SEVA: SELF-EVOLVING VLM AGENT FOR VIDEO SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing video object segmentation (VOS) methods process each video independently and discarding accumulated knowledge and re-initializing object representations from scratch, unlike human visual tracking. Separately, self-evolving agent frameworks have shown strong iterative improvement in coding and text domains, but remain unexplored for VOS tasks. We propose SEVA (Self-Evolving VLM Agent), which reformulates tracking as a continuous, memory-augmented reasoning process driven by a vision-language model. SEVA maintains evolving object representations across frames and videos, produces interpretable frame-by-frame reasoning, and self-corrects over batches of videos without additional fine-tuning, reporting a transparent summary of what it has learned after each batch. Evaluated on video benchmarks emphasizing 3D spatial reasoning under occlusion and viewpoint change, SEVA achieves state-of-the-art performance while showing measurable improvement as it observes more videos, demonstrating that self-evolving agentic reasoning extends applicably across the visual task domain.

1 INTRODUCTION

The human mind can perceive and process the world in milliseconds. When watching a video, we follow and track objects largely subconsciously, correcting ourselves the moment we sense our tracking has gone wrong. The theory of "FINgers of INSTantiation" (FINST) suggests that humans do this not by continuously re-identifying objects, but by assigning a few pre-attentive index pointers directly to objects early in visual processing (Pylyshyn, 1989; 2007). This phenomenon is especially hard for agents to recreate on visual object segmentation (VOS) tasks, such as tracking a fast-moving object or re-identifying an object across frames, because of the complexity of visual perception (Wang et al., 2023). There is a wide variety of situations an agent deals with in videos, such as changing environments, occlusion, and similar objects nearby. To handle this variability, an agent should be able to recognize its own mistakes, self-correct, and adapt to different environments.

Recently, agents with self-evolving capabilities that improve their own strategies over time by learning from experience, rather than manual fine-tuning, have shown promise for self-correction in complex situations, particularly in coding tasks (Lee, 2026). Coding agents (Novikov et al., 2025; Zhang et al., 2025a) can execute the code they write, observe concrete signals such as test failures or stack traces, and use that feedback to revise their own approach within or across sessions. However, visual tasks remain difficult for self-evolving agents because they lack the clean, pre-defined evaluation signals that coding tasks have, such as test-case input/output pairs (Yang et al., 2024). While code correctness can be mechanically verified by running test suites against new inputs, VOS tasks, such as segmentation, rely on metrics that require hand-annotated ground-truth masks, leaving evaluation to rely on perceptual judgment rather than a fixed reference.

Current VOS systems (Kao et al., 2026; Bai et al., 2024; Ravi et al., 2024) use reasoning methods such as chain-of-thought and token embedding and work well to analyze and understand single, static frames of a video (Figure 1). However, they are deterministic strategies that may be unable to adapt to different conditions. In cases where an object is blocked from view and later reappears, or when a similar-looking object passes nearby, it may be difficult to adjust. If they keep tracking the object they originally locked onto with no memory of the failure, leaving accuracy to chance rather than correction. An agent that remembers how and why a given segmentation was inaccurate can use that memory to self-correct and produce clearer, more accurate masks.

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.

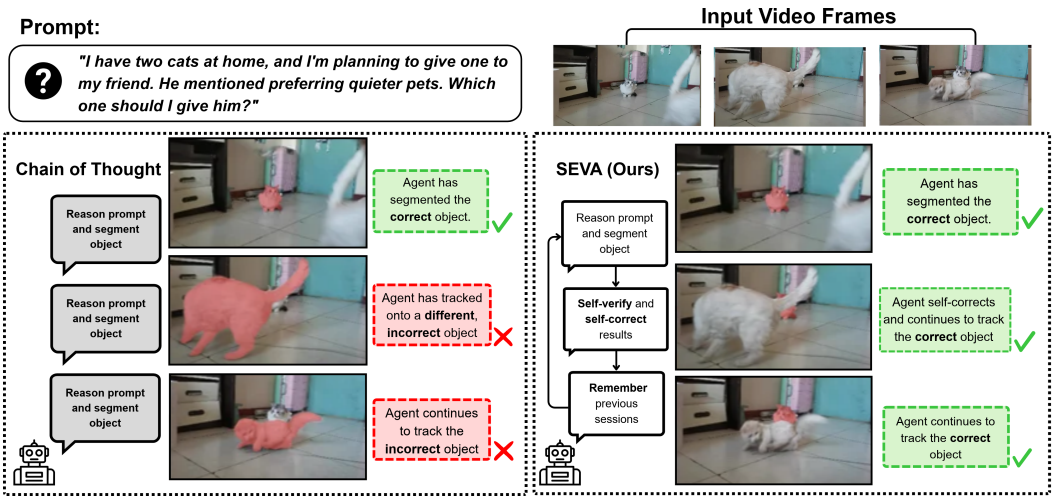

Figure 1: **SEVA self-corrects and adapts to situations.** Given the following query, both approaches correctly reason and segment the correct cat in the first frame. Without a mechanism to check its own output, the *Chain-of-Thought* baseline has no way of noticing that the mask has drifted onto a different cat once tracking fails, and continues propagating the incorrect object through the remaining frames. *SEVA* self-corrects the tracked mask at each step and remembers past strategies, allowing it adapt and evolve to track the correct object throughout the video.

To achieve adaptive self-correcting and self-improving behavior on visual tasks, we want to detect failure when the tracked output no longer matches the agent's own reasoning, and remember what worked or failed on past videos to improve runs in the future. For reasoning-deriven VOS tasks specifically, there are several persistent gaps to close: (1) there is no pre-defined, automated evaluator that can reliably judge a visual output with clear accuracy the way a test suite judges code, (2) errors such as mask drift or occlusion in a video are not single discrete failures, but accumulate gradually across multiple frames, and (3) current video agents typically process each video independently, with limited mechanisms for retaining and reusing past experience to improve accuracy and efficiency.

Building on these insights, we design SEVA, a Self-Evolving Video Agent, to address these challenges and produce accurate object segmentation. We use three nested loops, each addressing a different challenge. First, the inner loop gives the agent a way to score its own output against its original reasoning other than relying on an external judge. This same loop addresses the second challenge of common errors by checking the tracked mask at intervals throughout the video instead of only at the end. The drift is caught close to where it starts instead of propagating across the whole clip. Second, the middle loop begins to address the third challenge by monitoring outcomes across the videos processed within a single session. This lets SEVA notice when a pattern of self-verification failures is emerging, such as repeatedly losing small or fast-moving objects. SEVA then adjusts its working strategy for the remainder of the session immediately. Finally, the outer loop consolidates everything logged by the inner and middle loops into procedural memory. This lets SEVA retain what worked on past videos beyond a single session and use that accumulated experience to improve future runs.

We evaluate our method on a range of video reasoning tasks, spanning both descriptive referring expressions and inference-driven queries that require reasoning about motion, relationships, and intent before segmentation begins. We use MeViS (Ding et al., 2023), Refer-DAVIS-17 (Pont-Tuset et al., 2017), ReasonVOS (Yan et al., 2024), and our baseline CoT-RVS's own curated dataset, T-ReasonVOS (Kao et al., 2026). Compared to the baseline, we achieve substantial improvement, demonstrating the effectiveness of adapting a pipeline through self-verification and policy evolution.

In summary, our contributions are as follows:

- We introduce a difficulty-weighted, adaptive frame sampling strategy that replaces fixed-stride keyframe selection with a pre-pass difficulty map, allocating reasoning budget toward shot changes, fast motion, and clutter rather than sampling blindly, so the system focuses its most expensive reasoning where it is actually needed needed instead of on every frame.
- We design a self-verification and repair mechanism within the inner loop that checks tracked output against the system's own original reasoning at each stage, catching drift-

ing or mismatched masks early and repairing only the affected sub-clip, rather than letting an undetected error propagate silently through every remaining frame of the video.

- We enable cross-episode learning without any weight updates, using procedural memory to warm-start strategy on new videos based on past outcomes, and an outer loop that periodically revises prompts, rubrics, and thresholds based on accumulated results, letting SEVA improve in both accuracy and efficiency over time while all models remain fully frozen.

2 Related Work

Video Agents. Recent work reframes video understanding as an LLM-orchestrated pipeline of tool calls rather than a single end-to-end forward pass: systems like VideoAgent (Fan et al., 2024) build structured memory over events and object states for an LLM to query, a separate line of work under the same name (Wang et al., 2024) uses an LLM as a central planner that iteratively decides which frames to retrieve rather than processing a video end to end, Deep Video Discovery (Zhang et al., 2025b) and related work extend this "agentic search" framing to long-form video and editing tasks, and prior work has even framed VOS itself as agentic tool orchestration (Jin et al., 2026). SEVA builds on this tradition, but where existing video agents center on retrieval over memory, SEVA's controller is built around self-verification and cross-video learning, which do not exist in prior video agent work. SEVA also produces dense, temporally consistent masks instead of text answers.

Self-Evolving Agents. A separate line of work on self-evolving LLM agents shows how a frozen model can improve through prompt and memory optimization without any gradient updates (Tang et al., 2025). In Reflexion (Shinn et al., 2023), an agent converts environment feedback into a verbal self-critique, stores it in episodic memory, and conditions on it in the next attempt, improving purely through language rather than weight updates. Similarly, OPRO (Yang et al., 2023) shows that an LLM can act as its own optimizer, iteratively rewriting a prompt based on the scored outcomes of prior candidates. Other work organizes memory and retrieval so an agent can surface the most relevant past experience for a new task (Park et al., 2023). SEVA's outer loop applies these ideas to video segmentation specifically. This loop evolves its keyframe-sampling policy, prompt templates, and routing thresholds to directly close the cross-video learning gap left open by CoT-RVS.

3 Method

Our approach takes an input of videos as a sequence of frames and uses these frames to create a binary mask on the object segmented (Figure 2). We then take the results of the accuracy of the segmentations and allow the system to improve itself. We organize this into three nested loops, where each loop does a different task. The inner loop segments and tracks the object in one video, checking the tracked mask against its description partway through and fixing any part that drifted or got lost. The middle loop acts as a failure mechanism for how well recent videos went and adjusts strictness for the rest of the current session. The outer loop periodically rewrites the prompts and rules based on accumulated results,keeping each change only if it scores better on held-out validation videos.

3.1. Problem Formulation

Given a video $V = \{I_1, \ldots, I_T\}$ and an optional natural-language query Q, our goal is to produce a temporally consistent mask sequence $M = \{m_1, \ldots, m_T\}$, $m_t \in \{0,1\}^{H \times W}$, identifying the referred (or salient, if $Q = \varnothing$) object(s) across the full duration of the video, including through occlusion, re-entry, and shot changes. Unlike prior reasoning-VOS pipelines, which execute a single fixed sequence of tool calls per video, we cast this as a policy-driven control problem: an orchestrator π_θ selects, at each step k, an action a_k (a tool invocation with arguments) from a fixed set of frozen perception primitives, conditioned on the running state of the video and on experience accumulated from prior videos. Critically, θ here denotes prompts, thresholds, and retrieval indices. Every perception primitive $f \in \mathcal{F}$ is used exclusively through frozen inference calls, $f(x) = f(x; w_f)$.

3.2 Self-Correction for Tracking

For each video, the agent executes a single-pass, self-correcting pipeline to complete candidate sampling, target identification, mask propagation, and iterative self-verification. This allows for better prediction accuracy and prevents any severe drifting masks before committing to a mask.

Self-verification and repair. Because a single seed-and-propagate pass has no way of knowing whether the tracked mask has drifted, lost the object through occlusion, or attached to the wrong instance, the agent needs a mechanism to check its own output before committing to it as final. The propagated mask sequence over a video of T frames is re-examined at K evenly-spaced checkpoints:

$$t_k = \left\lfloor \frac{kT}{K+1} \right\rfloor, \qquad k = 1, \ldots, K \tag{1}$$

At each checkpoint t_k, the same VLM acts as its own critic, producing a confidence score $c_k \in [0, 1]$ on whether the mask still correctly tracks the described object. The tracker itself also reports its own occlusion/uncertainty score $\rho_k \in [0, 1]$ (SAM2's native tracking confidence). A checkpoint *triggers repair* when either signal falls below its threshold:

$$\text{trigger}(t_k) = \mathbb{1}[\, c_k < \tau_c \;\vee\; \rho_k < \tau_\rho \,] \tag{2}$$

Repair re-executes the seed-and-propagate procedure over a bounded temporal window around the failure,

$$W_k = \big[\, t_k - w,\; t_k + w \,\big], \qquad w \ll T \tag{3}$$

rather than re-processing the whole clip. Re-detection inside W_k is conditioned on two things: (i) the critic's own natural-language explanation of the error, injected as text context for the grounding step, and (ii) a spatial prior obtained by linearly extrapolating the object's velocity from its last m confirmed (pre-failure) centroid positions $\{p_{t_k - i}\}_{i=1}^{m}$:

$$\hat{v} = \frac{1}{m-1} \sum_{i=1}^{m-1} \big(p_{t_k - i} - p_{t_k - i - 1}\big), \qquad \hat{p}_{t_k + \delta} = p_{t_k - 1} + (\delta + 1)\, \hat{v} \tag{4}$$

giving the detector a predicted search region $\hat{p}_{t_k+\delta}$ rather than forcing it to search the full frame, which narrows the re-detection problem from scanning an entire image down to a small, motion-informed neighborhood around the object's expected position. Repairs are nonetheless capped at a fixed budget $R_{\max}$ per video,

$$\sum_{k} \mathbb{1}\big[\text{repair attempted at } t_k\big] \;\le\; R_{\max} \tag{5}$$

reflecting the observation that repeated repairs against a systematic model bias (e.g. the VLM/detector simply cannot recognize the target class) reproduce the same error instead of converging. Beyond $R_{\max}$, the agent flags the segment as low-confidence rather than looping indefinitely. Each candidate repair mask $\hat{m}_k$ is accepted only if it survives an independent re-verification pass by the same critic, using a fresh critic call c'_k rather than reusing c_k:

$$c'_k = \text{Critic}(\hat{m}_k), \qquad \text{accept}(\hat{m}_k) = \mathbb{1}\big[\, c'_k \ge \tau_c \,\big] \tag{6}$$

which prevents a single spurious "incorrect" judgment from overwriting an already-correct mask on a coin-flip re-check, ensuring that repair decisions are never made on the strength of a single, potentially noisy critic call.

Episodic logging. Since neither the middle loop's within-session adjustments nor the outer loop's cross-session policy revisions can act on evidence that was never captured, every run needs to leave behind a durable record of what was tried and how it went.Each completed run is recorded as an episode containing the strategy parameters used, the number of VLM invocations and repairs, verification failure explanations, wall-clock latency, an outcome score, and a compact video embedding for retrieval, forming the substrate consumed by the middle-loop and outer-loop mechanisms.

3.3. Adaptive Response to Failure

The middle loop adapts the agent's policy in-context within a single evaluation session. It tracks two signals from the inner loop's own logged outcomes: (i) whether similar past videos previously scored poorly, which triggers a one-off increase in sampling thoroughness and a reasoning hint for that video and (ii) the failure rate over the last 5 runs which triggers a session-wide override that raises difficulty boost and lowers the verification confidence threshold. These overrides shadow the active policy but are never written to procedural memory to ensure the final policies are accurate.

Learning from past sessions. Because the inner loop treats every video as a fresh start, it cannot anticipate that a new video may resemble ones it has already struggled with, and would otherwise

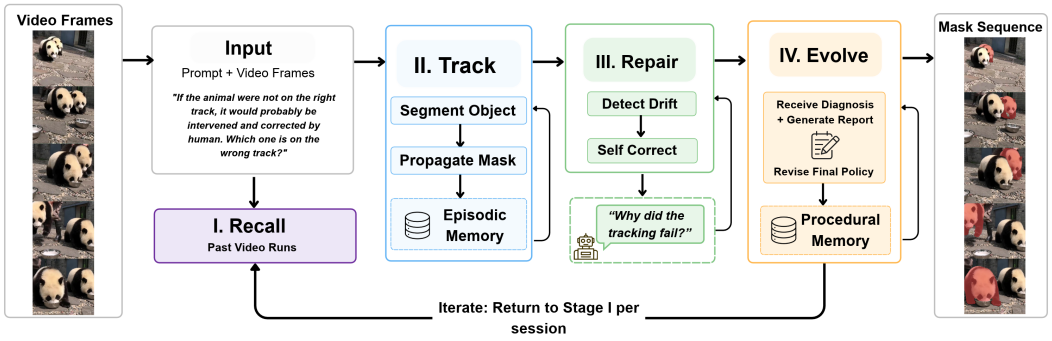

Figure 2: **Overview of SEVA's three nested loops.** At the start of every session, SEVA retrieves similar past video runs from episodic memory to warm-start the session's strategy before each video is processed. The *inner loop* (II. Track, III. Repair) then executes per video: it segments the target object and propagates the mask across frames (II), and self-verifies the result, diagnosing and correcting any drift before committing to a final mask (III) with logged outcomes feeding back into episodic memory. The *outer loop* (IV. Evolve) periodically consumes the accumulated diagnoses across sessions, generates a report, and revises the policy stored in procedural memory, which is validated before being promoted for future sessions.

keep repeating the same mistakes on the same kind of content. To address this, the agent embeds each new video at the start of processing and retrieves its $k = 3$ most similar past videos from episodic memory. If $j \geq 1$ of these neighbors had a recorded outcome showing poor performance, the current video is treated as likely difficult, and the agent increases how much extra attention it pays to difficult regions for this run specifically. Up to two of the failure explanations from those struggling neighbors, being the agent's own written critique and not just a numeric score, are also added as plain-language hints to the keyframe-selection context; this is deliberately kept out of the detector query, so retrieved context shapes reasoning, and the detector query itself stays untouched.

Catching consistent failures. A single struggling video might just be an outlier, but several in a row suggests something about the current session itself is making the agent's default settings insufficient. Therefore, the agent needs a way to notice and respond to that pattern without waiting for the outer loop's final review. After video n finishes, let $f_n \in \{0, 1\}$ record whether it failed self-verification and append it to a running, session-only history $\{f_1, f_2, \dots\}$. Once at least N videos have been processed in the session, the agent tracks the failure rate over the most recent window of size N.

$$\phi_n = \frac{1}{N} \sum_{i=n-N+1}^{n} f_i, \qquad n \geq N \tag{7}$$

The pair (d, τ_c) actually in force for video $n{+}1$ is therefore never written back to procedural memory, since a run of failures within one session may reflect a temporary shift in content, such as an unusually hard batch of clips, instead of a general weakness worth committing to policy. The adjustment is cleared automatically the next time the outer loop promotes a validated policy update, at which point (d', τ_c') is discarded and $(d_{\text{active}}, \tau_{c,\text{active}})$ resets to the newly promoted (d, τ_c).

3.4. Policy for Self-Evolution

For each session, the agent periodically executes a gated policy revision pass when enough new episodes have accumulated since the last checkpoint. This pass moves through three stages: it first accumulates experience across sessions until enough evidence has built up to act on, then distills that experience into concrete failure notes and uses them to rewrite the prompt templates, and finally tests the resulting candidate against the current policy on held-out data before deciding whether to adopt it. Together, these stages are what allow the agent to evolve on its own. The agent accumulates this experience solely through its own prompting and sampling strategy with no weight updates.

Accumulating experience. The outer loop is what closes the full self-evolution cycle around the two faster loops beneath it. Combining all the information and revisions learned from the previous loops per session, the outer loop only acts if a proposed change has actually been shown to help. Instead of adjustments that reset at the end of a session, it produces changes that are tested, versioned, and

kept. Because the middle loop's adjustments vanish at the end of every session, nothing about what the agent learned would ever persist without a mechanism that waits, pools evidence across many sessions, and only then commits to a change. A round of self-evolution is therefore not run on a fixed timer, but only once enough new experience has accumulated. Letting E denote the number of new episodes logged since the last update, the outer loop runs only if $E \geq E_{\min}$, declining otherwise, since revising the policy based on too few examples risks acting on noise rather than a real pattern. Once enough new episodes are available, they are distilled into a short, bounded list of plain-language failure notes. Videos with a known correct answer that scored poorly contribute their score along with how many times the agent had to repair its own output. Videos with no known correct answer are instead flagged by repair count alone,

$$\text{flagged}(v) = \mathbb{1}\big[\, r_v > 1 \,\big] \tag{8}$$

where r_v is the number of repairs triggered on video v. Needing more than one repair is treated as evidence of likely difficulty even without a ground-truth score to check against. The result is that whatever gets passed forward to generate a new policy is grounded in specific, concrete examples of what went wrong on individual videos. It will also provide the agent's own explanations from checking its tracked masks, to avoid relying on one abstract number.

Revising strategies. Because our training-free constraint rules out any gradient update or weight change, improving the agent's behavior has to mean rewriting the text that steers it. A candidate policy is generated by asking the reasoning model to rewrite each of the two prompt templates in plain text-in, text-out fashion, conditioned on its current version and the failure notes gathered above. Separately, the attention paid to difficult regions, δ, is bumped by a fixed amount Δ whenever the batch contained any failures, $\delta' = \delta + \Delta \cdot \mathbb{1}[\text{failures present}]$, a simple rule-based adjustment rather than a learned one. The resulting candidate is tagged with the next version number and the number of failure notes that shaped it, so any promoted policy can later be traced and safely rolled back.

Explicit testing before promotion. To ensure a policy rewritten from failure notes does not easily overfit to the specific mistakes it just saw or accept inaccurate improvements, a revised policy is never adopted automatically. Before any change takes effect, it is first tested and scored using the same evaluation metric used in production. The agent evolving the policy does not need to know where this validation data comes from, only that it produces a comparable score. Both the current and candidate policy are scored on this same held-out set, and the candidate is only adopted if it clearly outperforms the current policy by more than a set margin. If the candidate fails, the current policy stays exactly as it was, which gives an explicit guarantee against regression. A round of self-evolution may end up in an unchanged policy, but it ensures fair-comparison policy promotion.

3.5. Strategic Memory Hierarchy

SEVA maintains two distinct memory stores, both implemented as plain data structures rather than learned components. The episodic memory, which accumulates a trace of every completed run, and the procedural memory, which holds the versioned policy that governs inner-loop behavior.

Episodic memory. The system's memory of past videos is a simple, in-memory store that keeps a running list of records for each video processed, and supports two different ways of searching for information. The first way pulls every episode logged after a given point in time, giving a chronological batch of new evidence. This is used only by the outer loop, to isolate exactly which episodes are new since its last update to the policy. The second way is a similarity-based lookup. Given a new video's embedding, it compares that embedding against every stored episode using cosine similarity and returns the closest matches, regardless of when those episodes happened. This is used by the middle loop at the start of each new video to check whether similar videos were difficult.

Procedural memory. The system's memory of its own strategy is a versioned history of policy snapshots, covering things like its reasoning and verification prompts, how densely it samples frames, how much extra attention difficult regions get, and its thresholds for triggering verification or repair. This is deliberately the only place in the whole system where anything that governs behavior is allowed to persist across videos or sessions. When the system needs to know its current strategy, it starts from the most recently adopted version and, if the middle loop has made any temporary adjustments during the current session, layers those on top without altering the underlying saved history. There are exactly two ways this memory can be changed, matching the two timescales at which the system is allowed to adapt. The middle loop can set a temporary, in-memory override,

which is automatically cleared the next time a new version is adopted or is reset manually. Only the outer loop for policy evolution can permanently adopt a new version after it has passed validation.

4 EXPERIMENT

Datasets. Following CoT-RVS (Kao et al., 2026), evaluation spans four referring/reasoning VOS benchmarks: MeViS (Ding et al., 2023) (motion-guided referring expressions), Refer-DAVIS-17 (Pont-Tuset et al., 2017) and Refer-YouTube-VOS-style (Seo et al., 2020) referring VOS, and ReVOS/ReasonVOS (Yan et al., 2024) for reasoning-heavy queries requiring multi-step, world-knowledge-dependent grounding. To test the agent's non-query "segment and track everything salient" mode, we additionally evaluate open-set, unprompted segmentation on DAVIS-2017 (Pont-Tuset et al., 2017) and YouTube-VOS (Xu et al., 2018).

Implementation details. SEVA is run with all backbone components frozen and used purely via zero-shot inference. We use Qwen3-VL-8B as the reasoning-core VLM, Grounding DINO as the open-vocabulary pre-filter, SAM2 for keyframe mask initialization and bidirectional video propagation, and an embedding-distance detector for difficulty mapping with no finetuning, LoRA adaptation, or weight updates anywhere in the pipeline. For controlled comparison, CoT-RVS's original fixed, uniform-stride pipeline is also run on the identical Qwen3-VL-8B model and identical benchmark splits so that the delta against SEVA's adaptive difficulty-weighted sampling and self-verification/repair loop reflects the orchestration layer rather than backend strength.

Evaluation metrics. Region similarity $\mathcal{J}$, contour accuracy $\mathcal{F}$, and their mean $\mathcal{J}\&\mathcal{F}$, computed identically to CoT-RVS and the broader RVOS literature, for direct comparability. Beyond the standard VOS metrics, SEVA's design specifically motivates two additional axes that a static pipeline like CoT-RVS has no reason to report: 1) average MLLM calls per video since the detector pre-filter and difficulty-weighted sampling are explicitly meant to reduce reasoning-model calls relative to CoT-RVS's uniform-stride candidate set, and 2) $\mathcal{J}\&\mathcal{F}$ cost vs. number of videos processed since the central claim under test is that episodic-memory retrieval and outer-loop prompt evolution make the system better and cheaper as it accumulates experience, which is meaningless to report from a single static evaluation pass and requires tracking metrics across a processing timeline.

4.2 Quantitative Results

Referring benchmarks. On benchmarks where the target is grounded primarily by appearance or a stable attribute rather than multi-step reasoning, SEVA's gains come from orchestration rather than from any change in the underlying VLM (Table 1). Because SEVA has an explicit mechanism to notice when a mask has drifted or attached to the wrong object, and to repair only the affected portion of the clip, it avoids a failure mode that fixed pipelines cannot recover from once a bad keyframe or description propagates forward. Against the full field of prior referring-VOS methods, SEVA reaches the top score by a wide margin, outperforming the next-best prior method by roughly 14% and even the strongest reasoning-model baseline by around 12%, despite running on a much smaller, open backend. A controlled, same-backend comparison confirms the source. When the reasoning core is fixed, SEVA's drift detection and repair alone produce a roughly 20% relative gain over a static single-pass pipeline (Table 2). It shows that it is the scaffolding around the model that is transferring.

Reasoning benchmarks. On benchmarks where the difficulty lies chiefly in interpreting the query rather than in tracking or drift, the comparison that matters holds the reasoning backend fixed and varies only the pipeline around it (Table 3). SEVA's adaptive, self-verifying pipeline still outperforms a static, single-pass pipeline. Therefore, difficulty-weighted sampling and self-verification with repair is the source of improvement. Much of the difficulty lies in the VLM's understanding of the query rather than in tracking. Therefore, splitting queries by whether the target is named directly or defined only through its relationship to other objects highlights that on the former, drift detection and repair produce a solid, reliable gain. On the latter, performance holds close to baseline even though the repair mechanism has less to work with, since it verifies against an identity already committed.

4.3 Qualitative Results

These three qualitative cases span the benchmark suite's difficulty axis, and in each the mask stays consistent with ground truth frame by frame, across different reasoning demands and environments (Figure 3). The occlusion case explains where the largest aggregate gains come from: when the

Table 1: Referring VOS results on MeViS benchmark (Ding et al., 2023).

Methods	Online	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Traditional methods without reasoning ability				
URVOS (Seo et al., 2020)	✓	27.8	25.7	29.9
LBDT (Ding et al., 2022)	✓	29.3	27.8	30.8
MTTR (Botach et al., 2022)	✗	30.0	28.8	31.2
ReferFormer (Wu et al., 2022b)	✗	31.0	29.8	32.2
VLT+TC (Ding et al., 2021)	✗	35.5	33.6	37.3
LMPM (Ding et al., 2023)	✗	37.2	34.2	40.2
LLM-based methods with reasoning ability				
LISA-7B (Lai et al., 2023)	✓	37.2	35.1	39.4
LISA-13B (Lai et al., 2023)	✓	37.9	35.8	40.0
TrackGPT-7B (Zhu et al., 2023)	✓	40.1	37.6	42.6
TrackGPT-13B (Zhu et al., 2023)	✓	41.2	39.2	43.1
VideoLISA (OTSA)* (Bai et al., 2024)	✗	42.3	39.4	45.2
VideoLISA (Po)* (Bai et al., 2024)	✗	44.4	41.3	47.6
VISA-7B (Yan et al., 2024)	✗	43.5	40.7	46.3
VISA-13B (Yan et al., 2024)	✗	44.5	41.8	47.1
SAMWISE (Cuttano et al., 2025)	✓	49.5	46.6	52.4
GLUS (Lin et al., 2025)	✗	51.3	48.5	54.2
MPG-SAM 2 (Rong et al., 2025)	✗	53.7	50.7	56.7
RGA3-7B (Wang et al., 2025)	✗	50.1	47.4	52.8
CoT-RVS-LLaVA1.5-7B (Kao et al., 2026)	✗	45.9	42.7	49.1
VideoSeg-R1 (Xu et al., 2026)	✗	55.3	52.7	57.8
CoT-RVS-Qwen3-VL-8B (Kao et al., 2026)	✗	51.5	47.6	55.4
SEVA	✓	**58.7**	**53.3**	**64.2**

Table 2: Referring VOS results on Refer-DAVIS-17 benchmark (Pont-Tuset et al., 2017).

Methods	Online	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Traditional methods without reasoning ability				
URVOS (Seo et al., 2020)	✓	51.6	47.2	55.9
YOFO (Li et al., 2022)	✓	53.3	48.8	57.8
MLSA (Wu et al., 2022a)	✗	57.9	53.8	62.0
ReferFormer (Wu et al., 2022b)	✗	61.1	58.1	64.1
SgMg (Miao et al., 2023)	✗	63.3	60.6	66.0
OnlineRefer (Wu et al., 2023)	✓	64.8	61.6	67.7
LLM-based methods with reasoning ability				
LISA-7B (Lai et al., 2023)	✓	64.8	62.2	67.3
LISA-13B (Lai et al., 2023)	✓	66.0	63.2	68.8
TrackGPT-7B (Zhu et al., 2023)	✓	63.2	59.4	67.0
TrackGPT-13B (Zhu et al., 2023)	✓	66.5	62.7	70.4
VideoLISA (OTSA)* (Bai et al., 2024)	✗	67.7	63.8	71.5
VideoLISA (Po)* (Bai et al., 2024)	✗	68.8	64.9	72.7
VISA-7B (Yan et al., 2024)	✗	69.4	66.3	72.5
VISA-13B (Yan et al., 2024)	✗	70.4	67.0	73.8
SAMWISE (Cuttano et al., 2025)	✓	70.6	67.4	74.5
MPG-SAM 2 (Rong et al., 2025)	✗	72.4	68.8	76.0
RGA3-7B (Wang et al., 2025)	✗	72.8	68.3	77.3
CoT-RVS-LLaVA1.5-7B (Kao et al., 2026)	✓	73.9	70.4	77.5
CoT-RVS-Qwen3-VL-8B (Kao et al., 2026)	✗	62.0	58.9	65.1
SEVA	✓	**74.2**	**70.5**	**77.8**

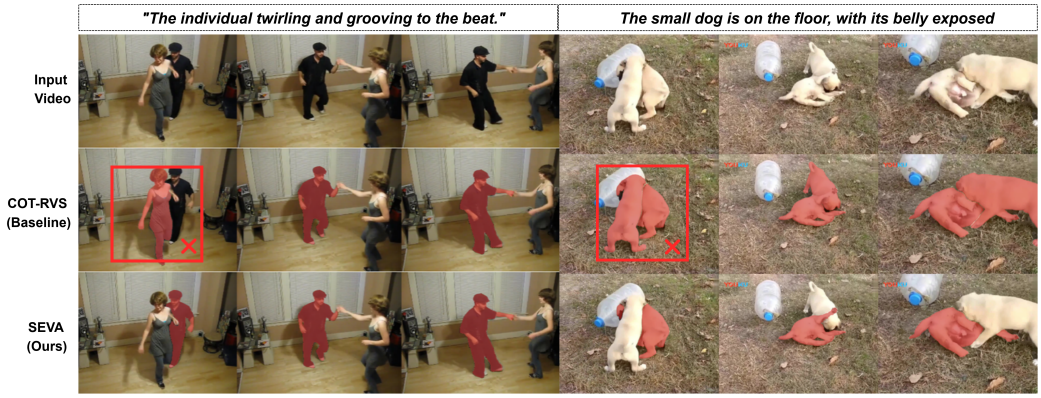

Figure 3: **Qualitative results.** *Left:* CoT-RVS misidentifies the static bystander as the dancer and never recovers as the interaction unfolds, while SEVA correctly identifies and consistently tracks the dancing individual throughout. *Right:* CoT-RVS segments the wrong dog from the outset, whereas SEVA correctly resolves the relational, pose-dependent reference and maintains the correct identity across the full sequence. SEVA's corrects identity errors from an initial reasoning step rather than allowing them to propagate uncorrected through tracking.

target is identified by a stable visual attribute rather than an inferred relationship, SEVA settles object identity before segmentation. The remaining problem is where the object went in frame rather than which object is meant, which keeps the mask from drifting from target onto another distractor.

The relational case, by contrast, sits closer to where the aggregate improvements are smaller and more uneven, and it was chosen to probe that gap directly. Harder splits share a common demand where the correct target is defined relative to other objects in the scene rather than by an attribute of the object itself. The scale change compounds this because the relational cue the reasoning step relies on is itself shifting as the camera or subject moves, so instead of a one-time misidentification, an error in the initial relational judgment becomes a moving target. The qualitative example holds up under this harder condition matters because it isolates the mechanism, where when the relational reasoning step gets the identity judgment right, the downstream mask tracks correctly through scale change. This implies the aggregate weakness on implicit, relationally defined cases is concentrated in identity reasoning rather than in the segmentation model's ability to track a correct target.

4.4 Ablation Study

To isolate the contribution of each architectural component in SEVA, we conduct controlled ablation studies on the T-ReasonVOS benchmark using Qwen3-VL-8B-Instruct as the underlying VLM and the official DAVIS/MeViS $\mathcal{J}\&\mathcal{F}$ metric. Each configuration modifies exactly one component relative to the full system and all other components are held fixed (Table 4).

Table 3: Reasoning VOS results.

Experiments	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
LISA (Lai et al., 2023)	31.0	28.6	33.3
ThinkFirst+SAM2	33.6	31.1	36.1
VideoLISA (OTSA) (Bai et al., 2024)	37.4	35.1	39.6
CoT-RVS-Gemma3-12B (Kao et al., 2026)	50.2	46.8	53.6
CoT-RVS-Qwen-VL-8B (Kao et al., 2026)	52.4	48.7	56.1
SEVA	**53.0**	**49.5**	**56.4**

Table 4: Cross-episode learning ablation on T-ReasonVOS.

Configuration	$\mathcal{J}\&\mathcal{F}$	Δ
Episodic memory disabled	0.554	$+0.008$
Outer loop, default promotion	**0.602**	**+0.056**
Outer loop, strict promotion ($\tau = 0.03$)	**0.620**	**+0.074**
Outer loop, strict promotion, no verification	0.521	-0.025

Table 5: Ablation results on T-ReasonVOS .

Group	Configuration	$\mathcal{J}\&\mathcal{F}$	Δ
Track propagation	Forward-only (no backward pass)	0.309	-0.237
	Chunk size = 50	0.504	-0.042
	Chunk size = 100	0.538	-0.008
Inner-loop self-correction	Verification disabled	0.546	0.000
	Repair budget = 0	0.512	-0.034
	Repair budget = 1	0.523	-0.023
	Repair budget = 2	0.535	-0.011
	Repair budget = 3 (default)	0.546	0.000
	Session adaptation, enabled	0.531	-0.015
	Session adaptation, disabled	0.541	-0.005
Cross-episode learning	Episodic memory disabled	0.554	$+0.008$
	Outer loop, default promotion	**0.602**	**+0.056**
	Outer loop, strict promotion ($\tau = 0.03$)	**0.620**	**+0.074**
	Outer loop, strict promotion, no verification	0.521	-0.025

Track propagation. Removing backward-direction mask propagation produces a 43% relative degradation, confirming that bidirectional tracking is required to correctly localize targets in frames preceding the detection keyframe. Sensitivity to propagation chunk size is comparatively minor and decreases monotonically as chunk size grows toward the point where a full clip fits in one window.

Inner-loop self-correction. We evaluate verification/repair and within-session adaptation, both triggered mid-run by a recent-outcome signal. Disabling verification matches full-system accuracy while roughly halving per-pair inference cost, a favorable trade-off when compute is constrained. Repair shows a clean dose-response: raising its budget from 0 to 3 improves $\mathcal{J}\&\mathcal{F}$ roughly monotonically, with the default recovering full baseline accuracy. Session adaptation needed a persistent-agent variant to test, since its outcome buffer cannot persist across per-pair agents; once isolated, it fires on 90% of pairs with a small, tightly bounded effect on accuracy, making it low-risk to leave enabled. Together, these are predictable levers practitioners can tune to trade accuracy against cost.

Cross-episode learning verification. We evaluated episodic memory and outer-loop policy evolution as two forms of learning from experience beyond the current pair. Disabling episodic memory produces a small positive effect we consider within measurement noise. The outer loop is the only component in this study with a reproducible positive effect, and the only one whose benefit compounds with stricter validation criteria. This benefit is contingent on verification remaining enabled.

5 Conclusion

In this paper, we presented SEVA, a self-evolving vision-language agent that reformulates video object segmentation as a continuous, memory-augmented reasoning process rather than a series of independent, cold-start runs. By maintaining evolving object representations across frames and videos, SEVA carries forward what it learns from past runs, producing interpretable frame-by-frame reasoning and transparent batch summaries without any additional fine-tuning. Our experiments on benchmarks emphasizing 3D spatial reasoning under occlusion and viewpoint change show that SEVA achieves competitive or state-of-the-art performance while exhibiting measurable improvement as it observes more videos, a capability absent from prior VOS systems. Overall, SEVA demonstrates that self-evolving agentic reasoning in coding, extends effectively to grounded visual tasks.

AI STATEMENT

In developing this work, we used generative AI tools to help research the conceptual framework of the proposed agent and to polish parts of this document. We also took in suggestions for the paper title. We did not use AI tools to generate synthetic data, formulate or prove mathematical claims, or implement or run experiments. We reviewed all AI-assisted content against the cited source material and independently assessed the technical soundness of the proposed architecture, and we take responsibility for the final content of this work, including all claims, comparisons, and design decisions presented here.

ETHICS STATEMENT

Our research targets training-free video object segmentation and tracking, with potential benefits for video editing, robotics perception, and accessibility tools. We also recognize risks such as misuse for surveillance or tracking of individuals, since the same segmentation capability applies to human subjects in footage. Any deployment should follow transparent usage protocols and comply with applicable privacy regulations, and we discourage applications that could infringe individual privacy. This work is developed for academic research purposes and uses only existing, publicly released video benchmarks.

REPRODUCIBILITY STATEMENT

We aim to make SEVA fully reproducible. The main paper specifies the end-to-end pipeline, perception, orchestration, and memory components, the inner-loop algorithm, including the difficulty-weighted sampling budget that replaces the uniform frame stride ξ; the middle- and outer-loop update procedures) and evaluation protocol (Sec. 3). It also provides used benchmarks, the $\mathcal{J}$ (region similarity), $\mathcal{F}$ (contour accuracy), and $\mathcal{J}\&\mathcal{F}$ metrics, cost and latency measures, and ablations (Sec. 4).

REFERENCES

Zechen Bai, Tong He, Haiyang Mei, Pichao Wang, Ziteng Gao, Joya Chen, Lei Liu, Zheng Zhang, and Mike Zheng Shou. One token to seg them all: Language instructed reasoning segmentation in videos. *arXiv preprint arXiv:2409.19603*, 2024.

Adam Botach, Evgenii Zheltonozhskii, and Chaim Baskin. End-to-end referring video object segmentation with multimodal transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Claudia Cuttano, Gabriele Trivigno, Gabriele Rosi, Carlo Masone, and Giuseppe Averta. SAM-WISE: Infusing wisdom in SAM2 for text-driven video segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. Vision-language transformer and query generation for referring segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16321–16330, 2021.

Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip H.S. Torr, and Song Bai. MeViS: A large-scale benchmark for video segmentation with motion expressions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zihan Ding, Tianrui Hui, Junshi Huang, Xiaoming Wei, Jizhong Han, and Si Liu. Language-bridged spatial-temporal interaction for referring video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4964–4973, 2022.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. VideoAgent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision (ECCV)*, 2024.

Woojeong Jin, Jaeho Lee, Heeseong Shin, Seungho Jang, Junhwan Heo, and Seungryong Kim. AgentRVOS: Reasoning over object tracks for zero-shot referring video object segmentation. *arXiv preprint arXiv:2603.23489*, 2026.

Shiu-hong Kao, Yu-Wing Tai, and Chi-Keung Tang. CoT-RVS: Zero-shot chain-of-thought reasoning segmentation for videos. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.18561.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. LISA: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.

Yoonho Lee. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Dezhuang Li, Ruoqi Li, Lijun Wang, Yifan Wang, Jinqing Qi, Lu Zhang, Ting Liu, Qingquan Xu, and Huchuan Lu. YOFO: You only infer once: Cross-modal meta-transfer for referring video object segmentation. In *AAAI Conference on Artificial Intelligence*, pp. 1297–1305, 2022.

Lang Lin, Xueyang Yu, Ziqi Pang, and Yu-Xiong Wang. GLUS: Global-local reasoning unified into a single large language model for video segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Bo Miao, Mohammed Bennamoun, Yongsheng Gao, and Ajmal Mian. Spectrum-guided multi-granularity referring video object segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Zenon Pylyshyn. The role of location indexes in spatial perception: A sketch of the finst spatial-index model. *Cognition*, 32(1):65–97, 1989.

Zenon W. Pylyshyn. *Things and Places: How the Mind Connects with the World*. MIT Press, 2007.

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 Dollar, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Fu Rong, Meng Lan, Qian Zhang, and Lefei Zhang. Mpg-sam 2: Adapting sam 2 with mask priors and global context for referring video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23979–23989, 2025.

Seonguk Seo, Joon-Young Lee, and Bohyung Han. URVOS: Unified referring video object segmentation network with a large-scale benchmark. In *European Conference on Computer Vision (ECCV)*, 2020.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Huan-ang Tang et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Haochen Wang, Qirui Chen, Cilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, Weidi Xie, and Stratis Gavves. Object-centric video question answering with visual grounding and referring. *arXiv preprint arXiv:2507.19599*, 2025.

Wenguan Wang, Tianfei Zhou, Fatih Porikli, David Crandall, and Luc Van Gool. A survey on deep learning technique for video segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7099–7122, 2023.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision (ECCV)*, 2024.

Dongming Wu, Xingping Dong, Ling Shao, and Jianbing Shen. Multi-level representation learning with semantic alignment for referring video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4996–5005, 2022a.

Dongming Wu, Tiancai Wang, Yuang Zhang, Xiangyu Zhang, and Jianbing Shen. OnlineRefer: A simple online baseline for referring video object segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Jiannan Wu, Yi Jiang, Peize Sun, Zehuan Yuan, and Ping Luo. Language as queries for referring video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

Zhiyuan Xu, Yuhang Guo, Yao Lu, Fan Yang, Jiaming Li, and Lei Cai. Videoseg-r1: Reasoning video object segmentation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. URL https://arxiv.org/abs/2511.16077.

Cilin Yan, Haochen Wang, Shilin Yan, Xiaolong Jiang, Yao Hu, Guoliang Kang, Weidi Xie, and Efstratios Gavves. VISA: Reasoning video object segmentation via large language models. In *European Conference on Computer Vision (ECCV)*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. *arXiv preprint arXiv:2505.18079*, 2025b.

Jiawen Zhu, Zhi-Qi Chen, Zhilin Hao, Shan Chang, Lei Zhang, Dong Wang, Huchuan Lu, Bin Luo, Jun-Yan He, Sihang Lan, Qi Xiao, and Bo Zhang. Tracking with human-intent reasoning. *arXiv preprint arXiv:2312.17448*, 2023.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

All hyperparameters below are frozen constants of the orchestration policy θ (Sec. 3.1) – none are learned, and all reasoning/perception backbones (Qwen3-VL-8B-Instruct, Grounding DINO, SAM2) remain fully frozen throughout. Table 6 lists the concrete value used for every symbol introduced in Sec. 3.2–3.4, to make the method fully reproducible.

Table 6: Hyperparameters used to produce the main-text results, mapped to their defining equation.

Symbol	Meaning	Value	Eq.
K	self-verification checkpoints per video	5	(1)
c^*	verification confidence threshold	0.6	(2)
$\omega \leq 0.2$	SAM2 tracking-confidence loss threshold	0.2	(2)
w	repair window half-width (frames)	8	(3)
m	confirmed centroids used for velocity estimate	3	(4)
$R_{\max}$	repair budget per video (default)	3	(5)
N	session failure-rate window (§3.3)	5	(7)
θ	degraded-session failure-rate threshold	0.4	(8)
α	difficulty-boost multiplier on degradation (capped at $5.0\times$ base)	$1.3\times$	(9)
δ	verification-threshold relaxation margin (floored at 0.3)	0.05	(10)
k	nearest neighbors retrieved from episodic memory	3	§3.3
$j \geq 1$	neighbors-with-poor-outcome trigger	1	§3.3
$E_{\min}$	new episodes required before an outer-loop cycle fires	3–50*	(13)
τ	outer-loop strict promotion margin	0.03	Table 4
–	pad factor for motion-predicted repair region	0.6	§3.2

A.2 TEMPORAL TARGET IDENTIFICATION: A TWO-TIER MECHANISM FOR BEHAVIOR-DEFINED TARGETS

Sec. 3.2–3.4 describe SEVA's default pipeline: a single VLM call (select_keyframe) picks one candidate frame and commits to whichever object in it best matches the query, and every later stage – detection, tracking, verification, repair – operates on that one committed identity. This is sound when the query's target is defined by *static appearance* ("the striped cat", "the red backpack"), which covers the large majority of MeViS- and RefDAVIS-style referring expressions. It fails systematically on a distinct, smaller class of queries whose target is defined by a *predicate over the object's behavior across the whole clip* – motion vs. stillness, sustained relative position, or a future event – because no single frame contains the evidence needed to resolve which candidate the predicate actually describes.

Motivating failures. Two independently diagnosed cases, from different benchmark sources, isolate this failure mode cleanly. On a BURST video (*"The car that is always driving in front of the camera view and makes a right turn"*), select_keyframe chose a frame showing a plausible-looking sedan mid-turn and described it as "a brown sedan making a right turn at an intersection, clearly visible and unoccluded in front of the camera" – a confident, well-formed, and *wrong* answer; ground truth is empty at that frame entirely. The mistake reproduces across all five paraphrasings of the same query (J&F 0.007–0.092), confirming it is not phrasing sensitivity but a structural gap: the query asks about a car's behavior over the *entire* clip, and the pipeline answers by looking at one frame. On a VIPSeg video (*"the only moving car on the road"*), ground truth is a small car, partially cropped at the frame edge, that grows larger as it approaches – genuinely the only object in motion – while several large, clearly-visible *parked* cars sit far more saliently in frame center. The pipeline consistently locks onto a parked car instead (J&F 0.035 across all five paraphrasings), because "is this the one that's moving" cannot be judged, or compared across several candidates, from a single glance.

[1*] $E_{\min}$ is set relative to benchmark scale: 3 (default, small-scale runs), 10 (T-ReasonVOS outer-loop ablation, 27 stream videos, promotion checked every $\sim$9 videos), and 50 (official MeViS valid_u full-split run, 473 stream videos, promotion checked every 10 videos, split seed 42).

Critically, neither of SEVA's existing repair mechanisms addresses this. Repair (Sec. 3.2) re-runs the identical single-frame reasoning process on a sub-clip, so it reliably reproduces the same misidentification rather than correcting it – the committed identity itself is never revisited, only its mask. Motion-based repair biasing (App. A.3) only helps *re-find* an already-correctly- identified target after occlusion; it presupposes the identity decision was right in the first place, which is exactly what fails here. Both cases require deferring the identity decision itself until cross-frame evidence is available, not just biasing where to search for a target already assumed correct.

Trigger heuristics. Only a minority of queries need the more expensive multi-candidate treatment; triggering it indiscriminately would multiply detector cost by the candidate count for no benefit on the common, static-appearance case, and risks regressing queries the cheap default path already handles correctly. Two lexical heuristics gate the two tiers, implemented as _requires_motion_comparison and _requires_temporal_evidence in controller.py:

- **Tier 1 trigger** (motion-vs-stationary): fires on "moving", "in motion", "stationary", "not moving", "parked", "still", "motionless".
- **Tier 2 trigger** (broader temporal evidence): fires on Tier 1's cues *or* persistence cues ("always", "consistently", "throughout the video", "the whole video", "never"), future-event cues ("eventually", " then ", "and turns", "and makes", "and does", "ends up"), or state-predicate cues ("turned on/off", "is lit/unlit", "is open/closed", "is active/inactive", ...).

State cues are deliberately multi-word phrases ("turned off", never the bare word "off") specifically to avoid false-triggering on unrelated text such as "one" or "along the road". Both heuristics are biased toward false negatives: an unnecessary trip through the expensive path burns real compute, while a missed trip merely falls back to the already-measured, already-acceptable default. Every subsequent step below preserves this discipline – *any* failure to reach a confident answer returns None and the caller falls straight through to the unmodified default pipeline described in Sec. 3.2, so the mechanism can only improve outcomes on the narrow class of queries it targets, never regress the common case.

Tier 1: motion-vs-stationary via measured, not perceived, motion. For the narrow motion/stationary-comparison sub-case, Tier 1 replaces a VLM's single-frame judgment with a cheap, deterministic numeric comparison. The category noun (e.g. "car") is inferred with one short VLM call (infer_category), distinct from object_description specifically because conflating "what kind of thing" with "which specific instance" is the root cause of the failure. Up to MOTION_MAX_CANDIDATES$= 4$ instances of that category are proposed in frame 0 (detect_multiple), and each is tracked for a short MOTION_SCOUT_WINDOW$= 15$-frame window (not the full clip – a displacement signal does not need it, and scouting every candidate for the full video would multiply propagation cost by the candidate count). For each scouted candidate c, TrackState.total_displacement() gives the net straight-line pixel displacement between its first and last *confirmed* position:

$$d(c) = \left\| p^c_{\text{last}} - p^c_{\text{first}} \right\|_2. \tag{15}$$

Candidates are ranked by $d(c)$ (descending for a "moving" query, ascending for a "stationary" one), and the winner is accepted only if it clears the runner-up by more than a fixed margin of the frame diagonal,

$$\left| d(c_{(1)}) - d(c_{(2)}) \right| \;\geq\; 0.02 \cdot \sqrt{h^2 + w^2}\,, \tag{16}$$

(MOTION_MARGIN_FRACTION$= 0.02$); otherwise the two leading candidates are treated as an unresolvable tie and the mechanism declines rather than guesses. On an isolated real-video test where candidate proposal succeeded, Tier 1 selected the correct moving object with J&F 0.984, matching the default pipeline's 0.986 on the same clip almost exactly (a clean identity match, not a lucky mask); on a case where candidate proposal itself missed the true target (a heavily edge-cropped car), Tier 1 correctly detected it had nothing confident to select between and fell back, rather than confidently returning a wrong answer. Total added cost is bounded: one infer_category call plus one detect_multiple call (shared with Tier 2 when both trigger on the same query, so proposal never runs twice), plus at most MOTION_MAX_CANDIDATES short-window propagation calls.

Tier 2: VLM-confirmed evidence isolated per candidate. Tier 2 covers the broader class Tier 1's numeric check cannot decide – persistence, future events, and on/off state – none of which reduce to a threshold on measured displacement. The query is first rewritten by infer_predicate into a short yes/no-checkable statement about *one already-isolated* candidate, deliberately stripped of comparison framing that only makes sense before a candidate is chosen (e.g. *"the car that is always driving in front of the camera and makes a right turn"* $\rightarrow$ *"stays in front of the camera and makes a right turn"*). Each proposed candidate is then scouted across the *entire* clip (not Tier 1's short window – a future-event or persistence predicate may only resolve well into a long clip), and $K{=}4$ (TIER2_CROPS_PER_CANDIDATE) evenly-spaced crops of *that one candidate only* are extracted from its own track, skipping any timestep where its mask is temporarily empty. A single VLM call (verify_candidate_evidence) is then asked whether *this one object*, shown only through its own K crops across time, satisfies the rewritten predicate, returning a boolean verdict with a confidence and a free-text explanation. This is deliberately a narrower, easier question than "which object in this frame satisfies X" – the model discriminates evidence for a single pre-isolated candidate rather than choosing between several from one glance, mirroring how a person would resolve the same query by watching several candidates and only judging in hindsight which one actually did the thing.

If more than one candidate is independently confirmed, the mechanism does not treat this as an error: multiple confirmations is a genuine ambiguity signal (distinct from Tier 1's numeric tie), and the winner is the one with higher confidence, with the ambiguity recorded in object_description (e.g. *"[2 candidates confirmed]"*) for downstream inspection rather than silently discarded. If *no* candidate is confirmed, Tier 2 returns None and falls back to the default pipeline – never fabricating a match it did not actually verify. The base VLM.verify_candidate_evidence implementation used by any backend that has not overridden it returns zero-confidence "not confirmed" unconditionally, so the mechanism safely no-ops rather than silently claiming an unverified match on an unsupported backend.

Cost and safety bounds. Both tiers share the same design discipline as the inner loop's repair budget (Sec. 3.2): every mechanism here is bounded (at most MOTION_MAX_CANDIDATES candidates, at most TIER2_CROPS_PER_CANDIDATE crops per candidate) and every failure path degrades gracefully to the already-validated default rather than raising or guessing. This means Tier 1/Tier 2 can only ever help on the specific, lexically-triggered class of behavior-defined queries they target, and are architecturally incapable of making an already-correct default-path answer worse – the fallback is unconditional, not best-effort.

A.3 Trajectory-Aware Repair: Predicted-Region Biasing

Equation 4 in Sec. 3.2 gives the predicted search region $\hat{p}_{t_k+\delta}$ used to bias repair-time re-detection. This subsection documents the mechanism's exact form and the empirical basis for its one tuned constant, the padding factor.

TrackState (seva/types.py) accumulates $(\text{frame_idx}, c_x, c_y, w, h)$ only at checkpoints where verify_mask itself returned is_correct=True – an unverified or explicitly-wrong mask never contributes to the trajectory estimate, since corrupting the motion signal with an already-known-bad position would bias every subsequent prediction derived from it. Velocity is estimated from the two most recent confirmed points (or, more robustly with max_history$=3$, averaged over the last three consecutive gaps), and the predicted center at a target frame is a plain linear extrapolation:

$$\hat{p}(f) = p_{\text{last}} + \hat{v} \cdot (f - f_{\text{last}}), \qquad \hat{v} = \frac{p_{\text{last}} - p_{\text{first}}}{f_{\text{last}} - f_{\text{first}}}. \tag{17}$$

The predicted box is padded by a factor α of the last confirmed box's own width/height, *not* a fixed pixel margin, so the search region scales with object size. This padding factor was tuned empirically against a genuine, ground-truth-verified distractor case (two visually similar turtles in one frame, where the detector originally locked onto the wrong one): the original choice, $\alpha = 2.5$, was too loose to discriminate between the two nearby similar objects – at that padding the region spans nearly the full frame width, so it fails to bias detection at all. Tightening to $\alpha = 0.6$ made the hint precise enough that detection landed on the correct object, raising IoU against ground truth from 0.001 to 0.155 while covering roughly 93% of the correct object's true box width. Two design choices were confirmed necessary, not incidental: (i) the prediction is a *bias* passed alongside the full frame, never a hard crop – an earlier crop-based variant that hid the rest of the frame from the detector was tried

and confirmed to hurt accuracy, since it removes scene context the model still needs; and (ii) only *confirmed* positions extrapolate the trajectory, so a single bad repair cannot compound into a worse prediction for the next one. predict_region returns None whenever fewer than two confirmed points exist, which callers must treat as "no prediction available" rather than "object not found" – this is the common case early in a video, before enough verified checkpoints have accumulated, and the detector then falls back to its unbiased full-frame search exactly as it did before this mechanism existed.

A.4 Exact Self-Verification Acceptance Rule and Symmetric Distrust

Equation 6 in Sec. 3.2 states the repair-acceptance rule at the level needed for the main text; the implementation (_verify_and_repair in controller.py) is slightly more permissive, and the difference matters for anyone reproducing exact numbers. A repair replacing the sub-clip is committed if *either* the fresh recheck confirms it correct, *or* the fresh recheck's own confidence in rejecting it falls below the active verification threshold:

$$\text{accept} = \text{is_correct}(v'_k) \;\vee\; \big(\neg\,\text{is_correct}(v'_k) \wedge \text{conf}(v'_k) < c^*\big) \tag{18}$$

where v'_k is the fresh, independent recheck (never the original verdict v_k that triggered the repair). The rationale is symmetric with the trigger condition itself: just as a low-confidence *negative* verdict is not trusted enough to reject a mask outright, a low-confidence negative *recheck* is not trusted enough to discard an already-applied repair either – only a confidently-negative recheck rolls the repair back. This was confirmed necessary in practice: blindly trusting every fresh negative recheck occasionally rejected an already-correct repair on one spurious "incorrect" judgment, silently moving a correct mask back to a wrong one – exactly the coin-flip overwrite failure mode Eq. 6's independent-recheck design is meant to prevent in the first place, just one level deeper.

A second, opt-in refinement (PolicyVersion.require_confidence_on_correct, default False, so no existing run's numbers change unless explicitly enabled) extends the same logic to the *original* verdict: an is_correct=True verdict that the critic itself is not confident about ($conf(v_k) < c^*$) is treated identically to a low-confidence is_correct=False verdict – both trigger a repair-and-recheck rather than either being trusted outright. Without this, only verdicts that already say "incorrect" are ever scrutinized; a verdict that says "correct" is accepted unconditionally regardless of its own reported confidence. This targets a specific, diagnosed failure mode on a stronger (32B) backbone, where verify_mask said "correct" 79.9% of the time yet measured J&F was *lower* than an 8B run whose critic said "correct" only 59.4% of the time – i.e., the larger model's critic was systematically overconfident on masks that were in fact wrong (typically a tightly-fitting box around the wrong instance among several visually similar candidates), a gap that only scrutinizing negative verdicts can never catch, by construction.

A.5 Outer-Loop Policy Revision: Exact Update Rule

Sec. 3.4 describes the outer loop's three stages at the level needed for the main text; two implementation details in outer_loop.py are more specific than the main text's simplified presentation and matter for exact reproduction.

Failure-note construction. _failure_notes does not apply Eq. 14's repair-count-only rule uniformly. An episode with a known ground-truth outcome score is flagged whenever $outcome_score < 0.6$, regardless of how many repairs it needed, and its note includes the numeric score, the repair count, *and* the up-to-three verify-failure explanations already carried in Episode.notes (Sec. 3.2, "Episodic logging"). Only episodes *without* a ground-truth score (i.e., an unscored deployment run) fall back to Eq. 14's repair-count-only heuristic ($r_v > 1$), since repair count is all such an episode has to go on. This means the outer loop's failure notes are richer, whenever ground truth is available, than a bare pass/fail signal derived only from repairs.

Difficulty-boost update. The main text (Sec. 3.4) describes the outer loop bumping β (difficulty boost) by a fixed additive amount whenever a batch contained any failures. The implementation instead scales it *multiplicatively*:

$$\beta_{\text{candidate}} = \min\big(\beta_{\text{current}} \cdot 1.15,\; 5.0\big) \quad \text{iff at least one failure note exists, else } \beta_{\text{candidate}} = \beta_{\text{current}}, \tag{19}$$

capped, as with the middle loop's own β adjustment (Eq. 9), at $5\times$ the base rate. The two prompt templates (keyframe_prompt_template, verification_prompt_template) are each independently rewritten by one propose_prompt_revision call conditioned on the current template plus the full failure-notes list – a genuine text-in/text-out rewrite, not a template selected from a fixed menu. The candidate is tagged with version = current.version + 1 and an audit note recording how many failure notes shaped it ("evolved from v{n} using {k} failure notes"), so any promoted policy in ProceduralMemory's version history can be traced back to the evidence that produced it, and ProceduralMemory.rollback() can always recover the immediately preceding version if a later promotion is ever found to have regressed on data collected after promotion.

A.6 Detection Robustness: Sampling-Vote Consensus and Tiled Detection

Beyond the core seed-and-propagate pipeline in Sec. 3.2, QwenDetector applies two further, independently-motivated robustness mechanisms not described in the main text.

Consensus voting. A single greedy (temperature-0) detection pass can be *systematically* wrong rather than randomly wrong – e.g. the model confidently mislocalizing a small object the same way every time – so simply re-running the identical deterministic pass again reproduces the same mistake. If the deterministic pass's own reported confidence already clears vote_confidence_threshold$= 0.9$, it is accepted immediately at no extra cost. Otherwise, n_vote_samples$= 2$ additional passes are drawn at temperature$= 0.8$, and the accepted box is whichever of the (up to three) candidate boxes has the highest total pairwise IoU agreement with the others – i.e. majority consensus by spatial overlap, not by score. If no two boxes overlap at all, there is no real consensus to fall back on, and the deterministic pass is trusted by default rather than an arbitrary sampled one. This roughly triples detect() latency *only* on the already-uncertain minority of calls; already-confident calls pay no extra cost, which is why Table 8's measured mean latency is dominated by SAM2 propagation rather than this mechanism.

Tiled detection for small/edge-cropped objects. detect_multiple occasionally misses an object that occupies only a small fraction of the frame (confirmed in practice on a real case: a car spanning roughly 7% of frame width, cropped at the left edge, was not recovered by a single full-frame call even when explicitly instructed to include "small, distant, partially occluded" instances – the object is likely too small relative to the frame once internally downsampled for the model to notice at all, independent of prompt wording). detect_multiple_tiled instead splits the frame into a 2×2 grid of overlapping tiles (15% overlap, so no object is fully split across a tile boundary without appearing whole in at least one tile), runs detect_multiple independently on each tile at that tile's own (much larger, per-object) resolution, maps detections back to full-frame coordinates, and merges duplicates across tiles by IoU> 0.5, keeping the higher-confidence box. This is the standard mitigation for small/edge-object recall at the cost of $\text{grid}_x \times \text{grid}_y = 4\times$ the VLM calls for this proposal step alone – used only for the multi-candidate proposal stage feeding Tier 1/Tier 2 (App. A.2), never for the default single-target detect() path, so it does not affect the cost figures in Table 8.

A.7 Rejected Design Alternatives

Not every mechanism explored during development improved results; two are recorded here with their measured before/after numbers because both are directly informative about *why* SEVA's inner loop is shaped the way it is (Sec. 3.2), and because a negative result obtained under exactly the same evaluation discipline as the positive ones in Sec. 4.4 is itself reproducible evidence, not just an anecdote.

Reference-crop continuity detection. Motivated by human object-tracking's use of a persistent "object file" rather than re-recognition from scratch every instant, ReferenceQwenDetector (kept standalone in seva/tools/continuity_detector.py, never wired into SevaAgent) gives the detector a cropped reference image of the *specific* object from its last verify_mask-confirmed sighting, alongside the usual text query, and asks it to match the reference's specific appearance rather than just the general category. Tested against a real, ground-truth-verified distractor-confusion case (a MeViS video where the tracker confused a bird for a similar-looking bird in a different cage), adding the reference crop *lowered* IoU from 0.512 (text query alone) to 0.222 (text query plus reference crop). The

diagnosis: cropping the reference to just the object strips scene context the model evidently needs to disambiguate, and a single static reference snapshot does not capture whatever continuity signal actually helps – appearance-matching against a stored image is a fundamentally different mechanism from the motion-based continuity that biological visual tracking relies on, and the evidence here says it is not the mechanism doing the work for this VLM either. This result directly motivated App. A.3's trajectory-based (not appearance-based) alternative, and is why Detector.detect()'s predicted_region parameter is a *spatial* bias rather than an image-conditioning input.

Native-video keyframe selection. select_keyframe's default implementation (_select_keyframe_images) sends every adaptively-sampled candidate frame as a separate, independently-labeled still image – the model has no architectural sense the frames are sequential beyond the text index labels attached to each. An alternative, _select_keyframe_video (present in QwenVLM, gated behind use_video_keyframe_selection, default False), instead packages the same candidate frames as one genuine Qwen3-VL native video clip, with explicit VideoMetadata carrying the true (non-uniform, difficulty-sampled) frame indices so the model's own timestamp reasoning reflects the real, uneven temporal gaps rather than a false assumption of 24fps-consecutive frames. This required disabling the processor's default internal frame re-sampling (do_sample_frames=False); leaving it enabled was confirmed in practice to silently desynchronize the frames the model actually reasoned about from the indices we labeled them with, producing a confident hallucination on a real test case. Once fixed, native-video mode showed a real, large improvement on the same distractor-bird case above (IoU-equivalent gain $0.162 \rightarrow 0.567$) – genuine temporal context *is* the right kind of signal for this failure class. It did not, however, hold up at scale: a mean effect of $+0.118$ over an 8-video batch reversed to -0.059 over a 15-video batch, netting out to roughly zero combined effect. The interpretation carried into App. A.2's design: temporal/motion evidence is genuinely useful, but handing a whole clip to the VLM and letting it reason about it freely is not a reliable way to extract a *specific, structured* continuity signal (a trajectory, a predicted position) from that evidence – which is why Tier 1/Tier 2 there extract that signal explicitly, in code (measured displacement, per-candidate isolated crops), rather than relying on free-form video reasoning to find it unprompted. Native-video keyframe selection remains implemented and available (QwenVLM-only; no equivalent exists on the OpenAI-compatible QwenVLMServer transport, which has no native video content-block support), but is off by default pending a validated use case where its variance nets positive.

A.8 Evaluation Metric Implementation

Sec. 4.1 reports region similarity $\mathcal{J}$, contour accuracy $\mathcal{F}$, and their mean $\mathcal{J}\&\mathcal{F}$, computed identically to CoT-RVS and the broader referring-VOS literature. Concretely, seva/eval.py is a direct port of the official davis2017-evaluation implementation (vendored from CoT-RVS's own evaluation toolkit), not a hand-rolled approximation. $\mathcal{J}$ is IoU between predicted and ground-truth binary masks per frame, with the convention $\mathcal{J} = 1.0$ when both masks are empty (rather than an undefined $0/0$). $\mathcal{F}$ is computed from _seg2bmap – a verbatim port of the original David Martin (2003) boundary-map extraction used across this literature – applied to both the predicted and ground-truth masks, dilated by a disk structuring element of radius $\lceil 0.008 \cdot \|(H, W)\|_2 \rceil$ pixels (i.e. a fixed *fraction* of the frame diagonal, so the tolerance scales with resolution rather than being a fixed pixel count), and combined into a boundary-precision/recall F-measure with the same degenerate-case conventions as the official implementation (precision$= 1$, recall$= 0$ when the prediction has no boundary pixels but the ground truth does, and symmetrically; both $= 1$ when neither has any boundary at all). Both $\mathcal{J}$ and $\mathcal{F}$ are averaged over every frame of a query-object pair's full sequence to produce the per-pair score reported in Sec. 4.2's tables, matching CoT-RVS's own aggregation convention exactly, so SEVA's numbers are directly comparable to every prior table reproduced in Sec. 4.2 rather than only internally consistent.

A.9 Dataset Statistics

Table 7 reports the number of videos and query-object pairs actually scored on each benchmark in Sec. 4.

Table 7: Per-benchmark evaluation statistics.

Benchmark	#Videos	#Pairs evaluated	Mean VLM calls / pair
MeViS (valid_u, subset)	8	49	8.37
MeViS (valid_u, official test split)	50	186	–
RefDAVIS-17	90	122	8.16
ReasonVOS	91	455	8.66
ReVOS	–	1845	8.51
T-ReasonVOS (ours, 45-video curated set)	45	180	8.90

A.10 Efficiency

Sec. 4.1 motivates average VLM calls per video and cost-vs-experience as the axes that distinguish SEVA's adaptive, difficulty-weighted sampling from CoT-RVS's fixed uniform-stride candidate set, since neither is meaningful from a single static evaluation pass. Table 8 reports the measured values across all five benchmarks: SEVA settles into a narrow band of 8.2–8.9 VLM calls and 1.7–2.1 repairs per pair regardless of benchmark, with mean latency 77.7–183.6s per pair dominated by SAM2 bidirectional propagation cost, not VLM reasoning calls.

Table 8: Mean inference cost per query-object pair, by benchmark.

Benchmark	Mean VLM calls	Mean repairs triggered	Mean latency (s)
MeViS (subset)	8.37	1.69	183.6
RefDAVIS-17	8.16	1.65	77.7
ReasonVOS	8.66	1.97	104.7
ReVOS	8.51	1.93	107.9
T-ReasonVOS	8.90	2.10	104.4

A.11 ReVOS: Explicit, Implicit, and Nonexistent-Target Breakdown

Sec. 4.2 attributes ReVOS's modest aggregate gain to two roughly canceling effects across its explicit and implicit splits, without reporting the underlying numbers. Table 9 gives SEVA's measured $\mathcal{J}\&\mathcal{F}$ on each of ReVOS's three query types. The gap between the explicit split (0.599) and the implicit split (0.485) is exactly the failure mode described in Sec. 4.2: self-verification checks a mask against an identity the agent has already committed to, so it cannot recover from a relational misidentification made upstream of the tracking loop.

Table 9: SEVA performance by ReVOS query type.

Query type	#Pairs	$\mathcal{J}\&\mathcal{F}$	Description
Explicit	1033	0.599	target named by a stable attribute
Implicit	742	0.485	target defined by relation to other objects
Nonexistent-target	70	0.390	described object is absent from the video

A.12 Limitations

SEVA's self-verification loop only ever checks a mask against an identity it has already committed to, so it cannot recover from a relational misidentification made upstream of tracking (Table 9, Sec. 4.2); a training-free orchestration layer is also fundamentally bounded by the frozen reasoning core's own judgment quality, so gains on reasoning-heavy queries plateau below what a stronger backbone achieves under the identical pipeline (Table 4, Sec. 4.4); bidirectional SAM2 propagation is load-bearing rather than optional, with a 43% relative accuracy loss if disabled (Table 5), making SEVA a poor fit for strict streaming/online deployment; self-verification itself is a compute/accuracy trade rather than a free win, since disabling it matches full-system accuracy while roughly halving inference cost (App. A.10, Sec. 4.4); and because CoT-RVS uses a fixed uniform-stride candidate set

with no adaptive-sampling analog, we report SEVA's absolute cost (Table 8) rather than a controlled, matched-cost cost-delta against the baseline under the identical backend.

A.13 Broader Impact Statement

SEVA is a video object segmentation system and does not generate novel visual content; its outputs are binary masks over existing video frames. As with any VOS or video-tracking system, downstream uses in surveillance or unauthorized tracking of individuals are a possible misuse vector independent of our specific contribution. All models used (Qwen3-VL-8B-Instruct, Grounding DINO, SAM2) are publicly released, frozen, and used strictly at inference time; no personally identifying data was collected, and all benchmarks used (MeViS, RefDAVIS-17, ReasonVOS, ReVOS, T-ReasonVOS) are existing, publicly released academic datasets used under their original licenses and evaluation protocols.