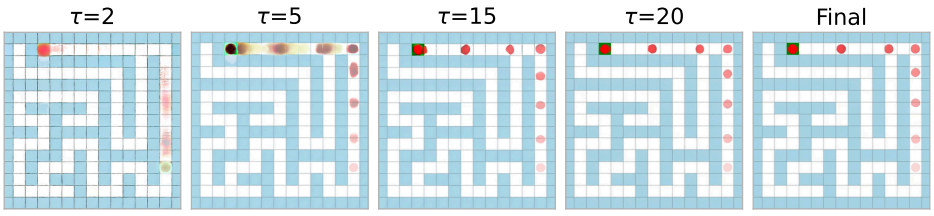


Figure 21: **VR-Bench, maze_1 (hard).** A harder VR-Bench puzzle on the blue/teal maze_1 texture. Despite higher path complexity, the $\hat{x}_0$ prediction at τ=5 already captures the coarse trajectory shape, which persists through to the final generation.



Figure 22: **VR-Bench, maze_2 (easy).** The maze_2 texture uses green grass and gray rocks. The ghost trail at τ=2 is largely noise, but by τ=5 the planned path is clearly committed and matches the final output.

E.2 CHEAP GALLERY

Each row shows the ghost-trail of each chain segment, with the first panel as the pivot seed followed by the chain segment and the final stitched video.



Figure 23: **Wan2.2 chain, size 6 (norm), lake 20, maze 002.** Two-segment chain.



Figure 24: **Wan2.2 chain, size 10 (vary), lake 20, maze 002.** Size 10 maze with varied lake ratio. The longer solution path necessitates chaining.

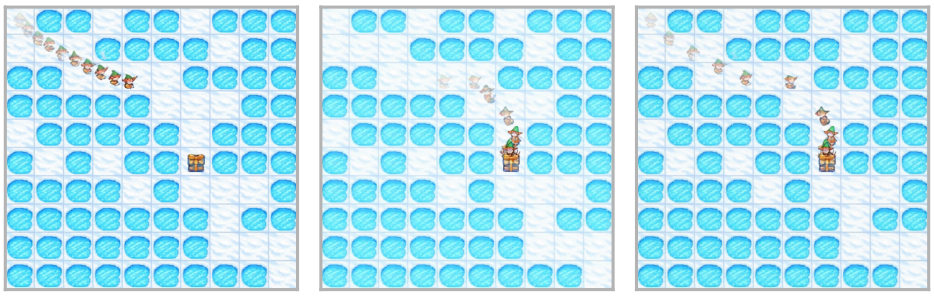


Figure 25: **Wan2.2 chain, size 10 (vary), lake 80, maze 003.** High lake density leaves fewer safe cells, creating a narrow corridor that chaining navigates successfully.



Figure 26: **Wan2.2 chain, VR-Bench maze_3_medium, puzzle 0004.** Chaining applied to a VR-Bench maze with a different visual texture than Frozen Lake. The agent navigates a medium-difficulty procedurally generated maze across multiple segments.

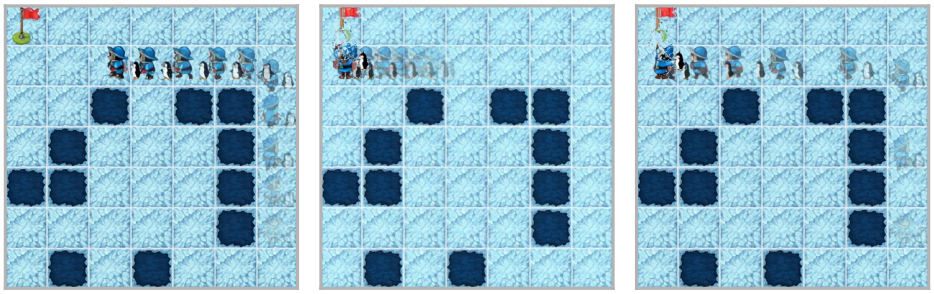


Figure 27: **Wan2.2 chain, VR-Bench trapfield_2_medium, puzzle 0002.** A trapfield environment where the agent must avoid trap cells (analogous to lakes). Chaining enables completion of longer trapfield paths.

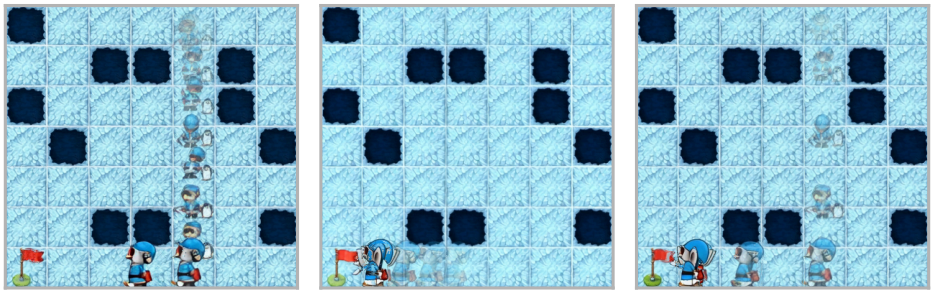


Figure 28: **Wan2.2 chain, VR-Bench trapfield_2_medium, puzzle 0005.** Another trapfield example showing successful multi-segment navigation through a different puzzle layout.



Figure 29: **HunyuanVideo-1.5 chain, size 4 (vary), lake 65, maze 004.** Chaining with HunyuanVideo's step-distilled schedule (T=8). The pivot and chain segments are stitched to form a complete solution.



Figure 30: **HunyuanVideo-1.5 chain, size 6 (norm), lake 65, maze 004.** A size 6 maze with 65% lake density. Chaining extends HunyuanVideo's effective planning horizon beyond a single generation.



Figure 31: **HunyuanVideo-1.5 chain, size 6 (vary), lake 80, maze 009.** Despite the constrained environment, chaining produces a valid multi-segment solution.

E.3 Failure Mode Gallery

Ghost-trail visualizations of representative failures from both models and domains. Left two panels: Wan2.2; right two panels: HunyuanVideo-1.5.

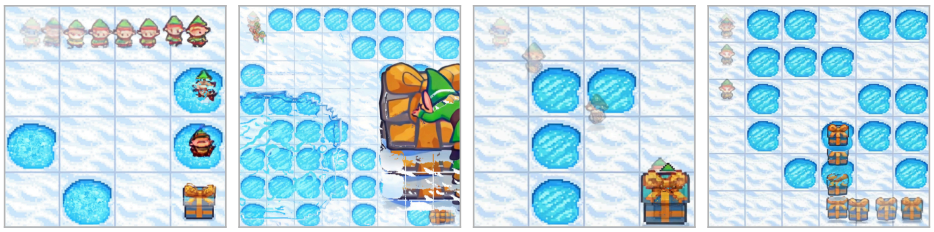


Figure 32: **Constraint violations — Frozen Lake.** Ghost-trail visualizations showing two sub-types: *lake entry* (the elf crosses into a frozen-lake cell, violating the environment constraint) and *gift movement* (the goal tile shifts position during generation).

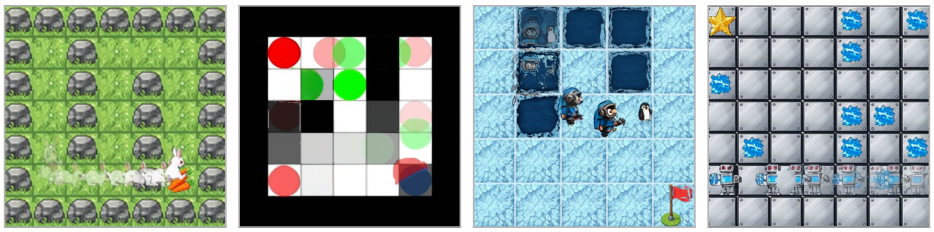


Figure 33: **Constraint violations — VR-Bench.** The agent enters a forbidden cell (wall or trap) in procedurally generated mazes and trapfields. Each panel uses a different texture to illustrate that the failure mode is consistent across visual appearances.

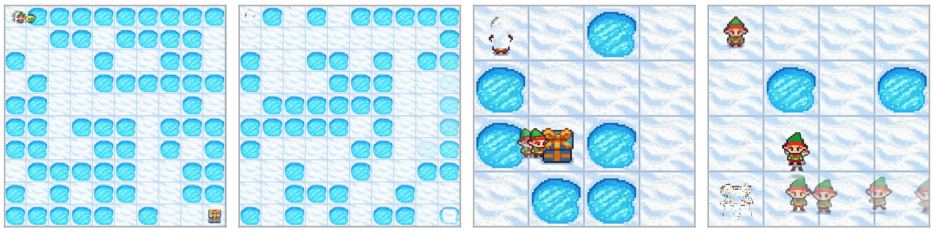


Figure 34: **Degenerate failures.** The model produces a video with little meaningful agent motion. The ghost trail collapses to a single position, indicating that the elf remains static throughout generation.



Figure 35: **Horizon-limited failures.** The generated trajectory respects all constraints but fails to reach the goal within the generation window (81 frames for Wan2.2; 121 for HunyuanVideo). *Valid stall*: the agent follows a correct prefix but stops moving before reaching the goal. *Wrong route*: the agent takes a legal but sub-optimal path and runs out of frames.