SpecAR: Distributed Target Blocks for Fast Autoregressive Video Generation

Hudeifa Hassan
Statite

Abstract

Autoregressive video generation repeatedly invokes a large target model over long rollouts, incurring substantial computational cost. Speculative approaches reduce this cost by assigning most blocks to a smaller drafter and invoking the target only on selected blocks. Existing methods rely on online perceptual routing to select target blocks, yet we find that per-block scoring provides no statistically significant quality improvement over random acceptance at matched rates. Controlled placement ablations further demonstrate that evenly spaced target blocks outperform router-selected and random placements at fixed compute, while minimizing the longest uninterrupted draft run. Based on this principle, we present **SpecAR**, a training-free mixed-rollout decoder that predetermines evenly spaced target anchors before generation without learned routers, auxiliary reward models, or runtime scoring overhead. Evaluated on Krea Realtime 14B and Helios 14B, SpecAR achieves up to **10.54× speedup over target-only generation and 1.84× over SDVG**, while matching or exceeding SDVG across all evaluated operating points on VisionReward and Restricted VBench Quality-6 and retaining up to 99.7% of target-only quality. Project page: `https://statite.ai/specar/`

1 Introduction

Autoregressive video models (Huang et al., 2025a) generate video sequentially in blocks, conditioning each new block on the preceding generation. Producing long rollouts requires repeated execution of a large target model, making inference computationally expensive. Speculative video generation reduces this cost by delegating most blocks to a smaller drafter while reserving target-model computation for selected positions along the rollout. This reduces the frequency of expensive target-model evaluations, but leaves the allocation of a limited target-generation budget as a central design problem.

Existing speculative video methods, such as SDVG (Hu & Zhang, 2026), use online image-quality scores to decide when to invoke the target model. This routing incurs substantial scoring overhead yet yields no statistically significant quality gain over random block selection at a matched target budget. The underlying limitation is that quality degradation arises from cumulative trajectory drift across consecutive draft blocks, which individual block scores fail to capture. Online routing therefore permits long uninterrupted draft runs in which errors compound, making the spacing of target interventions essential to controlling drift.

Generation quality depends on both the number and placement of target blocks. At a fixed target budget, quality degrades as the longest uninterrupted draft run increases across placement strategies, linking generation quality to the spacing of target interventions (Appendix C.5). Increasing the target budget, however, does not consistently improve quality: it improves results on Krea Realtime and Helios but reduces quality on MAGI-1 (Appendix D). Target-budget selection must therefore account

Preprint.

for the relative quality of the two models, while target-block placement must control how long the drafter runs without intervention.

These observations establish two distinct decisions: how many blocks the target should generate and where those blocks should be placed. The budget depends on the relative generation quality of the target and drafter. For a given budget, evenly spaced target blocks minimize the longest uninterrupted draft run under the rollout constraints, by a short counting argument (Appendix B). Target positions can therefore be determined before generation, eliminating the need for online scoring.

We introduce **SpecAR**, a training-free mixed-rollout decoder that implements this principle through a predetermined, evenly distributed schedule. Given a rollout horizon of H blocks and a target budget of k, SpecAR assigns exactly k blocks to the target model, with the remaining $H - k$ blocks generated by the drafter. Target positions are fixed before generation and spaced as evenly as possible across the rollout, minimizing the longest uninterrupted draft run for the specified budget. Unlike methods that select target blocks through online quality assessment, SpecAR requires no learned router, auxiliary reward model, or per-block scoring. By separating target-budget selection from block placement, SpecAR provides explicit control over target computation without introducing additional selection overhead.

We evaluate SpecAR on two target–drafter pairs in which the target sets the quality ceiling: Krea Realtime 14B (Millon & Krea AI, 2025) with a Self-Forcing 1.3B drafter, and Helios 14B (Yuan et al., 2026) with its distilled 14B drafter. Across the evaluated target budgets, SpecAR achieves 2.04–2.67× and 4.37–10.54× speedups over target-only generation, respectively, while running faster than SDVG at each corresponding operating point with no loss in quality. On Krea Realtime, SpecAR retains up to 99.7% of target-only Restricted VBench Quality-6, and on Helios its quality rises with the target budget. A negative case study on MAGI-1 24B with its 4.5B drafter (Appendix D) shows quality falling as the target budget grows. These results demonstrate the efficiency of fixed target-block scheduling while highlighting the dependence of generation quality on target-budget allocation.

This paper makes the following contributions:

- We demonstrate that balanced target placement outperforms online perceptual routing and random placement under matched target budgets, establishing target-block scheduling as an important design dimension for autoregressive video generation.
- We introduce SpecAR, a training-free mixed-rollout decoder that evenly distributes target blocks, minimizing the longest uninterrupted draft run without online scoring or auxiliary reward models.

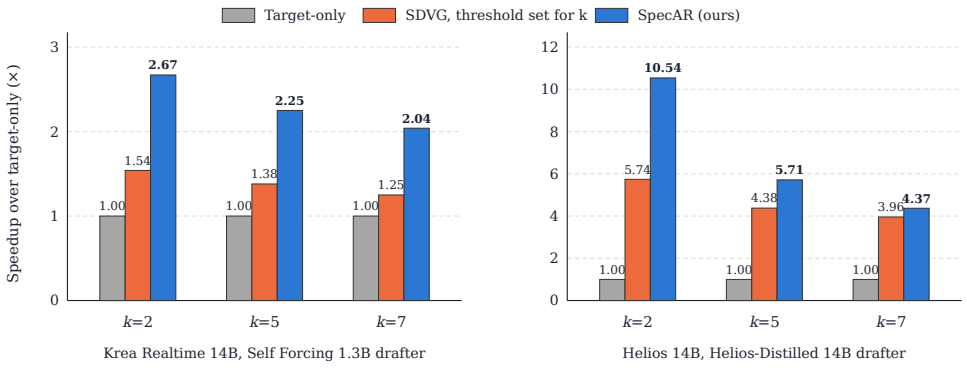


Figure 1: Speedup over target-only generation for SpecAR at each target budget k and for SDVG at the corresponding operating point (threshold selected using held-out prompts to approximate SpecAR's draft fraction; SDVG's realized acceptance differs, see Table 1), on two target–drafter pairs; each panel has its own axis. SpecAR is 2.04–2.67× faster than the target on Krea Realtime and 4.37–10.54× on Helios, and faster than SDVG at every budget. Full results in Table 1 and Table 2.

- We demonstrate quality–latency improvements across Krea Realtime and Helios. SpecAR achieves 2.04–2.67× and 4.37–10.54× speedups over target-only generation, respectively, while outperforming SDVG in latency and matching or exceeding its evaluated quality metrics. A MAGI-1 case study further reveals the importance of target-budget selection.

2 Related Work

2.1 Speculative Decoding

Speculative decoding accelerates autoregressive generation by using a smaller model to propose future outputs and a larger target model to verify them. In language models, this verification is defined over a shared discrete vocabulary, allowing proposed tokens to be evaluated directly against the target distribution. Early methods preserve the target distribution through modified rejection sampling (Leviathan et al., 2023; Chen et al., 2023), while later work improves proposal quality or reduces serialization in the draft–verify loop. Medusa (Cai et al., 2024) predicts multiple future tokens with auxiliary heads, EAGLE (Li et al., 2024) and EAGLE-3 (Li et al., 2025) construct stronger drafters from target-model features, DFlash (Chen et al., 2026) uses block-diffusion drafting, and Speculative Speculative Decoding (Kumar et al., 2026) overlaps drafting with verification. Despite these differences, they operate in the same basic regime: the proposal consists of discrete tokens for which the target model provides a tractable token-level distribution. SpecAR addresses a different setting. In block-autoregressive video diffusion, each autoregressive step produces a continuous video block, so the token-level verification mechanism used in language-model speculative decoding is not directly available.

2.2 Speculative Decoding for Visual Autoregressive Models

Speculative decoding has also been extended to visual autoregressive models that generate discrete visual tokens. LANTERN (Jang et al., 2025) and GSD (So et al., 2025) relax token-level verification to account for redundancy among visually similar tokens, while SJD (Teng et al., 2025) and SCD (So et al., 2026) use Jacobi-style decoding without a separately trained drafter; SCD additionally demonstrates large speedups on token-autoregressive video generation. VVS (Dong et al., 2026) reduces target evaluations by skipping verification on a scheduled subset of steps. Despite these differences, all of these methods retain a discrete token sequence for which target probabilities are available, allowing verification to be exact, relaxed, or selectively skipped. The discrete-token verification framework does not extend directly to continuous video blocks. In block-autoregressive video generation, target allocation instead determines which model generates each committed block of the evolving trajectory.

2.3 Speculative Decoding for Block-Autoregressive Video

Block-autoregressive video diffusion generates a continuous latent block at each autoregressive step, as in Self-Forcing (Huang et al., 2025a) and Wan-style models (Team Wan et al., 2025). Unlike discrete-token generation, these models do not expose a tractable target probability for each generated block, so standard token-level acceptance tests cannot be applied directly. SDVG (Hu & Zhang, 2026) addresses this by generating candidate blocks with a smaller drafter and using an auxiliary perceptual model to decide online whether each candidate should be retained or regenerated by the target. SpecAR changes this allocation mechanism. The target budget and target positions are fixed before generation, eliminating content-dependent routing and its associated scoring cost. This allocation also differs from methods such as T-Stitch (Pan et al., 2025), which switch between small and large models across denoising timesteps within a single sample. SpecAR instead allocates the target across autoregressive blocks, so the choice of model determines which blocks are committed to the trajectory and therefore condition subsequent generation.

3 Method

SpecAR replaces online target-block selection with a fixed, balanced schedule determined before generation. Given a rollout horizon and a target-generation budget, it distributes target blocks across

the sequence while using the drafter for the remaining blocks. This section describes the mixed-rollout execution, the scheduling rule, and the resulting bound on uninterrupted draft runs.

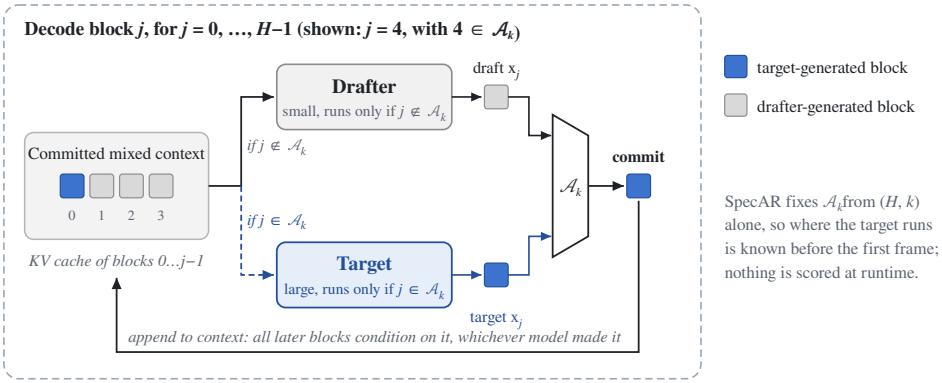


Figure 2: SpecAR decoding. The target schedule $\mathcal{A}_k$ is fixed from H and k before generation, with block 0 always assigned to the target. At each step j, the schedule dispatches generation to the target if $j \in \mathcal{A}_k$ and to the drafter otherwise. The committed block is appended to the mixed context, so every later block, from either model, conditions on it.

3.1 Mixed-Rollout Decoding

Consider an autoregressive video generator that produces a sequence of H latent video blocks using a large target model and a smaller drafter. A target-generation budget k specifies how many blocks are generated by the target, with the remaining $H - k$ blocks generated by the drafter. Because the target schedule $\mathcal{A}_k$ is fixed before generation begins, each block $j \in \{0, \ldots, H-1\}$ is dispatched directly to its assigned model: at anchor positions ($j \in \mathcal{A}_k$) the target generates and commits the block, and at all remaining positions ($j \notin \mathcal{A}_k$) the drafter generates and commits it. Both models condition on the previously committed context, so every block influences subsequent generation regardless of which model produced it, and target anchors continue from the current mixed rollout rather than reconstructing a separate target-only trajectory. The choice of target positions therefore determines how target and draft outputs are interleaved, and different placements can produce different rollouts even when the target budget is unchanged. Within a run of consecutive drafted blocks, each block conditions on blocks the drafter has just committed, so drafter errors carry forward through the run. A target block does not restore the target-only trajectory, but the run that follows it starts from context that includes target output. Placement therefore sets how many drafted blocks can follow one another, the quantity bounded in Section 3.3.

3.2 Balanced Target-Block Placement

SpecAR determines target positions using only the rollout horizon H and target budget k. The initial block is always generated by the target, and the remaining $k - 1$ target blocks are distributed as evenly as possible across the sequence. For $1 \le k \le H$, the target schedule is defined as

$$\mathcal{A}_k = \left\{ \left\lfloor \frac{jH}{k} + \frac{1}{2} \right\rfloor \;\middle|\; j = 0, \ldots, k-1 \right\}.$$

Blocks whose indices belong to $\mathcal{A}_k$ commit the target output; all other blocks commit the drafter output. For example, with $H = 36$ and $k = 7$, SpecAR assigns target blocks at positions $0, 5, 10, 15, 21, 26$, and 31, limiting the longest uninterrupted draft run to five blocks.

Because the schedule depends only on H and k, all target positions are known before generation begins. SpecAR therefore requires no learned router, auxiliary reward model, or online quality scoring to determine where target computation is allocated.

3.3 Bounded Draft Runs

For a fixed target budget, different placement strategies produce draft runs of different lengths. Concentrating target blocks near the beginning of a rollout can leave a long sequence generated exclusively by the drafter, whereas balanced placement distributes these draft runs more uniformly.

SpecAR minimizes the longest uninterrupted draft run among schedules containing k target blocks with the initial block fixed to the target. Its balanced schedule achieves

$$L_{\max}(\mathcal{A}_k) = \left\lceil \frac{H}{k} \right\rceil - 1.$$

To see this, the $H - k$ drafted blocks are partitioned into k runs following the target anchors. No placement can make the longest run shorter than $\lceil (H-k)/k \rceil$, and the balanced schedule attains this bound. For instance, with $H = 18$ and $k = 2$, placing both target blocks at the beginning leaves a draft run of 16 blocks, whereas placing them at positions 0 and 9 reduces the longest run to eight blocks. Appendix B writes out this counting argument for every H and k.

This bound characterizes the structure of the schedule rather than guaranteeing generation quality. Section 4.5 and Appendix C.5 compare balanced placement with random, concentrated and router-selected schedules at the same target budget; in each comparison the balanced schedule, which has the shortest longest run, scores highest.

4 Evaluation

4.1 Experimental Setup

Models and architectures. We evaluate SpecAR on two streaming block-autoregressive video diffusion models, which commit and decode each block before generating the next; this is the setting speculative video generation targets, and we do not evaluate non-streaming autoregressive generators. The first pairs Krea Realtime 14B (Millon & Krea AI, 2025), a Wan2.1-T2V-14B model adapted for causal block-autoregressive generation through Self Forcing, with the released Self-Forcing 1.3B DMD drafter (Huang et al., 2025b). The second pairs Helios-Base 14B (Yuan et al., 2026) with Helios-Distilled 14B, a distilled few-step variant of the same model. Helios is a chunk-autoregressive diffusion transformer fine-tuned from Wan2.1-T2V-14B that generates 33-frame chunks conditioned on a multi-scale window of previous chunks and a first-frame anchor; the target denoises each chunk with 50 steps and classifier-free guidance, while the drafter uses six steps over three pyramid stages without guidance. Both pairs use their original checkpoints without additional training. A third pair, MAGI-1 24B with MAGI-1 4.5B, is reported as a negative case study in Appendix D.

Platforms and implementation. Krea Realtime and Helios experiments run on a single NVIDIA H100 80GB GPU. All configurations use BF16 precision and FlashAttention-3 (Shah et al., 2024). Krea Realtime uses lazy target KV caching, avoiding eager construction of target KV state for drafter-generated blocks. SpecAR directly dispatches each block to its assigned model according to

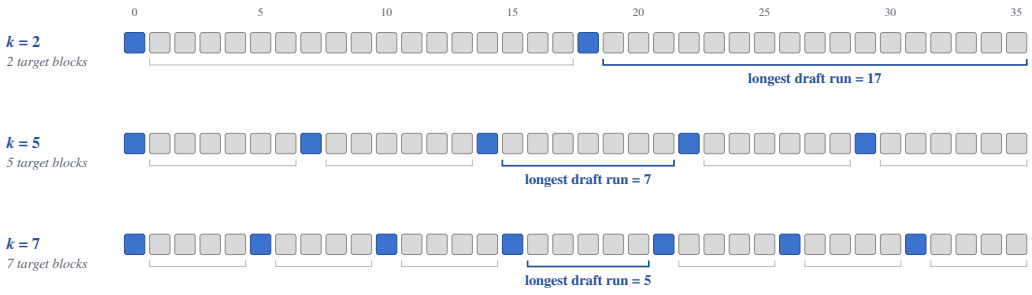


Figure 3: SpecAR target placement at $H = 36$ for the budgets evaluated in Section 4. Spreading the target blocks evenly keeps every draft run at most $\lceil H/k \rceil - 1$ blocks long: 17, 7 and 5 for $k = 2$, 5 and 7.

the predetermined schedule, without generating discarded draft candidates at target positions. SDVG uses its released implementation. On Helios, both models remain resident on the GPU, and each chunk is decoded as it is committed. SpecAR requires no checkpoint modifications or additional training.

Baselines. On each target–drafter pair, we compare SpecAR against two baselines under otherwise identical configurations (prompts, seeds, rollout horizon, denoising schedule, precision, and hardware). *Target-only* generates every block with the target model and provides the reference for quality and latency. *SDVG* (Hu & Zhang, 2026) uses an auxiliary perceptual model, ImageReward (Xu et al., 2023), to decide online whether draft candidates are accepted or regenerated by the target. We use its released implementation and select acceptance thresholds on held-out prompts to approximate SpecAR's target-generation budgets. This comparison evaluates fixed, balanced scheduling against content-dependent online routing, including its scoring overhead. Draft-only generation is additionally evaluated on Krea Realtime.

Workloads and metrics. We evaluate 36-block autoregressive rollouts on 150 prompts per model pair; every method, including target-only and draft-only generation, is run on the same 150 prompts, and SDVG thresholds are selected on a separate set of held-out prompts. We sweep the target budget $k \in \{2, 5, 7\}$, corresponding to 5.6–19.4% target-generated blocks. This horizon exposes the effect of target placement: concentrating seven target blocks at the beginning leaves 29 consecutive draft blocks, whereas balanced placement limits the longest draft run to five. We additionally evaluate $k = 9$ in the routing ablation. Krea Realtime generates 26.6-second 832 $\times$ 480 videos from MovieGenVideoBench text prompts (Polyak et al., 2024), and Helios generates 74-second 640 $\times$ 384 videos at 16 fps from HeliosBench text prompts (Yuan et al., 2026). Quality is measured by whole-video VisionReward (Xu et al., 2026) and Restricted VBench Quality-6 (Huang et al., 2024): VBench's Quality Score restricted to six dimensions (subject consistency, background consistency, motion smoothness, dynamic degree, aesthetic quality and imaging quality), with each dimension rescaled by VBench's leaderboard normalization and dynamic degree weighted one half.

VisionReward is computed over 24 frames spanning each video to capture quality throughout the rollout. For systems performance, we report end-to-end generation latency, real-time factor (RTF), and speedup over target-only generation. Latency measures elapsed time from prompt submission to the final decoded frame, including denoising, routing, VAE decoding, and data transfers, but excluding model loading. On Krea Realtime, prompts are text-encoded once before timing begins (about 50 ms per video, the same for every method). RTF is the generated video duration divided by latency, averaged across videos. Speedup is computed as the ratio of target-only mean latency to the method's mean latency.

4.2 Main Results

Table 1 summarizes the main operating points on both target–drafter pairs. On Krea Realtime, SpecAR achieves 2.04-$2.67\times$ speedup over target-only generation and is 1.63-$1.73\times$ faster than the corresponding SDVG configurations across all evaluated target budgets. On Helios, where the drafter is a few-step distillation of the target, SpecAR achieves 4.37-$10.54\times$ speedup over target-only generation and is 1.10-$1.84\times$ faster than SDVG at every matched budget, with higher VisionReward at each of them.

The target budget k sets a single quality–speed trade-off. Increasing k from 2 to 7 lowers SpecAR's speedup over target-only generation from 2.67$\times$ to 2.04$\times$ and raises VisionReward from 0.0876 to 0.0972; at $k = 5$ and 7, VisionReward exceeds target-only generation (0.0914). Across the whole range, SpecAR is faster than every SDVG configuration: its slowest setting ($k = 7$, 2.04$\times$) is faster than SDVG's fastest (1.54$\times$), and at every matched budget it also has a higher mean VisionReward (Figure 4, left). Helios follows the same trade-off (Figure 4, right): raising k from 2 to 7 lowers the speedup from 10.54$\times$ to 4.37$\times$ and raises VisionReward from 0.1223 to 0.1375. SpecAR's margin over SDVG narrows with k on this pair, from 1.84$\times$ to 1.10$\times$, because the router's per-decision cost is small next to a Helios target chunk, yet SDVG remains slower at every budget while retaining fewer draft blocks than its target budget allows.

4.3 Quality

Figure 5 shows how the target-generation budget affects quality on both pairs. On Krea Realtime, Restricted VBench Quality-6 rises from 77.25 at $k = 2$ to 77.85 at $k = 7$, approaching the target-only score of 78.06. Helios follows the same direction: Quality-6 rises from 77.05 to 77.73 and whole-video VisionReward from 0.1223 to 0.1375 over the same budget range, and at every budget SpecAR scores above the corresponding SDVG configuration on both metrics (Table 2). The trend reverses on MAGI-1 24B with its 4.5B drafter: Quality-6 falls from 75.87 to 72.17 over the same range (Appendix D). Allocating more target computation therefore does not necessarily improve generation quality; the trade-off depends on the relative quality of target and drafter.

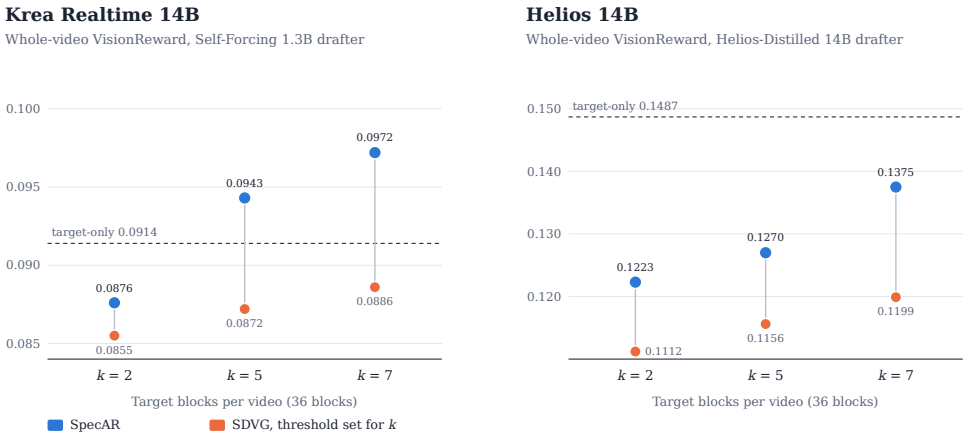


Figure 4: Whole-video VisionReward versus target budget on Krea Realtime (left) and Helios (right) at $H = 36$, for SpecAR and for SDVG with its threshold set for the same budget. Dashed lines mark target-only quality. Across $k \in \{2, 5, 7\}$, SpecAR is faster than the corresponding SDVG configuration on both pairs, with VisionReward at least as high.

Table 1: End-to-end quality and performance of SpecAR and SDVG on Krea Realtime and Helios (one H100 each), with $H = 36$. SDVG thresholds are selected using held-out prompts to approximate SpecAR's draft fraction $(H - k)/H$. VR denotes whole-video VisionReward, RTF is generated video duration divided by latency, and Accept is the fraction of draft-generated blocks retained. Speedups are relative to target-only generation. Helios (1,185 frames at 16 fps): SDVG runs the residual form of its router with thresholds selected on held-out prompts. Helios memory is the generation process's peak allocation with both models resident; SDVG rows add the 2.4 GB of its scorer, which runs as a separate process.

Model	Method	VR	RTF	Latency (s)	Speedup	Accept (%)	Mem (GB)
Krea 14B Self Forcing 1.3B drafter 832×480 · 36 blocks · 1×H100	Target-only	0.0914	0.3039	87.6	1.00×	—	45.50
	Draft-only	0.0828	0.9175	29.0	3.02×	—	45.38
	SDVG τ_1' (-2.2696)	0.0855	0.4711	56.8	1.54×	96	48.87
	SpecAR (k=2)	0.0876	0.8111	32.8	**2.67**×	94	45.38
	SDVG τ_3' (-2.1560)	0.0872	0.4353	63.5	1.38×	92	48.87
	SpecAR (k=5)	0.0943	0.6827	39.0	**2.25**×	86	45.38
	SDVG τ_4' (-1.9430)	0.0886	0.3986	70.3	1.25×	88	48.88
	SpecAR (k=7)	0.0972	0.6214	42.9	**2.04**×	81	45.38
Helios 14B Helios-Distilled 14B drafter 640×384 · 36 chunks · 1×H100	Target-only	0.1487	0.0254	2911.5	1.00×	—	65.7
	Draft-only	0.1085	0.6002	123.4	23.59×	—	65.7
	SDVG τ_2 (-1.190)	0.1112	0.1460	507.3	5.74×	93	68.1
	SpecAR (k=2)	0.1223	0.2681	276.2	**10.54**×	94	65.7
	SDVG τ_5 (-0.859)	0.1156	0.1113	665.4	4.38×	87	68.1
	SpecAR (k=5)	0.1270	0.1453	509.6	**5.71**×	86	65.7
	SDVG τ_7 (-0.713)	0.1199	0.1007	735.8	3.96×	84	68.1
	SpecAR (k=7)	0.1375	0.1111	666.5	**4.37**×	81	65.7

At $k = 9$ on Krea Realtime, SpecAR achieves a whole-video VisionReward of 0.1005, compared with 0.0875 for the corresponding SDVG configuration and 0.0914 for target-only generation. On Restricted VBench Quality-6, SpecAR retains 99.8% of the target-only score (77.87 versus 78.06).

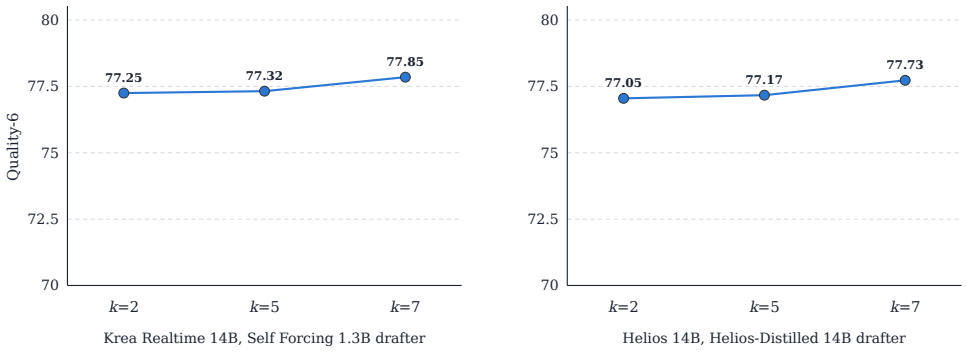


Figure 5: Restricted VBench Quality-6 versus target budget k for SpecAR on Krea Realtime and Helios ($H = 36$). Increasing k from 2 to 7 raises Quality-6 from 77.25 to 77.85 on Krea Realtime and from 77.05 to 77.73 on Helios. The two pairs use distinct evaluation workloads; full metric breakdowns appear in Table 2, and the MAGI-1 case study in Appendix D shows the opposite trend.

Table 2: Generation quality across target-only, SDVG, and SpecAR operating points on both model pairs. Restricted VBench Quality-6 aggregates six VBench quality dimensions with VBench's leaderboard normalization and weights (§4.1). Whole-video VisionReward is reported separately. Higher scores indicate better performance.

Model	Method	Quality-6	Subject consist.	Background consist.	Motion smooth.	Dynamic degree	Aesthetic	Imaging	VisionReward
Krea 14B Self Forcing 1.3B drafter	Target-only	78.06	83.74	88.70	98.29	82.00	58.56	69.11	0.0914
	Draft-only	76.77	83.96	88.61	97.85	73.00	57.89	68.56	0.0828
	SDVG τ_1' (-2.2696)	76.95	83.29	89.17	97.74	78.00	57.73	67.63	0.0855
	SpecAR (k=2)	77.25	83.74	89.41	97.84	77.00	58.23	68.09	0.0876
	SDVG τ_3' (-2.1560)	77.01	83.42	89.23	97.76	78.00	57.68	67.73	0.0872
	SpecAR (k=5)	77.32	84.47	89.74	97.79	75.00	58.34	68.23	0.0943
	SDVG τ_4' (-1.9430)	76.91	83.44	89.26	97.86	76.00	57.67	67.78	0.0886
	SpecAR (k=7)	77.85	84.61	89.92	97.92	78.00	58.70	68.39	0.0972
Helios 14B Helios-Distilled 14B drafter	Target-only	78.98	80.45	90.00	97.52	100.00	64.90	63.59	0.1487
	Draft-only	76.95	84.77	92.06	98.11	83.33	60.58	55.20	0.1085
	SDVG τ_2 (-1.190)	75.89	85.06	92.04	98.41	66.67	60.30	56.64	0.1112
	SpecAR (k=2)	77.05	83.73	91.11	98.32	83.33	60.67	57.43	0.1223
	SDVG τ_5 (-0.859)	75.88	84.90	92.04	98.37	66.67	60.48	56.70	0.1156
	SpecAR (k=5)	77.17	85.40	91.90	98.29	75.00	62.39	57.61	0.1270
	SDVG τ_7 (-0.713)	75.69	85.39	92.52	98.36	58.33	61.83	57.31	0.1199
	SpecAR (k=7)	77.73	86.63	92.13	98.30	75.00	62.69	58.62	0.1375

4.4 Systems Performance

We isolate the sources of SpecAR's performance gains on Krea Realtime at $H = 36$ and $k = 9$, using a single H100 with lazy target KV caching. Table 3a breaks down end-to-end generation latency by execution stage. SpecAR completes generation in 46.84 s, compared with 70.95 s for SDVG, a $1.51\times$ speedup. SDVG spends 26.16 s (37% of its total latency) on online routing, including candidate decoding for scoring, ImageReward evaluation, and host transfers. In contrast, SpecAR incurs no routing cost, while drafter, target, and VAE execution times remain comparable. Eliminating this routing path accounts for the observed latency improvement, rather than any acceleration of the underlying model computations.

Table 3b examines whether this advantage translates into faster video streaming. SpecAR and SDVG have nearly identical time to first frame (2.097 versus 2.098 s), but SpecAR reduces mean block-to-block latency from 1.915 to 1.193 s (38%) and the 95th-percentile interval from 3.737 to 2.792 s (25%). Consequently, steady-state real-time factor increases from 0.392 to 0.629. The benefit therefore appears primarily during sustained generation rather than at the first frame.

Table 3c measures aggregate generation throughput as the number of concurrent requests increases. SpecAR sustains 9.50–9.61 output frames/s, compared with 7.77–7.91 frames/s for SDVG and 4.95 frames/s for target-only generation. Throughput remains approximately constant as concurrency

increases because requests share a single model instance and GPU generation is serialized; additional requests overlap only queuing and host-side work. SpecAR's throughput improvement therefore comes from reducing per-generation execution cost, not from increased GPU batch parallelism.

Table 3a: End-to-end latency breakdown for Krea Realtime on one H100 ($H = 36$, SpecAR $k = 9$, lazy target KV caching). Times are reported in milliseconds per video, except total latency in seconds. SDVG spends 26.16 s on online routing, while SpecAR eliminates this overhead with comparable drafter, target, and VAE execution costs.

Method	Drafter	Target	Router	VAE decode	Context building	Frame delivery & host	Total = latency (s)
Target-only	—	73,558	—	9,869	604	4,586	88.62
SDVG (τ_5')	14,297	17,008	26,163	9,800	832	2,851	70.95
SpecAR	14,156	18,039	**—**	9,819	659	4,167	**46.84**

Table 3b: Single-request streaming performance on Krea Realtime ($H = 36$, SpecAR $k = 9$). Each generated block represents 0.75 s of video. We report time to first frame, mean and 95th-percentile block delivery intervals, and real-time factors. SpecAR reduces mean block-to-block latency from 1.915 s to 1.193 s relative to SDVG, with nearly identical time to first frame.

Method	Time to first frame (s) ↓	Block-to-block mean (s) ↓	Block-to-block p95 (s) ↓	Steady-state RTF ↑	End-to-end RTF ↑
Target-only	1.803	2.384	2.396	0.315	0.300
SDVG (τ_5')	2.098	1.915	3.737	0.392	0.375
SpecAR	2.097	**1.193**	**2.792**	**0.629**	**0.568**

Table 3c: Aggregate serving throughput on one H100 as request concurrency increases from 1 to 16, measured in generated frames per second across all requests. SpecAR sustains 9.50–9.61 frames/s, compared with 7.77–7.91 for SDVG. Throughput remains nearly constant because GPU generation is serialized across requests; concurrency overlaps only queuing and host-side operations.

Method	1 request	4 requests	16 requests	Video-seconds per second
Target-only	4.95	4.95	4.95	0.31
SDVG (τ_5')	7.86	7.91	7.77	0.49
SpecAR	**9.50**	**9.61**	**9.61**	**0.60**

4.5 Ablation

Table 4 isolates the effect of online routing at $H = 36$ on Krea Realtime and Helios. We replace SpecAR's fixed schedule with content-dependent, router-selected positions, keeping the generation configuration otherwise fixed. Replaying these positions without online scoring isolates the resulting generation quality, while enabling the router measures the cost of making the same decisions during generation. Online routing adds 15–25 s per video on Krea Realtime and about 120 s on Helios without changing the generated outputs or their quality. Router-selected placements also yield lower VisionReward across the evaluated operating points, although their realized target counts differ from SpecAR's fixed budgets. These experiments separate the quality effects of target placement from the execution overhead of online selection.

Earlier measurements of SDVG's released router on Krea Realtime point the same way (Appendix C.4). At the most aggressive threshold of its published sweep ($\tau = -2.5$), the router accepted every drafted block at $H = 36$, so only the forced first block was generated by the target. At a matched 75% acceptance rate, random acceptance scored as high as router-guided acceptance (VisionReward 0.07313 versus 0.07160, paired $t = +0.75$). Across the matched rates, the router's advantage over random acceptance is at most 0.0015 VisionReward at 95% confidence. A five-way placement comparison at fixed budget ($k = 9$, Appendix C.5, Table C5) further confirms that balanced spacing (0.1157 VR, $L_{\max} = 3$) outperforms random (0.1080), front-loaded (0.1020), router top-k (0.1001) and back-loaded (0.0966) schedules at identical compute.

5 Discussion and Limitations

Approximate generation. Unlike exact speculative decoding for language models, SpecAR does not preserve the target model's output distribution. Target and draft blocks are committed directly into a shared autoregressive trajectory without a distribution-preserving acceptance test. Although balanced placement minimizes the longest uninterrupted draft run, this structural guarantee does not imply a bound on generation error or guarantee target-equivalent quality. Quality preservation is therefore evaluated empirically.

Content-agnostic scheduling. SpecAR fixes target positions before generation, eliminating online selection overhead but preventing adaptation to the evolving video content. Difficult motion, appearance changes, or accumulated generation errors may benefit from additional target interventions at particular positions. Balanced placement controls the spacing of these interventions rather than identifying where they are most needed.

Target-budget selection. SpecAR assumes a specified target-generation budget rather than optimizing it. Our results show that increasing target involvement improves quality on Krea Realtime but can reduce quality on MAGI-1, indicating that the appropriate budget depends on the target–drafter pair. Jointly selecting the target budget and its placement remains an open direction.

6 Conclusion

We show that online perceptual routing incurs substantial overhead without a statistically significant quality advantage over random acceptance at matched rates. Controlled placement experiments further demonstrate that evenly spaced target blocks outperform random and router-selected placements at fixed compute. Building on these findings, we introduce SpecAR, a training-free decoder that replaces online routing with balanced target scheduling, minimizing uninterrupted draft runs without auxiliary scoring. Across Krea Realtime and Helios, SpecAR achieves up to 10.54× speedup over target-only generation and 1.84× over SDVG, while matching or exceeding SDVG on both quality metrics. Our MAGI-1 results further establish that target-budget selection depends on the relative quality of the target and drafter. These findings demonstrate that effective target allocation requires controlling both the amount and placement of target computation, rather than relying on per-block perceptual scoring.

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st*

Table 4: Ablation of online routing on Krea Realtime and Helios (one H100 each, $H = 36$; Krea Realtime with lazy target KV caching). Each model compares three configurations of the same mixed-rollout decoder: SpecAR's fixed, evenly spread schedule; the router's selected positions replayed without online scoring; and online routing (SDVG). Quality is whole-video VisionReward; Δ quality is relative to SpecAR at the same budget. Router-selected schedules average 6.7 and 8.5 target-generated blocks at $k = 7$ and $k = 9$ on Krea Realtime, and 6.4 and 7.9 on Helios; maximum draft-run lengths are averaged across videos. Speedups are relative to target-only generation. Latencies in this table come from a separate timing run.

Model	Method	Online router	Max draft run	Quality ↑	Δ quality	Latency (s) ↓	Speedup
Krea 14B Self Forcing 1.3B drafter 832×480 · 36 blocks · 1×H100	**SpecAR (k=7)**	×	**5**	**0.0973**	—	41.15	2.09×
	Router schedule, replayed (k=7)	×	26	0.0886	−0.0087	40.40	2.13×
	Online router, SDVG (k=7)	✓	26	0.0886	−0.0087	55.55	1.55×
	SpecAR (k=9)	×	**3**	**0.1005**	—	46.84	1.89×
	Router schedule, replayed (k=9)	×	24	0.0875	−0.0130	45.88	1.93×
	Online router, SDVG (k=9)	✓	24	0.0875	−0.0130	70.95	1.25×
Helios 14B Helios-Distilled 14B drafter 640×384 · 36 chunks · 1×H100	**SpecAR (k=7)**	×	**5**	**0.1375**	—	666.5	4.37×
	Router schedule, replayed (k=7)	×	22	0.1199	−0.0176	617.8	4.71×
	Online router, SDVG (k=7)	✓	22	0.1199	−0.0176	735.8	3.96×
	SpecAR (k=9)	×	**3**	**0.1395**	—	819.2	3.55×
	Router schedule, replayed (k=9)	×	18	0.1216	−0.0179	734.2	3.97×
	Online router, SDVG (k=9)	✓	18	0.1216	−0.0179	853.6	3.41×

International Conference on Machine Learning (ICML), volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235, 2024. arXiv:2401.10774.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.06036.

Haotian Dong, Ye Li, Rongwei Lu, Chen Tang, Shu-Tao Xia, and Zhi Wang. VVS: Accelerating speculative decoding for visual autoregressive generation via partial verification skipping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2511.13587.

Yuezhou Hu. spec-video-gen: Speculative decoding for autoregressive video generation (code). `https://github.com/yuezhouhu/spec-video-gen`, 2026. Branch `dev`, commit `e78618fb1ef8`; used as the SDVG routing and scoring implementation.

Yuezhou Hu and Jintao Zhang. Speculative decoding for autoregressive video generation. *arXiv preprint arXiv:2604.17397*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a. arXiv:2506.08009.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self-Forcing released checkpoints. Hugging Face repository `https://huggingface.co/gdhe17/Self-Forcing`, 2025b. File `checkpoints/self_forcing_dmd.pt` (DMD-distilled chunk-wise autoregressive Wan2.1-T2V-1.3B), revision `2f8b7792`, SHA-256 `a0413986...8f56a3`, 5,676,252,553 bytes.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818, 2024. doi: 10.1109/CVPR52733.2024.02060. arXiv:2311.17982.

Doohyuk Jang, Sihwan Park, June Yong Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. LANTERN: Accelerating visual autoregressive models with relaxed speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.03355.

Tanishq Kumar, Tri Dao, and Avner May. Speculative speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2603.03251.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286, 2023. arXiv:2211.17192.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948, 2024. arXiv:2401.15077.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2503.01840.

Erwann Millon and Krea AI. Krea Realtime 14B: Real-time, long-form AI video generation. Technical blog post, `https://www.krea.ai/blog/krea-realtime-14b`, October 2025. Wan2.1-T2V-14B distilled into a causal autoregressive model with Self Forcing. Model release: Hugging Face `krea/krea-realtime-video`, file `krea-realtime-video-14b.safetensors`, revision `9a2ceea2`, SHA-256 `792f6042...27eec7`, 28,577,102,160 bytes; code `github.com/krea-ai/realtime-video`.

Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, and Anima Anandkumar. T-Stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2402.14167.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Sand AI. MAGI-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2407.08608.

Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15375–15384, 2025. doi: 10.1109/ICCV51701.2025.01426. arXiv:2508.07747.

Junhyuk So, Hyunho Kook, Chaeyeon Jang, and Eunhyeok Park. Speculative coupled decoding for training-free lossless acceleration of autoregressive visual generation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. arXiv:2510.24211.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative Jacobi decoding. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.01699.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2304.05977.

Jiazheng Xu, Yu Huang, Jiale Cheng, Yuanming Yang, Jiajun Xu, Yuan Wang, Wenbo Duan, Shen Yang, Qunlin Jin, Shurun Li, Jiayan Teng, Zhuoyi Yang, Wendi Zheng, Xiao Liu, Dan Zhang, Ming Ding, Xiaohan Zhang, Xiaotao Gu, Shiyu Huang, Minlie Huang, Jie Tang, and Yuxiao Dong. VisionReward: Fine-grained multi-dimensional human preference learning for image and video generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11269–11277, 2026. doi: 10.1609/aaai.v40i13.38107. arXiv:2412.21059.

Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. *arXiv preprint arXiv:2603.04379*, 2026.

A Algorithm

Notation. Let M_T and M_D denote the target and drafter, and z_i the committed context when generating block i. A schedule $\mathcal{A} \subseteq \{0, \dots, H-1\}$ with $|\mathcal{A}| = k$ has anchors $0 = a_1 < \cdots < a_k < H$, with $a_{k+1} = H$; the draft runs are $r_j = a_{j+1} - a_j - 1$, so $\sum_j r_j = H - k$ and $L_{\max}(\mathcal{A}) = \max_j r_j$.

A.1 SpecAR Rollout

Algorithm 1 gives the complete SpecAR rollout. The anchor set is computed once from (H, k) before generation. At each block i, the schedule dispatches generation directly to the target if $i \in \mathcal{A}_k$ and to the drafter otherwise, updating only the active model's context cache from the committed trajectory.

Algorithm 1 SpecAR rollout

Require: horizon H; budget k; conditioning c; noise $\epsilon_0, \dots, \epsilon_{H-1}$; models M_T, M_D
$\mathcal{A}_k \leftarrow \{\lfloor jH/k + 1/2 \rfloor \ : \ j = 0, \dots, k-1\}$
for $i = 0, \dots, H-1$ **do**
  $z_i \leftarrow$ committed context window ending at block $i-1$
  if $i \in \mathcal{A}_k$ **then**
    update the target context cache from z_i
    $x_i \leftarrow M_T(z_i, c, \epsilon_i)$
  else
    update the drafter context cache from z_i
    $x_i \leftarrow M_D(z_i, c, \epsilon_i)$
  end if
  decode x_i with the streaming VAE and emit its frames
end for

Each active model updates its cache from the committed context z_i. Consequently, a target anchor continues the mixed target–drafter trajectory rather than restarting from a target-only trajectory.

B Proofs and Analysis

B.1 Minimum Longest Draft Run

We show that for $1 \le k \le H$ and any schedule $\mathcal{A}$ with $|\mathcal{A}| = k$ and $0 \in \mathcal{A}$, $L_{\max}(\mathcal{A}) \ge \lceil (H-k)/k \rceil$, and that the balanced schedule $\mathcal{A}_k$ of §3.2 attains this bound.

Lower bound. Order the k anchors as

$$0 = a_1 < \cdots < a_k < H,$$

set $a_{k+1} = H$, and let

$$r_j = a_{j+1} - a_j - 1$$

denote the j-th uninterrupted drafter run. Since the k runs contain all $H - k$ drafted blocks,

$$\sum_{j=1}^{k} r_j = H - k.$$

Therefore,

$$\max_j r_j \ \ge \ \left\lceil \frac{H-k}{k} \right\rceil.$$

Attainment. SpecAR places

$$a_j = \left\lfloor \frac{(j-1)H}{k} + \frac{1}{2} \right\rfloor, \qquad j = 1, \ldots, k.$$

For consecutive anchors,

$$a_{j+1} - a_j \in \left\{ \left\lfloor \frac{H}{k} \right\rfloor, \left\lceil \frac{H}{k} \right\rceil \right\},$$

so for $j < k$,

$$r_j \;\le\; \left\lceil \frac{H}{k} \right\rceil - 1 \;=\; \left\lceil \frac{H-k}{k} \right\rceil.$$

The same bound holds for the final run $r_k = H - a_k - 1$, since $a_k \ge (k-1)H/k - \frac{1}{2}$ gives $r_k \le H/k - \frac{1}{2} < \lceil H/k \rceil$. Hence

$$L_{\max}(\mathcal{A}_k) = \left\lceil \frac{H-k}{k} \right\rceil.$$

■

C Extended Results

C.1 Horizon and Hardware Controls

Table C1a reproduces SDVG's published configuration on its own rig and horizon, and Table C1b sweeps the horizon and the hardware.

Table C1a: Faithfulness reproduction of SDVG's published headline configuration ($\tau = -0.7$, the first row of its Table 1). Published cells (pub) are SDVG's Table 1 values (Hu & Zhang, 2026, Table 1): 1003 MovieGenVideoBench prompts, 2×A6000, $H = 9$, $\tau = -0.7$. Reproduction cells (repro) are our measurements on the same rig, horizon, and configuration, on the frozen evaluation split rather than a subset of their 1003 prompts. Pre-registered pass criterion: speedup within $\pm 0.15\times$ of published, accept within ± 5 points; the reproduction meets both (accept +0.4 points, speedup $-0.15\times$). The published draft-only endpoint (0.0644 VR, 25.7 s, 3.77×; Hu & Zhang, 2026, Table 1) was not part of the reproduction and is omitted. RTF and latency are the statistics defined in §4.1.

Arm	VR pub / repro	RTF pub / repro	Latency s (speedup) pub / repro	Accept % pub / repro	Scorer s/block pub / repro	Peak mem GB (repro)
Target-only	0.0788 / —	— / 0.0591	97.0 (1.00×) / 107.9 (1.00×)	—	—	39.91 / 16.52
SDVG τ=-0.7	0.0773 / —	— / 0.0952	60.9 (1.59×) / 75.0 (1.44×)	73.1 / 73.5	≈0.6 / 1.08	39.96 / 19.95

C.2 Example Generations

Figure 6 shows frames from four evaluation prompts, chosen by hand for visual variety, generated by target-only, SDVG with its threshold set for $k = 7$, and SpecAR at $k = 7$. The videos come from the main evaluation; the times shown are the Table 1 means for each method.

C.3 Paired Comparisons with SDVG

Table C3 pairs each SpecAR budget with an SDVG configuration under three matching rules and reports the paired VisionReward difference.

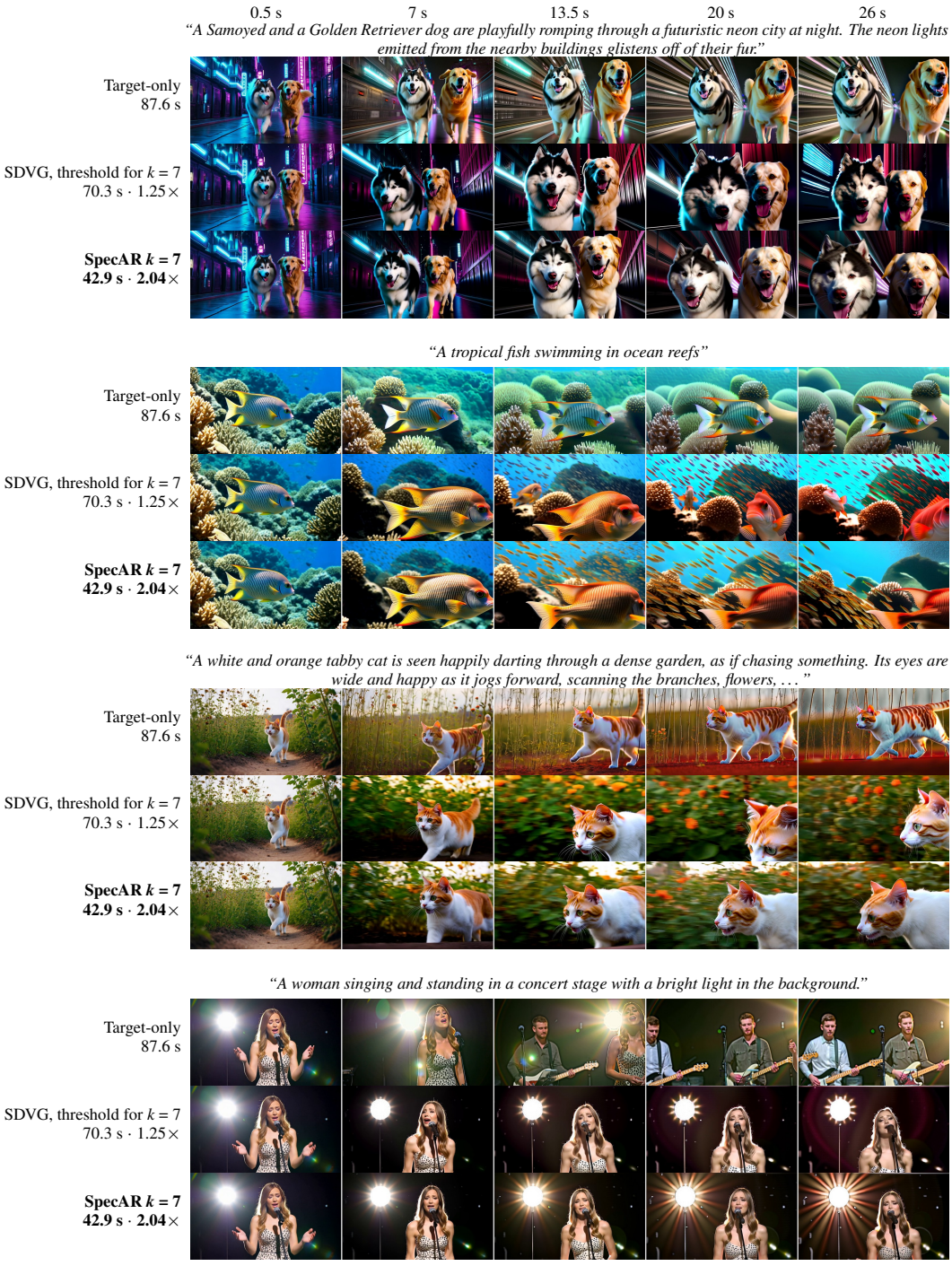


Figure 6: Example generations on Krea Realtime at $H = 36$ (26.6 s videos). Rows: target-only, SDVG with its threshold set for $k = 7$, and SpecAR $k = 7$, with the same prompt and seed in every row; columns are frames at 0.5, 7, 13.5, 20 and 26 s. Under each method is its mean generation time and speedup from Table 1 (one H100, lazy target KV caching, which leaves the generated frames unchanged).

C.4 Online Routing on Krea Realtime

These experiments precede the main evaluation and use SDVG's released ImageReward router on Krea Realtime. The router decides, for each drafted block, whether to keep it or regenerate it with the target; the first block is always regenerated.

The aggressive end of the published sweep leaves the router inactive. SDVG's paper sweeps the fixed threshold from $\tau = -0.7$, its headline configuration, to $\tau = -2.5$, its fastest operating point (2.09× at 88.9% acceptance on its own benchmark). At $\tau = -2.5$ on Krea Realtime at $H = 36$, the router rejected none of the drafted blocks. Every video's target share came only from the forced first block, so the measured draft share was 35/36 = 97.2% on every prompt. At $H = 18$, the draft share of 94.4% is the same arithmetic, 17/18.

Router-guided selection does not beat random selection. At the matched 75% rate, random acceptance scores 0.0015 higher than the router (paired $t = +0.75$, not significant). Repeating the comparison at five matched acceptance rates from 75% to 89% gives the same result at every rate

Table C1b: Runtime across hardware configurations and generation horizons. Video duration is fixed at 6.375 s for $H = 9$ and 26.625 s for $H = 36$, with identical durations across methods. RTF denotes mean per-video real-time factor; latency denotes mean generation time; speedup is computed from mean latencies (§4.1). On 2×A6000, SDVG's variable latency produces different rankings under mean RTF and mean latency: although SpecAR has lower mean latency at both horizons, SDVG has higher mean per-video RTF. VR here is the 1-frame-per-second VisionReward over the first 24 s used for this horizon and hardware sweep; every VR cell is an absolute score on that ruler, not a difference from target-only, so it differs from the whole-video VR of Table 1: on the same 1×H100, $H = 36$ videos, target-only scores 0.0987 here and 0.0914 there, SDVG $\tau = -0.7$ 0.1034 and 0.0954, and SpecAR $k = 9$ 0.1157 and 0.1005.

Rig	H	Arm	VR	RTF	Latency s (speedup)	Accept %	Scorer s/block	Peak mem GB
2×A6000	9	Target-only	0.0665	0.0591	107.9 (1.00×)	—	—	39.91 / 16.52
2×A6000	9	SDVG τ=-0.7	−0.0137	0.0952	75.0 (1.44×)	73.5	1.08	39.96 / 19.95
2×A6000	9	SpecAR k=3 (target-drive-budget matched, anchors 0/3/6)	−0.0198	0.0930	68.5 (1.57×)	66.7	—	39.91 / 16.52
2×A6000	36	Target-only	0.0747	0.0614	433.8 (1.00×)	—	—	39.94 / 16.52
2×A6000	36	SDVG τ=-0.7	−0.0050	0.0834	366.5 (1.18×)	49.7	1.07	39.99 / 19.95
2×A6000	36	SpecAR k=19 (target-drive-budget matched)	0.0049	0.0767	347.1 (1.25×)	47.2	—	39.94 / 16.52
1×H100	9	Target-only	0.0952	0.2550	26.9 (1.00×)	—	—	55.67
1×H100	9	SDVG τ=-0.7	0.0944	0.3379	19.5 (1.38×)	86.5	0.84	59.15
1×H100	9	SpecAR k=2 (target-drive-budget matched, anchors 0/5)	0.0898	0.4203	16.0 (1.68×)	77.8	—	55.67
1×H100	36	Target-only	0.0987	0.3039	87.6 (1.00×)	—	—	45.50
1×H100	36	SDVG τ=-0.7	0.1034	0.3422	87.0 (1.01×)	63.1	0.66	48.88
1×H100	36	SpecAR k=9 (accept-nearest)	0.1157	0.5708	46.7 (1.88×)	75.0	—	45.38

Table C3: Paired comparisons under three frozen rules. Equal acceptance: τ'_j selected for SpecAR $k = j$'s draft share. Nearest target-forward: pairs by target forwards. Nearest-RTF: SDVG's fastest configuration (τ'_1, RTF 0.471) is slower than SpecAR's slowest ($k = 9$, RTF 0.571), so no pair meets a matching tolerance and every SpecAR point is compared with τ'_1. Differences are paired over the same prompts and use the whole-video VisionReward of Table 1.

Matching rule (exact configs)	SDVG	SpecAR	ΔVR, SpecAR−SDVG
Equal acceptance: k=2 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.8111	+0.0021
Equal acceptance: k=3 vs τ'_2	τ'_2 · RTF 0.4658	RTF 0.7591	+0.0040
Equal acceptance: k=5 vs τ'_3	τ'_3 · RTF 0.4353	RTF 0.6827	+0.0071
Equal acceptance: k=7 vs τ'_4	τ'_4 · RTF 0.3986	RTF 0.6214	+0.0086
Equal acceptance: k=9 vs τ'_5	τ'_5 · RTF 0.3956	RTF 0.5708	+0.0129
Pooled, equal acceptance	—	—	+0.0069
Nearest-RTF: k=2 vs τ'_1	τ'_1 · RTF 0.4711 (SDVG's fastest)	RTF 0.8111	+0.0021
Nearest-RTF: k=3 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.7591	+0.0048
Nearest-RTF: k=5 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.6827	+0.0088
Nearest-RTF: k=7 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.6214	+0.0117
Nearest-RTF: k=9 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.5708	+0.0149
Pooled, nearest-RTF	—	—	+0.0085
Nearest target-forward: $\tau'2$ vs k=2	1.7 tgt / 35 draft	2.0 tgt / 34 draft	+0.0013
Nearest target-forward: $\tau'3$ vs k=3	2.9 tgt / 35 draft	3.0 tgt / 33 draft	+0.0032
Nearest target-forward: $\tau'4$ vs k=5	4.3 tgt / 35 draft	5.0 tgt / 31 draft	+0.0057
Nearest target-forward: $\tau'5$ vs k=7	6.1 tgt / 35 draft	7.0 tgt / 29 draft	+0.0097
Nearest target-forward: $\tau'5$ vs k=9	6.1 tgt / 35 draft	9.0 tgt / 27 draft	+0.0129
Pooled, forward-count-matched	—	—	+0.0065

(paired $|t| \leq 1.28$), with random acceptance scoring at least as high as the router at three of the five. Pooled over the four rates at which the two policies differ (at 89% both accept every drafted block), the router's paired advantage over random acceptance is -0.0011 VisionReward, with a 95% confidence interval of $[-0.0037, +0.0015]$ over 100 prompts. Per-block scoring therefore improves quality by at most 0.0015, about 2% of the router's mean score, and the point estimate favors random acceptance. Removing the forced first block lowers quality to 0.06719, so the quality in SDVG's router comes from forcing the first block to the target and from the acceptance rate, not from ImageReward's per-block scores. SDVG's own ablation agrees: forcing the first block and accepting at random gives 0.0771, against 0.0773 for the full router (Hu & Zhang, 2026, Table 2).

C.5 Placement Geometry and Trajectory Stabilization on Krea Realtime

Table C5 compares five placement policies at a strictly fixed budget of $k = 9$ target blocks ($H = 36$, 75% draft share) on Krea Realtime. All policies execute the exact same number of target and drafter forwards, running within 1% end-to-end latency (56.5–56.9 s, measured without lazy target KV caching). Quality is evaluated using the 1-fps VisionReward protocol over the first 24 s (as in Table C1b), where target-only generation scores 0.0987.

Balanced placement outperforms every alternative after Holm correction for the four comparisons (adjusted $p \leq 0.034$). $L_{\max}$ bounds the longest stretch the drafter runs without target correction; schedules that share a bound need not score identically. Every schedule whose bound reaches 27 (front-loaded, back-loaded and router top-k; 0.0966–0.1020) scores below every schedule whose bound stays at or below 13 (balanced and uniform random; 0.1080–0.1157). Quality degrades as anchors become more concentrated and the longest uninterrupted draft run $L_{\max}$ grows: front-loaded ($L_{\max} = 27$, 0.1020), router top-k ($L_{\max} \in [8, 27]$, 0.1001) and back-loaded ($L_{\max} = 27$, 0.0966) all trail the distributed schedules, whereas balanced SpecAR ($L_{\max} = 3$, 0.1157) outperforms both the concentrated and the random placements ($L_{\max} \in [7, 13]$, 0.1080) without stochastic variance or scoring overhead. Moreover, SpecAR exceeds target-only VisionReward (0.1157 versus 0.0987) because the 14B target exhibits progressive drift on 39% of prompts, where the Self-Forcing 1.3B drafter remains temporally stable (+0.0323 gain retained by SpecAR), while on the 55% of prompts where the 14B target is stronger, periodic target re-anchoring preserves target-only parity (-0.0032, not significant).

Table C4: Router-guided versus random acceptance on Krea Realtime at a matched 75% acceptance rate (SDVG threshold $\tau = -0.7$). VisionReward is paired across the same prompts; retention is relative to target-only generation.

Selection	First block forced to target	VisionReward ↑	Retention
ImageReward router (SDVG)	✓	0.07160	94.5%
Random acceptance	✓	0.07313	96.5%
Random acceptance	×	0.06719	88.7%

Table C5: Placement of nine target blocks on Krea Realtime ($H = 36$, 1×H100, same prompts and the same 75% draft share for every row). $L_{\max}$ is the longest uninterrupted draft run; for random and router-selected schedules it varies by prompt. VisionReward is the mean over prompts ± its 95% margin of error. Paired t statistics and two-sided p-values are for Balanced minus the row; Holm-adjusted p-values correct for the four comparisons.

Placement	$L_{\max}$	VisionReward ↑	Latency (s)	Paired t (vs. Balanced)	Two-sided p-value	Holm-adjusted p
Balanced (SpecAR)	**3**	**0.1157 ± 0.0061**	56.66	—	—	—
Uniform random	7–13	0.1080 ± 0.0064	56.55	+2.14	0.034	0.034
Front-loaded	27	0.1020 ± 0.0068	56.91	+2.45	0.015	0.030
Router top-k (SDVG scores)	8–27	0.1001 ± 0.0065	56.54	+2.61	0.010	0.030
Back-loaded	27	0.0966 ± 0.0070	56.76	+3.44	0.0008	0.0032

D MAGI-1: Target Budget and Generation Quality

We evaluate MAGI-1 24B as the target and MAGI-1 4.5B as the drafter. Under the evaluated settings, increasing target involvement reduces SpecAR's Quality-6 score. This experiment examines how target-budget selection depends on the relative quality of the two models.

D.1 Experimental Setup

We use the MAGI-1 24B and 4.5B base checkpoints without distillation or quantization. Both models use 32 denoising steps; the 4.5B checkpoint's default is 64. The workload consists of 150 image-to-video prompts initialized from VBench++ I2V first frames. Each video contains 36 one-second chunks at 24 fps and a resolution of 1280×720.

Experiments run on eight H200 GPUs with eight-way Ulysses context parallelism and no pipeline parallelism. Both models execute across all eight GPUs. KV caches remain on the GPUs, and the drafter's cache is reconstructed in the layout required by the target when switching models. End-to-end latency includes T5 text encoding.

We evaluate SpecAR at target budgets $k \in \{2, 5, 7\}$. SDVG thresholds are selected on held-out prompts to approximate the corresponding draft fractions; the realized fractions differ and are reported in Table D1. SpecAR speedups use target-only measurements on the same 150 prompts. SDVG speedups use the target-only reference from its separate evaluation run.

D.2 Quality and Latency

Draft-only generation completes in 856.4 seconds, compared with 2094.2 seconds for target-only generation. It scores 74.39 on Quality-6 and 0.0785 on VisionReward, compared with 70.74 and 0.0203 for target-only generation. SpecAR at $k = 2$ scores higher on Quality-6 (75.87) but lower on VisionReward (0.0634), and has the highest Quality-6 in Table D2.

Increasing SpecAR's target budget reduces Quality-6 from 75.87 at $k = 2$ to 74.93 at $k = 5$ and 72.17 at $k = 7$. Latency rises from 745.1 to 1114.0 seconds across the same budgets. SpecAR achieves 1.86-$2.81\times$ speedup over its target-only reference (Tables D1–D2).

These results establish that additional target computation does not necessarily improve generation quality.

Table D1: End-to-end quality and performance of SpecAR and SDVG on MAGI-1 ($8\times$H200), with $H = 36$; columns as in Table 1. Speedups are relative to target-only generation; the separate SDVG rerun uses its own target-only reference.

Model	Method	VR	RTF	Latency (s)	Speedup	Accept (%)	Mem (GB)
MAGI-1 24B MAGI-1 4.5B drafter $1280\times720 \cdot 36$ chunks $\cdot$ $8\times$H200	Target-only	0.0203	0.0172	2094.2	$1.00\times$	—	78.98
	Draft-only	0.0785	0.0420	856.4	$2.45\times$	—	72.30
	SDVG τ_1' (-1.5489)	0.0636	0.0280	1284.8	$1.63\times$	62	92.92
	SpecAR (k=2)	0.0634	0.0483	745.1	**2.81**$\times$	94	82.50
	SDVG τ_2' (-1.1082)	0.0560	0.0242	1485.3	$1.41\times$	75	92.92
	SpecAR (k=5)	0.0670	0.0383	940.4	**2.23**$\times$	86	88.70
	SDVG τ_3' (-1.0194)	0.0012	0.0193	1869.8	$1.12\times$	83	92.92
	SpecAR (k=7)	0.0143	0.0323	1114.0	**1.86**$\times$	81	89.83

Table D2: Generation quality on MAGI-1 across target-only, SDVG, and SpecAR operating points; columns as in Table 2.

Model	Method	Quality-6	Subject consist.	Background consist.	Motion smooth.	Dynamic degree	Aesthetic	Imaging	VisionReward
MAGI-1 24B MAGI-1 4.5B drafter	Target-only	70.74	71.75	82.84	98.82	100.00	45.57	53.02	0.0203
	Draft-only	74.39	78.47	85.15	98.36	100.00	50.94	58.29	0.0785
	SDVG τ_1' (-1.5489)	72.92	77.36	87.17	99.05	100.00	50.70	46.64	0.0636
	SpecAR (k=2)	75.87	81.75	88.58	98.68	100.00	52.97	54.83	0.0634
	SDVG τ_2' (-1.1082)	72.79	79.26	86.04	98.78	100.00	50.60	46.27	0.0560
	SpecAR (k=5)	74.93	79.88	87.96	98.71	100.00	53.86	51.69	0.0670
	SDVG τ_3' (-1.0194)	75.21	83.74	84.35	98.91	100.00	54.01	52.75	0.0012
	SpecAR (k=7)	72.17	79.47	83.89	98.70	100.00	46.48	49.92	0.0143

D.3 Quality Across the Rollout

We evaluate VisionReward separately on consecutive six-second segments. Target-only scores decline from 0.156 in the first segment to -0.045 in the final segment. From the second segment onward, target-only generation scores below SpecAR at $k = 2$, with differences of 0.056–0.085.

The quality decline is therefore visible within the rollout as well as in the aggregate scores. The 24B model also scored below the 4.5B model in a separate text-to-video check using MAGI-1's upstream inference script. The cause of this ordering remains unresolved, and the conclusions here apply to the evaluated configuration.

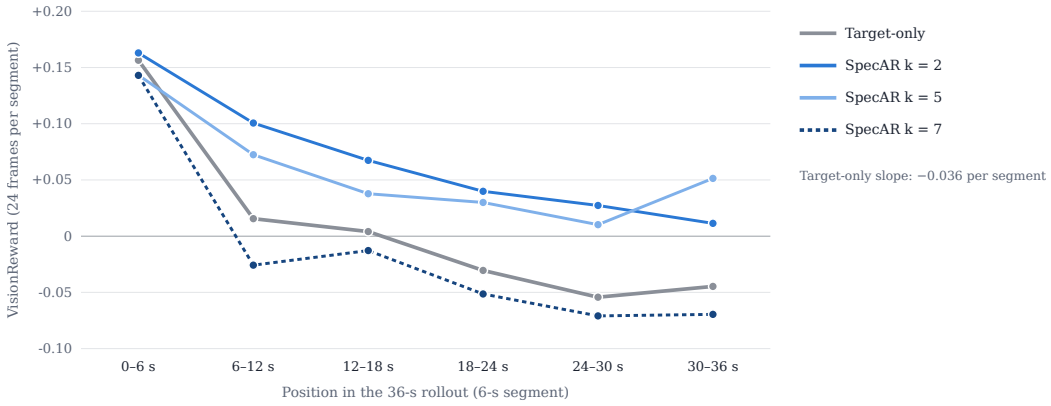


Figure 7: MAGI-1 VisionReward by 6-s segment of the 36-s rollout. Target-only falls along the rollout (slope -0.036 per segment); SpecAR $k = 2$ matches it in the first segment, where both follow the conditioning image, and stays above it in every later segment.

Table D3: Mean VisionReward per 6-s segment on MAGI-1 ($H = 36$), with the slope per segment.

Method	0–6 s	6–12 s	12–18 s	18–24 s	24–30 s	30–36 s	Slope per segment
Target-only	0.156	0.016	0.004	−0.031	−0.054	−0.045	**−0.036**
SpecAR (k=2)	0.163	0.101	0.067	0.040	0.027	0.011	−0.029
SpecAR (k=5)	0.143	0.072	0.038	0.030	0.010	0.051	−0.019
SpecAR (k=7)	0.143	−0.026	−0.013	−0.051	−0.071	−0.070	−0.035