VSAS-BENCH: Real-Time Evaluation of Visual Streaming Assistant Models

Supplementary Material

A. Dataset Preprocessing and Annotations

As detailed in Sec. 3.1, our benchmark comprises 44 videos sourced from existing datasets and 48 newly recorded videos. For all newly recorded content, we apply strict anonymization to remove personally identifiable information. Specifically, we run a face detector on every frame and blur all detected faces, using a threshold tuned for high recall to minimize missed detections. We additionally blur all visible vehicle license plates. No further personal data from human subjects is collected.

We generate dense annotations using GPT-5, applying the sliding-window with uniform-tail (SW+U) memory policy described in Sec. 4.1 with a buffer size of 42. Annotations are produced at 1 FPS, yielding free-form descriptions for every second of video. All auto-generated annotations are subsequently human-verified, with particular attention to maintaining consistency across adjacent frames, especially when the underlying scene remains unchanged. Additional examples with full prompts are provided in Fig. 7. Notably, our prompt design explicitly specifies the task granularity to minimize ambiguity and ensure that models respond at the appropriate level of detail.

B. Detailed Comparison with Recent Streaming Benchmarks

In Tab. 7, we report dataset statistics in comparison to two recently introduced video benchmarks. Our benchmark is of comparable scale to prior benchmarks when evaluated under realistic streaming settings. Although it contains fewer source videos, it provides richer and denser annotations, with a median duration of 1.0s between annotations and an average of 11.8 unique events per video, compared to 61.8s and 2.5 unique events in RTV-Bench. This dense multi-timestamp QA annotation (MTQA) is critical for evaluating temporal fidelity and latency effects in streaming models. Moreover, streaming models must be evaluated on contiguous, overlapping video segments to reflect real-world usage; when evaluated in this mode, our benchmark incurs a comparable single-GPU evaluation cost to RTV-Bench, demonstrating similar effective evaluation scale. We note that these datasets are complementary to our streaming dataset.

C. Robustness to category/task imbalance.

Not every video in the benchmark supports all task types, as is also the case in prior benchmarks such as OVO-Bench and RTV-Bench. Some task types are intentionally omitted when they are not meaningful. For instance, the Future task is excluded in certain Text Understanding/OCR scenarios, where videos often involve egocentric reading and queries target text that is already visible or previously observed, rendering future prediction ill-defined. To evaluate the impact of this imbalance, we report results with Inverse-Category and Inverse-Task reweighting in Tab. 8. Model rankings and overall conclusions remain unchanged, suggesting that the benchmark is robust to variation in task composition.

Benchmark	OVO-Bench	RTV-Bench	**Ours**
Number of video clips	**644**	522	92
MTQA median. duration between annotations (sec.)	4.0	61.8	**1.0**
Total Annotations	3207	4608	**18410**
Avg. unique events per video clip	2	2.5	**11.8**
Total number of unique events	3207	**4608**	2117
Single GPU eval. duration (mins.)	127	395	345

Table 7. **Benchmark Comparison** Single GPU eval duration is measured for Qwen2.5VL-7B-Instruct model.

Reweighting	Qwen3-VL-4B				Qwen3-VL-32B			
	Present	Cumulative	Future	Overall	Present	Cumulative	Future	Overall
Uniform (reported in main paper Tab. 2)	62.3	36.4	17.9	46.8	54.8	32.1	14.5	41.0
Inverse Category	64.6	38.5	26.4	48.4	58.1	37.3	19.9	43.5
Inverse Task	63.3	38.4	24.8	47.8	56.2	35.9	18.9	42.3
Inverse Category and Task	62.3	43.4	17.3	45.8	54.7	41.1	13.7	41.1

Table 8. Model accuracies for various reweighting schemes.

D. Model-specific Optimizations

Prior works like [33, 35] have demonstrated that model-specific optimizations like, KV-cache and embedding reuse strategies in streaming VLMs can substantially reduce inference latency by minimizing prefill time. To complement these approaches, we introduce an optimization technique inspired by speculative decoding methods in language modeling. This method extends speculative decoding to the streaming setting, enabling further reductions in latency without modifying or retraining the underlying model.

D.1. Self-Speculative Decoding for Streaming

To reduce inference latency in the streaming setting, we introduce a variant of self-speculative decoding tailored for continuous video input. Unlike prior approaches that rely on lightweight draft models, we re-purpose the previous timestep's generated tokens as the draft sequence and verify them against the current visual input. Because visual scenes in many streaming applications evolve gradually, the prior response is often still valid, allowing the model to approve the draft without generating new tokens. This verification step is highly efficient, as it can be executed in parallel and avoids full decoding. New tokens are produced only

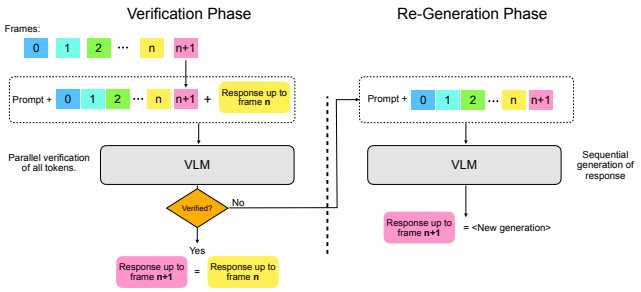

Figure 5. **Overview of self-speculative decoding for streaming VLMs**. The prior response serves as a draft to be verified for the next frame.

Self-Speculative Decoding	Cumulative Task (OCR)		Mean Avg. Latency (s)
	Accuracy	Consistency	
✗	21.5	93.0	5.8
✓	**55.1**	**96.2**	**1.5**

Table 9. **Self-Speculative Decoding for Streaming VLMs.** We evaluate streaming-adapted Qwen3-VL-8B model using the asynchronous protocol and compare the performance of the model with and without self-speculative decoding optimization.

when verification fails, i.e., when the visual stream exhibits a meaningful change requiring an updated response.

We evaluate this technique on a subset of text-recognition videos for cumulative task, with results summarized in Tab. 9. Incorporating our variant of self-speculative decoding yields improvements across all metrics. The substantial reduction in inference latency leads to a 33.6% gain in accuracy under asynchronous protocol, and consistency also increases because the model generates new outputs only when verification fails. This reduces unnecessary rewordings across timesteps, thereby lowering the incidence of inconsistent responses.

E. Accuracy evaluation using a Judge-LLM

For accuracy evaluation, we use GPT-5 (medium reasoning) [20] as a judge model to compare a model's output at each timestep with the reference ground-truth caption. We use a detailed judge prompt to clarify the scoring criteria for the different tasks in the benchmark. Specifically, the judge is designed to reduce variance across evaluation attempts (caused by different reasoning chains). The `<question>`, `<gt_answer>`, and `<model_response>` blocks at the end of prompt are replaced with the corresponding question, ground truth caption, and model response text.

F. Compute Cost Analysis

We evaluate models smaller than 8B on a single video using a single GPU, while sharding 8B models on 2 GPUs and 32/38B models on 4 GPUs. We distribute the evaluation on different videos across a cluster of GPUs. Evaluating Qwen3-VL-2B-Instruct on all videos distributed over 8 GPUs takes 1 hour (8 GPU hours), while evaluating Qwen3-VL-8B-Instruct takes 1.5 hours (12 GPU hours). We utilize H100 GPUs (AWS instance type p5.48xlarge) for all evaluations.

G. End-to-End Latency per Task

We report average end-to-end latency in Tab. 10. Under the asynchronous protocol, accuracy reflects the trade-off between computational delay and temporal fidelity, as delayed responses are evaluated against later frames (Sec. 3.2.2). GPT-5 is accessed via web API, and its associated network latency degrades performance under true streaming conditions compared to locally deployed models such as Qwen3VL and InternVL3.

	Qwen3VL-4B	Qwen3VL-8B	Qwen3VL-32B	FlashVStream (7B)	Dispider (7B)	GPT-5†
Present	1.3	1.6	2.4	0.8	1.7	42.4
Cumulative	2.1	2.2	4.1	1.4	1.2	63.7
Future	2.1	2.7	3.5	0.3	3.3	65.8

Table 10. Average end-to-end latency in seconds per task-type. Video and streaming models were evaluated on NVIDIA H100. † For GPT-5 its the measure of its API latency.

H. Results with SW+U Memory Policy

In Tab. 2, all streaming-adapted video VLMs evaluated under the asynchronous protocol use the sliding-window (SW) memory policy with a buffer size of 64. Here, we report results for streaming-adapted Qwen3-VL models using the sliding-window with uniform tail (SW+U) policy, which provides a better balance between recency and long-range temporal coverage. As shown in Tab. 11, SW+U consistently improves performance, particularly on cumulative tasks; for instance, Qwen3-VL-4B gains 5.2% in cumulative accuracy.

I. Effect of Camera Buffer Size

In Tab. 2, all models evaluated under the asynchronous protocol use a camera buffer of size 600, which is large enough to exceed the maximum number of frames in any video, ensuring that no frames are dropped and accuracy is influenced solely by model latency. However, in a realistic setup,

Model	Async. Accuracy ↑				Async. Consistency ↑
	Present	Cumulative	Future	Overall	
Qwen3-VL-2B [24]	54.0	29.9	9.6	39.0	96.5
Qwen3-VL-4B [24]	**61.6**	39.3	18.4	**47.4**	95.0
Qwen3-VL-8B [24]	57.8	**41.6**	**22.0**	46.8	95.8
Qwen3-VL-32B [24]	55.6	36.1	15.6	42.8	**96.7**

Table 11. Async protocol performance for Qwen3-VL family of models evaluated with SW+U memory policy.

Camera Buffer Size	Tasks			Overall
	Present	Cumulative	Future	
Qwen3-VL-2B				
1	53.8	28.4	9.6	38.5
8	54.2	30.4	9.8	39.3
16	**54.3**	**31.2**	10.1	**39.7**
600	53.6	30.5	**10.5**	39.1
Qwen3-VL-8B				
1	58.4	37.5	21.5	45.7
8	58.0	39.8	22.5	46.4
16	57.8	41.1	21.9	46.6
600	**62.4**	**52.6**	**35.1**	**54.8**

Table 12. **Effect of camera buffer size.** We evaluate Qwen3-VL 2B and 8B sizes using the asynchronous protocol. All runs use memory buffer size 64 and SW+U policy. We highlight the best numbers in each column.

systems cannot assume an unbounded camera buffer. We therefore ablate performance under varying camera buffer sizes, which induces frame drops when the model is too slow to keep pace with the incoming stream. This analysis highlights how constrained buffering alters a model's effective temporal context under the same memory policy. Our implementation exposes this parameter, allowing practitioners to adjust it to match their deployment constraints. In Tab. 12, we report the performance of streaming-adapted Qwen3-VL models for various camera buffer sizes. For models such as Qwen3-VL-8B, reducing the camera buffer to 16 leads to a substantial degradation, with overall accuracy dropping by nearly 15%.

You are the evaluator.
You will receive:
1) A user question
2) A model predicted response
3) A ground truth answer
Your task is to compare the model's response with the ground truth and decide if they match meaningfully.

########## Instruction on how to evaluate ##########

###
Your Output Format:

Return only a JSON dictionary with the following keys:

* 'pred': "yes" if the model response meaningfully matches the ground truth, "no" otherwise.
* 'score': an integer (not a string) between 0 and 3, based on how correct the model response is.

Do not provide any explanation, notes, or text outside the JSON output.
Example output:
'pred': 'yes', 'score': 2

###
Evaluation Rules:

Binary Match (pred key)

* Focus on meaningful equivalence between model response and ground truth.
* Accept synonyms, paraphrases, or reworded answers if meaning is preserved.
* Lists are acceptable if items are semantically aligned or paraphrased.
* All responses that qualify for Tier 3 (perfect match) are automatically labeled 'yes'.
* All responses that qualify for Tier 0 or 1 are automatically labeled 'no'.

Rubric Score (score key)

* Tier 0: No meaningful match.
* Tier 1: Some overlap, but major errors or missing key parts.
* Tier 2: Mostly correct with small mistakes or omissions.
* Tier 3: Perfect or near-perfect match for key elements in the question.

###

Specific Grading Guidelines

* A response is considered a match if it includes the key elements asked in the question. For named entities, different spellings are acceptable.
  * Example: "Valley Tavern" is an acceptable match for ground truth "A beer garden named 'The Valley Tavern'."
  * For general objects, synonyms are acceptable. Example: "shorts" is acceptable for "a pair of gray shorts" when the question only asks "what object."

* If the model response is conceptually related to the ground truth, give partial credit.
  * Example: ”backpack” instead of ”handbag,” ”bottle of green tea” instead of ”bottle of beer,” or ”cloth” instead of ”t-shirt.”
* When the question gives specific choices (e.g., ”Crosswalk,” ”Sidewalk,” or ”Motorway”), the model response must exactly match one of the choices for a tier 3 score.
  * Minor spelling differences are fine.
  * A synonym not in the list (e.g., ”Pavement” for ”Sidewalk”) gets tier 2.
  * Irrelevant responses get tier 0.
* For questions asking about an activity or cooking step (without choices):
  * Include all key elements : tier 3.
  * Miss some elements : tier 2.
  * Only vaguely capture a key element : tier 1.
  * Identify key elements based on the question.
    * Example: If the question asks for the latest cooking step and ground truth is ”Gather the ingredients and lay them out on the counter”, then:
      * Response ”Collect the ingredients” : tier 3.
      * Response ”The person stops pointing at the ingredients and turns to speak to the camera” : tier 1.
    * Example: If the question asks for the latest cooking step and ground truth is ”Mix beans and olive oil well.”, then:
      * Response ”Add corn into the mixture.” : tier 0.
      * Response ”Add black beans into the mixture.” : tier 1.
      * Response ”Combine well.” : tier 2.
      * Response ”Blend black beans and oil thoroughly.” : tier 3.
* When the question asks about a step from a given list of steps (e.g., for cooking or for computer tasks):
  * Model response exactly match ground truth or is its paraphrase capturing key element: tier 3.
  * Model response match ground truth but miss one important element: tier 2.
  * Irrelevant responses get tier 0.
  * For these cases do not assign score 1.
  * Example: Ground truth ”Click the Safari icon in the Dock to open Safari.”
    * Response ”Open Safari.” : tier 3.
    * Response ”Click on the icon in the Dock.” : tier 2.
    * Response ”Click on the button.” : tier 0.
* If the model response misses an important part of the ground truth, give partial credit.
  * Example: Ground truth ”bowl of rice.”
    * Response ”bowl.” : tier 1.
    * Response ”rice.” : tier 2.
* Empty strings, ”None,” or ”NA” (and similar responses) are considered a match.
* When the ground truth is a list and order is not important, grade based on the intersection-over-union (IOU) of items:
  * All items match : tier 3.
  * Most items match ($1 >$IOU> 0.8) : tier 2.
  * Few items match ($0 <$IOU< 0.8) : tier 1.
  * No match : tier 0.
  * Lists can be comma-separated, space-separated, or multiline.
  * Example: Ground truth ”Bowl of rice, beer bottle.” and response ”bowl, carrot” : tier 1.

* When the ground truth is a list and order is important (e.g., question asks for chronological order):
  * All items match with the same order as ground truth : tier 3.
  * All items match with the ground truth but order is different : tier 2.
  * Some items match with the ground truth : tier 1.
  * No match : tier 0.
* For transcription questions:
  * Tier 3: Exact match (minor character-level differences acceptable).
  * Tier 2: Mostly correct; less than 20% of words missing or changed.
  * Tier 1: Partially correct; more than 20% of words missing, changed, or added.
  * Tier 0: Not following the ground truth.
* For counting questions:
  * Exact number match : tier 3 (numeric or word form both acceptable).
  * Format matches but count is wrong : tier 1.
  * Format also incorrect : tier 0.

########## Data for evaluation ##########

Please evaluate the following video-based question-answer pair:

Question:
<question>

Ground truth answer:
<gt_answer>

Model predicted response:
<model_response>

Figure 6. Judge prompt used to evaluate accuracy of model response.

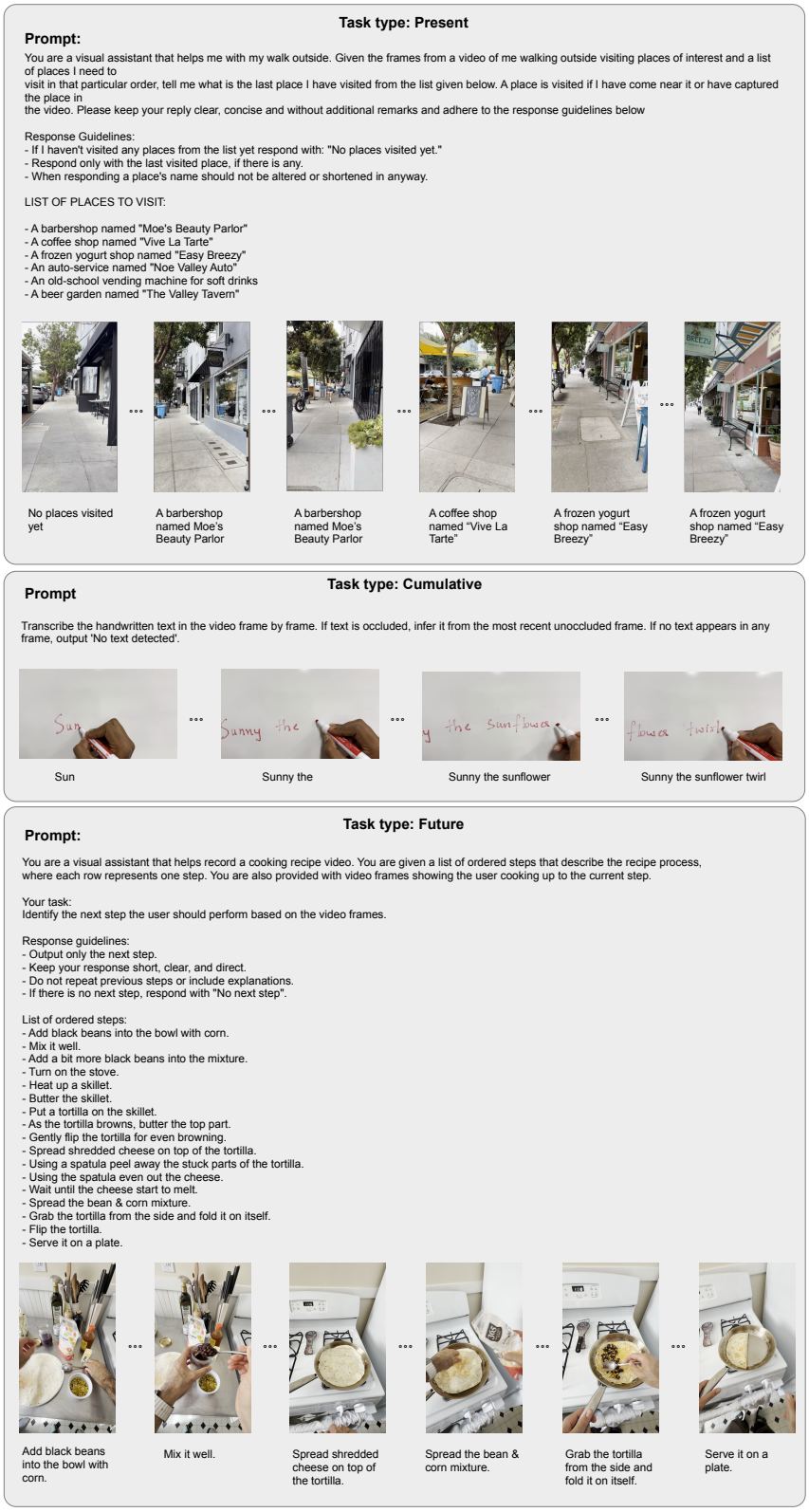

Figure 7. **Examples of VSAS-BENCH task types with frame-level annotations and full prompts.** VSAS-BENCH involves three task types, Present tasks, which focus on currently occurring events; Cumulative tasks, which require the model to reason over past events; and Future tasks, which focus on predicting upcoming events based on ongoing visual cues.