

Cheaper Than Frames: An OCR + Keyframe Index as a Perception Layer for GUI-Video Question Answering

Rushikesh Hiray Shubham Srivastava Azizur Rahman
Laraib Hasan Mohammad Hijjawi Harshit Taneja

University of Birmingham

rhiray03@gmail.com

shubham03workplace@gmail.com

er.azizurrahman@gmail.com

laraib.hasan45@gmail.com

mohammad.hijjawi1997@gmail.com

hxt532@student.bham.ac.uk

Abstract

Multimodal models that interpret screen recordings are usually fed *raw sampled video frames*, a faithful but costly perception layer: each frame is a large image input, and dense coverage multiplies token usage. We ask whether a *cheap, text-first index* (on-screen OCR text plus per-state change descriptions, built once by a small local pipeline) is a better-value perception layer for the same model. On GUI-World, a released benchmark of GUI recordings with human-annotated multiple-choice questions and an objective exact-match metric, we compare the two in a controlled setting (same model, same questions), and replicate across two flagship models. On Claude Opus 4.8, answering from the index reaches 0.850 accuracy vs. 0.902 for an 8-frame baseline ($n=286$) at 44.6% lower cost, and a lossless content compression cuts cost to 75.6% below frames with no measurable accuracy loss. On GPT-5.5 the trade-off is even stronger: the compressed index trails frames by only about 3% (0.874 vs. 0.895) while using 82.8% fewer input tokens. The index is also stable across models, the compressed result differing by only about three points between them (0.843 on Opus 4.8, 0.874 on GPT-5.5). The honest summary is most of the frame baseline’s accuracy at a fraction of the cost. We release the harness, the frozen manifest, and per-item outputs for reproducibility at <https://github.com/blueprintparadise/Screex>.

1 Introduction

Screen recordings, including screencasts, product demos, bug reproductions, and software tutorials, capture rich evidence about how software is used. Large multimodal models (LMMs) are increasingly asked to interpret such recordings: to summarize workflows, answer questions, or recover the sequence of actions. The standard interface is to sample frames from the video and provide those

images to the model. Frames are a direct representation, but they are also an expensive one: each image consumes many input tokens, and because any individual frame may miss the relevant moment, systems typically sample several frames, increasing cost without guaranteeing coverage.

We study an alternative perception layer motivated by a simple property of GUI recordings: much of the salient information is conveyed through text and changes in text. Menu labels, button captions, dialog contents, field values, and lines that appear or disappear after user actions often carry the evidence needed to answer questions about the recording. Our pipeline (SCREEX) constructs a compact *index* of a recording using a small local pass: OCR and keyframe segmentation, with no model calls (Figure 2, appendix). The resulting index is a sequence of UI states, each containing on-screen OCR text and the text *added* or *removed* relative to the previous state. Since the index is primarily text, it is considerably cheaper to provide to a model than the corresponding raw image frames.

The central question is a controlled, same-model comparison: *for a fixed LMM, how accurately can a SCREEX index answer questions about a GUI recording relative to raw sampled frames, and at what cost?* We answer this question on GUI-World (Chen et al., 2024a), using its released multiple-choice QA (MCQA) task and objective exact-match metric. This design keeps the evaluation focused on perception quality: there is no LLM judge, no semantic-threshold tuning, and no need to manufacture ground truth.

Contributions. (1) We introduce a controlled evaluation protocol that isolates the *perception layer* by holding the model and questions fixed while varying only the input representation (index vs. frames), and by measuring both accuracy and token cost. (2) On 286 desktop-domain GUI-World MCQs, the index comes within ~ 5 accuracy

points of an 8-frame baseline while reducing cost by $\sim 45\%$; a frame-budget sweep shows diminishing returns beyond 4 frames, indicating that the cost gap is not removed by simply using fewer frames. (3) We show that a *lossless* content compression of the index (removing a derivable diff representation and factoring repeated UI chrome) reduces input tokens by 56% with accuracy statistically indistinguishable from the uncompressed index. This yields a measured 75.6% cost reduction relative to raw frames and highlights that *data-lossless* encodings are not necessarily *accuracy-lossless*: model-legibility of the encoding still matters. (4) We provide a reproducible harness (a frozen manifest, cached per-item outputs, and cost accounting) together with a reference implementation, and discuss the main engineering control needed for long recordings: bounding the index size so that it remains within model context.

2 Background and Related Work

Several benchmarks address GUI-video understanding or desktop automation. We surveyed them for suitability to a *controlled video-understanding* evaluation with *released* ground truth. We adopt GUI-World (Chen et al., 2024a) because it provides human-annotated ground truth for an understanding task: multiple-choice questions scored by exact match. This avoids the central risk of an action-extraction study, namely constructing our own ground truth, and makes the comparison depend primarily on the perception layer and the model. Appendix B, Table 2, gives the full benchmark survey.

GUI-World’s original baselines evaluate proprietary and open LMMs, including GPT-4V, Gemini, and fine-tuned video models, using sampled frames as input. Our study is orthogonal: we fix one strong model and vary the *perception representation* (raw frames versus an index), which the original benchmark does not isolate. Other relevant settings, including action-extraction benchmarks such as Shangan (Chen et al., 2024b) and instructional-video automation benchmarks (Zhao et al., 2025; Lin et al., 2024; Zhao et al., 2023; Gao et al., 2024), either do not release understanding ground truth or evaluate task execution rather than perception, and so do not support the controlled comparison required here.

Text and structured representations of GUIs. Replacing pixels with a textual or structured repre-

sentation is well established for *single-screen* GUI agents: accessibility trees and DOMs as model inputs (Zhou et al., 2024), set-of-mark prompting over parsed elements (Yang et al., 2023; Lu et al., 2024), GUI grounding from screenshots (Cheng et al., 2024), and screenshot-as-text pretraining (Lee et al., 2023). Our work differs in three respects. First, our unit of analysis is a *recording*, not a screenshot: the index is a temporal sequence of UI states with text-change diffs, rather than a single parsed view. Second, the index is *model-free*, relying only on OCR and keyframe segmentation rather than an auxiliary detector or accessibility API, which screen recordings rarely expose. Third, we explicitly measure the cost-accuracy trade-off against raw frames for the same model and the compressibility of the representation, neither of which is the focus of the agent literature above.

3 Method

3.1 The SCREEX index (Arm B perception)

SCREEX converts a recording into a JSON *index* using a local, model-free pipeline. The pipeline consists of (i) frame sampling at a fixed rate (2 fps in this study); (ii) keyframe/state segmentation, which groups consecutive frames into UI *states* at points of visual or textual change, so that redundant near-identical frames are collapsed; (iii) OCR for each state using RapidOCR / ONNX Runtime, yielding on-screen text lines in reading order; and (iv) a text diff between consecutive states, recording the lines *added* and *removed*. These diffs capture the visible change associated with user actions, such as a menu opening, a field being filled, or a dialog appearing. The final index is a sequence of states, each containing OCR text and add/remove diffs, together with recording metadata. In this study the index is built *audio-off*: no speech transcription is used. Arm B is therefore a visual-only representation (OCR plus keyframe structure), directly comparable to Arm A’s visual frames. Because the representation is text, it is substantially cheaper to send to the model than the corresponding images.

3.2 The two perception arms

Both arms answer the *same* multiple-choice question with the *same* model and must output a single option letter. They differ *only* in the perception input. **Arm A, Raw frames (baseline).** We sample $N=8$ frames uniformly across the video and send them as images alongside the question and options.

This is the standard “frames as perception layer” baseline. **Arm B, SCREEX index.** We build the index and send its JSON text (OCR plus per-state text diffs) alongside the question and options. This arm is text-only and contains no images.

One asymmetry is intentional: the index is distilled from 2 fps sampling and keyframe segmentation, often summarizing tens or hundreds of frames, whereas Arm A receives 8 uniform frames. The comparison is therefore a *dense text index* versus *sparse frames*. This reflects the practical deployment choice: a text index can afford dense temporal coverage that would be expensive as image input. Consequently, part of the index’s accuracy may come from broader coverage rather than from representation alone; we revisit this point in the Limitations.

3.3 Task, metric, and cost

Task. The task is multiple-choice QA. Each item consists of a question, a set of options, and a gold letter. The model is instructed to answer only with the option letter. We parse the first *standalone* option letter from the reply, so that an “a” inside a word such as “answer” is not mistaken for option A. Unparseable replies are counted as incorrect. **Metric.** We report exact-match accuracy, defined as correct / total, both overall and by scenario. **Cost.** For each call, we log `input_tokens` and `output_tokens`. We then price them using the model’s published rates at the time of our experiments (claude-opus-4-8: \$5 / 1M input tokens and \$25 / 1M output tokens), allowing us to report a dollar cost per arm. Dollar figures are an illustrative snapshot: published rates may change, but the input-token reductions we report are pricing-independent. These figures include only *model-API* tokens. Building the index is a one-time local pass (OCR plus keyframe segmentation, with no model calls), amortized across questions about a recording, and is not included in the API-token cost.

4 Experimental Setup

Benchmark. We use GUI-World Annotation/benchmark/*.jsonl. Each record contains one MCQA item (Question, lettered Options, and Correct Answer, e.g., `[[B]]`) together with a video. **Domain.** We restrict evaluation to the desktop domain, namely the *software* and *website* scenarios. Mobile (iOS/Android) and XR are outside the scope of

this study and are excluded. The *multi(-window)* scenario was also excluded after a pilot (Section 6).

Sampling. From the 742 desktop-domain MCQs, we draw a fixed sample of 300 using seed 42. The sample is frozen in a committed manifest for reproducibility and consists of 212 software items and 88 website items. **Model.** Both arms use the same model, `claude-opus-4-8`, with `max_tokens = 1024`. **Arm A budget.** Arm A uses 8 uniformly sampled frames per video. Frames are written at the video’s *native* resolution as PNG (no resizing and no `detail` parameter); the API auto-downscales images above ~ 1.15 MP and tokenizes them at approximately $wh/750$.¹ Measured usage for the 8-frame arm averages 16,388 input tokens per item ($\approx 2,048$ tokens per frame plus the prompt), so frame cost scales roughly linearly with the frame budget. **Arm B settings.** Arm B builds an audio-off index at 2 fps, with JSON capped at 100,000 characters. This is the index’s “perception budget” (Section 6). **Harness.** A standalone package imports SCREEX as a library. Pure logic, including MCQA parsing, dataset filtering, sampling, and cost accounting, is unit-tested with fakes; the model, OCR, and dataset interfaces are thin wrappers. Each item’s output is cached, making the run resumable and re-scorable without additional model calls. Code and the frozen manifest are released at <https://github.com/blueprintparadise/ScreeX>.

Reported set. Of the 300 sampled items, 14 were excluded: 13 because the index, even after capping, exceeded the model’s request/context limit or hit a transient connection error, and 1 because the video file was a corrupt (truncated) download. The reported matched results use the 286 items for which *both* arms produced an answer, evaluated on the identical set.

5 Results

Table 1 gives the main controlled comparison, and Figure 1 plots the three arms as cost-accuracy operating points. A frame-budget sweep (Figure 3, appendix) is discussed afterward because it is run on a separate 60-item subset and supports only the shape of the frame-cost curve.

¹Here w and h are the image’s pixel width and height after the provider’s auto-downscaling; $wh/750$ is the provider’s published image-token approximation at the time of our experiments and is subject to change.

Arm	Exact-match accuracy			Input tok.	Cost (USD)	vs. frames
	Overall ($n=286$)	Software ($n=198$)	Website ($n=88$)			
A. Raw frames (8)	0.902	0.889	0.932	4.69M	23.46	n/a
B. SCREEX index	0.850	0.828	0.898	2.59M	12.99	−44.6%
B'. SCREEX index (compressed)	0.843	0.838	0.852	1.14M	5.73	−75.6%

Table 1: Main result: controlled comparison, same model, both-answered $n=286$. The index is within ~ 5 points of the 8-frame baseline at $\sim 45\%$ lower cost; lossless content compression compounds this to a 75.6% cost reduction versus frames. By paired exact McNemar, the frames>index gap is significant (discordant 31/16, $p=0.04$) whereas the index vs. compressed difference is not (13/11, $p=0.84$): compression is accuracy-neutral on the full set. Output tokens are negligible (single-letter answers): 1,057 / 858 / 1,189 respectively.

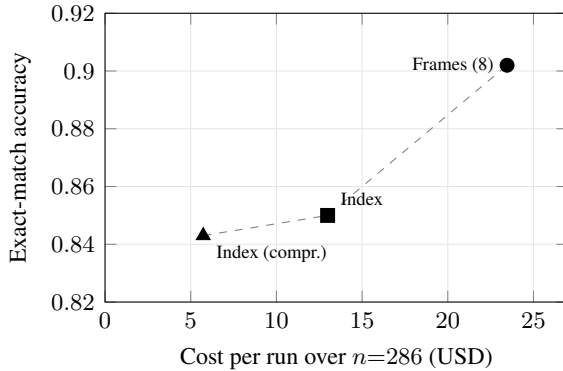


Figure 1: Cost-accuracy operating points (same model, $n=286$; numbers from Table 1). Moving from raw frames to the compressed index cuts cost by 75.6% (\$23.46 to \$5.73) for a ~ 6 -point accuracy reduction; the dashed line traces the frontier.

Efficiency. The raw index uses 55.4% of the frames arm’s input tokens, corresponding to a 44.6% cost reduction (\$12.99 vs. \$23.46). Applying the lossless content compression described in Section 5.1 to the same items (arm B’) reduces cost by a further $\sim 56\%$ relative to the raw index. Because the two savings operate on different bases, they compound to a measured 75.6% reduction relative to raw frames (\$5.73 vs. \$23.46; $\approx 4.0\text{K}$ vs. $\approx 16.4\text{K}$ input tokens per item). Output tokens are negligible for all arms.

Accuracy. Overall, the index is ~ 5 -6 points below the 8-frame baseline (0.850 raw / 0.843 compressed vs. 0.902). The gap is modest but statistically significant: paired exact McNemar gives $p=0.04$ for frames>index (31 frames-only-correct items vs. 16 index-only-correct items). Compression, in contrast, does not produce a significant change relative to the raw index (0.843 vs. 0.850; discordant 13/11, McNemar $p=0.84$). We report several paired comparisons in this section, so we

do not lean on the borderline $p=0.04$ in isolation: it is not corrected for multiple testing and would not survive a strict Bonferroni adjustment. The frames>index conclusion instead rests on the consistent *direction* and effect size of the gap, including its replication on a second flagship model (frames lead on both Opus 4.8 and GPT-5.5; Table 4). The result should therefore be read as most of the frame baseline’s accuracy at roughly a quarter of the cost, not as parity or superiority.

Coverage. The compressed arm also *answers 13 items that the raw index could not* (299 vs. 286). These are long recordings for which the raw index exceeded the request limit (Section 6); after compression, they fit. Compression therefore improves not only cost but also the index’s effective coverage.

Intent-to-treat. Table 1 reports the paired comparison on the 286 items answered by both arms, the appropriate set for McNemar testing, but it excludes the raw index’s hardest overflow cases. Intent-to-treat (Table 3, appendix) reports a complementary view: all 300 sampled items are counted, and any unanswered item is scored as wrong. Under this denominator, frames reach 0.860, the raw index reaches 0.810, and the compressed index reaches 0.840. Because the compressed arm answers 299 items rather than 286, its effective gap to frames narrows to ~ 2 points, a smaller deployment penalty than the matched comparison alone would suggest.

Frame-budget sweep. To check that the 8-frame budget does not itself explain the gap, we sweep $N \in \{1, 2, 4, 8, 16\}$ on a 60-item subset (Appendix D). Accuracy jumps from 0.750 (1-2 frames) to 0.883 at 4, then is flat (8/16: 0.850/0.833, within noise at $n=60$) while token

cost grows linearly: beyond a few frames, images add cost, not accuracy. This subset is unrepresentative (frames@8 score 0.850 here vs. 0.902 on the full 286), so we keep the index-vs-frames comparison on the full set. The takeaway: the index’s cost advantage cannot be recovered by cheaper frame sampling; fewer frames lose accuracy, more only add cost.

Is the gap a frame-selection artifact? The index uses keyframe segmentation while Arm A samples *uniformly*, so the index might look competitive only because the frame baseline is poorly chosen. We rerun the frames arm with its 8 frames picked by the index’s *own* segmentation ($n=274$ completed; on this set uniform frames@8 score 0.898 and the compressed index 0.836). Keyframe-selected frames@8 score *identically* to uniform (0.898 vs. 0.898; McNemar $p=1.0$) using $\approx 30\%$ fewer frames. The visual baseline is thus not under-optimized: given the index’s own frames it scores the same, so the ~ 6 -point deficit is representational, not a selection artifact (frames still lead the index here, 0.898 vs. 0.836, $p=0.0095$). The index’s value is cost, not accuracy parity.

Does it replicate on another model? Because the value argument could hinge on one model’s token accounting, we repeat both arms on a second flagship, GPT-5.5, on the same 286 items (Table 4, appendix; all answered). The pattern holds: the frames>index gap shrinks to 2.1 points (from 5.9 on Opus 4.8) while the compressed index’s token reduction grows to 82.8% (from 75.7%). As token savings are pricing-independent, the trade-off is a property of the representation, not of one model or price schedule.

5.1 Index compression: same accuracy at a fraction of the index cost

Much of the index’s token budget comes from redundancy. Each state contains both its full OCR snapshot and add/removed diffs, even though these are mutually derivable. In addition, persistent UI chrome (menus, toolbars, taskbars, and other repeated elements) appears in every state. In the largest sampled index, 1,618 OCR lines collapse to 359 unique lines ($4.5\times$ duplication). We therefore apply two lossless, model-free post-processing transforms: **(1)** keep a single representation per state (the snapshot) and remove the derivable diff fields; and **(2)** factor persistent UI by hoisting lines present in *every* state into a single

`persistent_ui` block, leaving each state with only its unique text. Across the 286 indexes, this reduces characters by 50% while remaining lossless at the per-state line-set level: each state’s text set reconstructs exactly, as verified by unit tests.

On an 80-item ablation (Table 5, appendix), content compression matches the raw index within sampling noise while nearly halving token usage. The one-item difference (72 vs. 71 of 80) is not statistically meaningful at this sample size. We also tested a compact TOON serialization on top of content compression. It saved a further $\sim 6\%$ tokens but cost ~ 1 additional item on the same subset, so we report content compression in plain JSON and leave format-level compaction for future work. This illustrates the distinction between *data-lossless* and *accuracy-lossless*: an encoding can preserve all information while becoming slightly less legible to the model.

Full-set result ($n=286$). We then run the content-compressed index on the same 286 items as the main study (arm B’ in Table 1). It scores 0.843 overall, compared with 0.850 for the raw index; this difference is not significant (McNemar $p=0.84$), with the small dip concentrated on the website scenario. The compressed index uses 1,139,952 input tokens (\$5.73), a 56.1% reduction relative to the raw index and, compounded with the index’s 44.6% saving over frames, a 75.6% reduction relative to raw frames. The full-set compression saving (56%) is larger than the 80-item ablation saving (48%) because the longest software indexes, which are under-represented in the 80-item subset, compress most strongly. Compression also recovers 13 long recordings whose raw index exceeded the request limit, so the compressed arm answers 299 items versus 286 for the raw index.

6 Analysis and Discussion

Index size is the key engineering variable.

The index is small on average (median ≈ 7 K tokens/item, p99 ≈ 34 K, max ≈ 48 K) but has a heavy tail: a few long software recordings (code editors, language-pack switches, dense or non-Latin text) produce indexes that exceed the model context, yielding “request too large” errors. Since such text tokenizes well above the naive 4-chars/token rule, a 200K-character cap was insufficient; we cap the index at 100K characters, uniformly subsampling states to fit. This is the index arm’s counterpart to the fixed 8-frame budget and leaves ev-

ery otherwise-successful item untouched ($\leq 48K$ tokens). A few pathological raw-JSON indexes still exceed the limit and are excluded from the raw-index matched set (Section 4); content compression (Section 5.1) recovers 13 of them, leaving only the corrupt video unanswered for the compressed arm. Compression is thus both cheaper and a partial remedy for overflow; importance-aware truncation (keep high-change states, summarize the rest) is the natural next step.

Why *multi* was dropped. A 30-item pilot included the multi-window scenario, where the text-only index performed markedly worse and produced the first oversized-index failures. Multi-window understanding plausibly depends on cross-window *visual* layout that a flat OCR index underrepresents. Because it is also a relatively small portion of the desktop domain, we restrict the main study to software plus website scenarios and leave multi-window evaluation for future work, likely requiring keyframe *images* in addition to text.

Where the index lags (and why). We inspect the 31 items that frames answer correctly and the index misses (24 software, 7 website, consistent with the scenario-level gap). Using question wording as a proxy, the index’s loss rate is higher on state/value questions (“which/what value or setting is displayed”: 10/66, 15.2%) than on action/click questions (20/191, 10.5%), consistent with the index struggling when the answer depends on exact visual state that OCR may distort, rather than on *what action* occurred. A split by the share of non-Latin characters in the index’s OCR reinforces this: on the 9 recordings with $\geq 10\%$ non-Latin text (language-pack switches and CJK UIs), the index scores 0.778 versus frames’ 0.889 (gap -11.1); on the 277 mostly-Latin recordings, it scores 0.852 versus 0.903 (gap -5.1). Because $n=9$ is small, this is directional, and non-Latin share is only a proxy for unmeasured OCR error. Both analyses point toward a *hybrid* index that combines text with a small number of keyframe thumbnails, trading some cost advantage for robustness on fine-visual and noisy-OCR cases.

Deployment and use. Beyond the benchmark, the system is deployed as a perception layer for screen-recording understanding in engineering and product workflows, chiefly debugging-session review, implementation traceability, and converting product demonstrations into draft how-to documen-

tation, rather than only as an experimental artifact. Over the course of development, the system has been used to process more than 500 screen recordings and support several thousand downstream question-answering and analysis requests across internal evaluation and engineering workflows. In practice, users employ the index to review bug reproductions, inspect implementation traces, locate workflow steps in long recordings, and generate first-pass documentation from product demos. Because the index is built once and reused across multiple questions, the approach has proven particularly useful for recordings that would otherwise require repeated manual review of the original video.

We report these figures as operational deployment signals rather than controlled measurements. The usage reflects real engineering workflows and exploratory adoption of the system, but the raters, task mix, and evaluation criteria are internal and uncontrolled. Consequently, these observations should be interpreted as evidence of practical utility rather than rigorous effectiveness measurements, and are distinct from the benchmark accuracy results reported above.

Limitations

Subset, not the full benchmark. We evaluate a frozen 300-item desktop-domain sample, with 286 items in the matched comparison. The sample and seed are committed for reproducibility. The results should be read as evidence for this desktop-domain setting, not for all of GUI-World. **Two base models.** The main study uses `claude-opus-4-8`; we replicate the headline trade-off on a second flagship model, GPT-5.5 (§5, Table 4). Both models show the same pattern (frames are modestly more accurate, while the index is much cheaper) but the precise gap and absolute accuracies vary by model. We do not claim generality beyond these two models, and cross-paper GPT-4V/Gemini comparisons are not implied. **Excluded items.** The matched comparison excludes 14/300 items: 13 due to oversized-index/connection failures and 1 due to a corrupt video. This biases the matched set toward shorter recordings. We therefore also report intent-to-treat accuracy over all 300 items (Table 3), where unanswered items are counted as wrong. **Text-only index.** Arm B contains no images. A hybrid text + keyframe index is not evaluated here. **Audio-off, desktop-only, MCQA-only.**

We do not use speech narration, do not evaluate mobile/XR settings, and do not evaluate free-form or LLM-judged QA. These are scope choices rather than claims of generality. **Frame budget.** We sweep $N \in \{1, 2, 4, 8, 16\}$ (Figure 3) and find diminishing returns beyond $N=4$, so the gap is not simply an artifact of choosing 8 frames. The sweep is run on a 60-item subset that is unrepresentative in both directions, so its absolute accuracies should not be compared to the 286-item main table. We claim only the diminishing-returns *shape* from this sweep. A full- n sweep, and sweeps on additional models, remain future work. **Multiple comparisons.** We report several paired McNemar tests and do not apply a family-wise correction; individual p -values (notably the headline $p=0.04$) are therefore uncorrected, and we rely on effect direction and cross-model replication rather than on any single test. **Coverage asymmetry.** The index is distilled from dense 2 fps sampling, whereas the frame baseline receives 8 uniform images. Some of the index’s accuracy therefore reflects broader temporal coverage rather than representation alone. The comparison is realistic for deployment, but it is not a pure isolation of text versus pixels. **Frame selection (tested).** Selecting the frame arm’s 8 frames using the index’s own segmentation, rather than uniform sampling, leaves frame accuracy unchanged (0.898 vs. 0.898, $p=1.0$; §5). Thus, the index’s deficit is not explained by an under-optimized uniform frame baseline. We do not sweep richer frame-selection strategies beyond this control. **Preprocessing cost not in the dollar figures.** Our dollar costs include only model-API tokens. Index construction (2 fps decode, OCR, and keyframe/state segmentation, with no model calls) takes a median of ≈ 50 s of CPU time per recording ($p_{90} \approx 135$ s, $\max \approx 197$ s; $n=49$) on a commodity laptop CPU using RapidOCR on CPU (Appendix D). This pass is model-free and parallelizable, and it is amortized across all questions about a recording. It is cheap in dollars but not free in wall-clock time, so the cost advantage is most appropriate for a build-once/answer-many setting rather than a single-question, latency-sensitive setting. **Compression significance.** A full-set paired McNemar test finds no significant raw-index vs. compressed-index difference ($p=0.84$, $n=286$), consistent with the 80-item ablation. This is evidence for a small or null effect at this sample size, not proof of exact equivalence. **Deployment metrics are uncontrolled.** Any deployment figures reported in the

paper (Section 6) are internal, operational signals, not a controlled study, and any usefulness rating is distinct from the benchmark’s exact-match accuracy. They would indicate real-world utility but should not be read as rigorous effectiveness measurements.

Ethics Statement

This study uses GUI-World (Chen et al., 2024a), a publicly released benchmark of GUI screen recordings with human-annotated multiple-choice questions. We add no new human-subjects data and conduct no annotation. Screen recordings may contain incidental on-screen text, such as file names or window titles. We do not collect or publish raw recordings; we release only the harness, the frozen question manifest (which references the public benchmark), per-item model outputs (option letters and token counts), and derived indexes constructed from the public videos. The system additionally provides a redaction mode that masks secrets and personally identifiable information in on-screen text and blurs the corresponding image regions before any index is written, so that indexes can be shared without exposing sensitive content. The study measures the cost and accuracy of a perception layer and does not involve deployed decisions affecting users. We report dollar costs using published API rates so that the efficiency claim is auditable. We identify no specific risks beyond those already associated with the underlying public benchmark.

References

- Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2024a. [Gui-world: A dataset for gui-oriented multimodal llm-based agents](#).
- Yanting Chen, Yi Ren, Xiaoting Qin, Jue Zhang, Kehong Yuan, Lu Han, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024b. [Sharingan: Extract user action sequence from desktop recordings](#). *arXiv preprint arXiv:2411.08768*.
- Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. [SeeClick: Harnessing GUI grounding for advanced visual GUI agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. ArXiv:2401.10935.

Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchun Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, Hengxu Wang, Luowei Zhou, and Mike Zheng Shou. 2024. [AssistGUI: Task-oriented PC graphical user interface automation](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. ArXiv:2312.13108.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. [Pix2Struct: Screenshot parsing as pretraining for visual language understanding](#). In *International Conference on Machine Learning (ICML)*. ArXiv:2210.03347.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Qinchun Wu, Mingyi Yan, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. 2024. [VideoGUI: A benchmark for GUI automation from instructional videos](#). In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*. ArXiv:2406.10227.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. [OmniParser for pure vision based GUI agent](#). *arXiv preprint arXiv:2408.00203*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. [Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V](#). *arXiv preprint arXiv:2310.11441*.

Dehai Zhao, Zhenchang Xing, Xin Xia, Deheng Ye, Xiwei Xu, and Liming Zhu. 2023. [SeeHow: Workflow extraction from programming screencasts through action-aware video analytics](#). In *International Conference on Software Engineering (ICSE)*. ArXiv:2304.14042.

Henry Hengyuan Zhao, Kaiming Yang, Wendi Yu, Difei Gao, and Mike Zheng Shou. 2025. [Worldgui: An interactive benchmark for desktop gui automation from any starting point](#). *Preprint*, arXiv:2502.08047.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. [WebArena: A realistic web environment for building autonomous agents](#). In *International Conference on Learning Representations (ICLR)*. ArXiv:2307.13854.

A System Pipeline

B Benchmark Survey

C Reproducibility

Repo layout. A `study/` harness (including `compress.py`, `ablation.py`, and `frame_sweep.py` with its committed `frame_sweep_results.json`), unit tests, a frozen sample manifest, a vendored

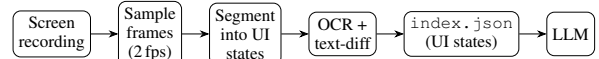


Figure 2: The SCREEX pipeline: a model-free local pass (no model calls) turns a recording into a compact text index of UI states (on-screen OCR text plus per-state add/remove diffs), which any LLM reads in place of raw image frames (a text index needs only a language model, not a multimodal one).

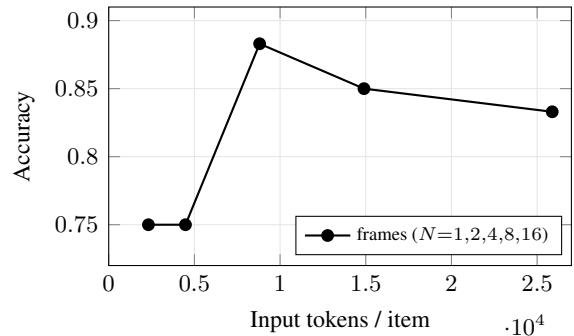


Figure 3: Frame-budget sweep (frames only, 60-item subset). Accuracy rises sharply from 1-2 to 4 frames, then shows diminishing returns (8 and 16 frames do not improve on 4) while token cost grows linearly ($\approx 2.3K$ tokens/frame). The subset is unrepresentative (frames@8 score 0.850 here vs. 0.902 on the full 286), so it anchors only the shape claim; the index-vs-frames comparison is the full-286 one in Table 1.

copy of the SCREEX package, and cloud/parallel run helpers. Per-item model outputs are cached under `study/cache/outputs/`; compression-ablation outputs under `study/cache/ablation/`.

Key knobs. `FRAME_BUDGET=8`, `INDEX_FPS=2.0`, `INDEX_AUDIO=False`, `MAX_INDEX_CHARS=100000`, `SAMPLE_SIZE=300`, `SAMPLE_SEED=42`, pricing \$5/\$25 per 1M input/output tokens.

Scoring. Accuracy is exact-match on the parsed option letter; cost is summed token usage at the listed prices. Results over the both-arms-answered set ($n=286$) are in Section 5. **Content compression.** `single_representation (mode="snapshot")` drops the derivable diff fields; `factor_persistent_ui` hoists all-states OCR lines into a `persistent_ui` block. Lossless at the per-state line-set level (unit-tested); 50% character reduction across the 286 cached indexes.

Dataset	Task	Public data	GT shipped	Metric
GUI-World (Chen et al., 2024a)	GUI video understanding	Yes	Yes (keyframes, captions, 24.8K MCQs)	Exact-match MCQA
Sharingan (Chen et al., 2024b)	Extract user action sequence	Partial	No (GT unreleased)	P/R over typed actions
WorldGUI (Zhao et al., 2025)	Desktop GUI automation	Yes	Audio-to-text; task success	Task success rate
VideoGUI (Lin et al., 2024)	GUI automation from videos	Yes	Yes (eval reimpl. needed)	Hierarchical plan/action
SeeHow (Zhao et al., 2023)	Workflow from screen-casts	Yes	No	n/a
AssistGUI (Gao et al., 2024)	Task-oriented automation	Yes	Execution-based	Task success (live envs)

Table 2: GUI-video benchmarks surveyed. “GT shipped” refers to ground truth for *understanding* tasks, such as released question/answer pairs, as distinct from *execution* ground truth, such as task-success checks in a live environment. Among the surveyed options, GUI-World is the only benchmark that ships real, human-annotated understanding ground truth with an objective metric.

Arm	ITT acc. ($n=300$)	Correct	Answered
Raw frames (8)	0.860	258/300	286
SCREEX index	0.810	243/300	286
SCREEX index (compr.)	0.840	252/300	299

Table 3: Intent-to-treat accuracy ($n=300$; unanswered = wrong). The compressed index (0.840) trails frames (0.860) by only ~ 2 points and is closer than the raw index (0.810), because compression recovers 13 long recordings the raw index could not fit. Discussed in Section 5.

Model	Frames	Compr. index	Gap / tok. red.
Opus 4.8	0.902	0.843	$-5.9 / -75.7\%$
GPT-5.5	0.895	0.874	$-2.1 / -82.8\%$

Table 4: Cross-model replication ($n=286$, both arms; discussed in Section 5). The frames>index accuracy gap and the index’s large input-token reduction hold on both providers’ flagships; on GPT-5.5 the gap is smaller and the token saving larger. GPT-5.5 per-scenario: frames 0.891/0.905, index 0.861/0.905 (software/website). “Gap” is index minus frames accuracy; “tok. red.” is the compressed index’s input-token reduction vs. frames.

D Measured Run Metadata

Items sampled: 300 (software 212 / website 88). Reported: 286. **Frames arm:** 4,687,055 input + 1,057 output tokens, \$23.46. **Index arm (raw):** 2,594,215 input + 858 output tokens, \$12.99. **Index token distribution (successful items):** median $\approx 7.0\text{K}$, $p_{90} \approx 20.1\text{K}$, $p_{99} \approx 33.6\text{K}$, max $\approx 48.4\text{K}$ input tokens. **Compression ablation (80-item subset):** raw index 0.900 (72/80,

Index encoding	Acc.	Correct	vs. raw
Raw index (baseline)	0.900	72/80	n/a
Content-compressed	0.887	71/80	-47.6%
+ TOON serialization	0.875	70/80	-53.2%

Table 5: Compression ablation on an 80-item subset using the same model and questions. The index is built once and re-encoded, so only the encoding varies. “vs. raw” denotes input-token reduction. Content compression matches the raw index within sampling noise (a one-item difference, 72 vs. 71) while reducing input tokens by 47.6%.

714,621 input tok) vs. content-compressed 0.887 (71/80, 374,666 input tok, -47.6%). TOON serialization on the same compression: 0.875 (70/80, -53.2%). Verbose-JSON control for the same compression: 0.887. **Merged full-set compression ($n=286$):** content-compressed index 0.843 (software 0.838 / website 0.852), 1,139,952 input + 1,189 output tokens, \$5.73. That is -56.1% vs. the raw index and -75.6% vs. raw frames. The compressed arm answered 299 items (+13 recovered from raw-index overflow errors). **Frame cost:** 8-frame arm = 16,388 input tokens/item ($\approx 2,048$ tokens/frame); frames written at native resolution as PNG. **Significance (paired exact McNemar):** frames vs. raw index, discordant 31 (frames-only-correct) / 16 (index-only-correct), $p=0.040$; raw index vs. compressed, discordant 13 / 11, $p=0.84$. **Intent-to-treat ($n=300$, unanswered = wrong):** frames 0.860 (258/300, answered 286); raw index 0.810 (243/300, answered 286); compressed index 0.840 (252/300, answered 299). **Discordant losses (frames right, index**

wrong; $n=31$): 24 software / 7 website; by question wording, loss rate 10/66 (15.2%) on state/value questions vs. 20/191 (10.5%) on action/click. **Non-Latin OCR (paired):** $\geq 10\%$ non-Latin characters ($n=9$): index 0.778, frames 0.889 (gap -11.1); $< 10\%$ ($n=277$): index 0.852, frames 0.903 (gap -5.1). Non-Latin share is a proxy for OCR error (no OCR GT available). **Index build latency (audio-off; 2 fps decode + RapidOCR + keyframe/state segmentation; one recording at a time, $n=49$):** min 3.6 s, median 50.5 s, mean 58.9 s, p90 135.4 s, p99/max 196.6 s (median recording 2.7 MB). Hardware: Intel i7-1355U (10C/12T), 39.7 GB RAM; RapidOCR on CPU (ONNX Runtime CPU build, integrated GPU unused). The pass is model-free and parallelizable across recordings (the full study run used 8 workers). **Frame-selection control ($n=274$, same model/prompt/resolution/scoring):** uniform frames@8 acc 0.898 (8.0 frames/item, \$23.46); keyframe-selected frames@8 0.898 (5.5 frames, \$16.89); keyframe-selected frames@4 0.876 (3.5 frames, \$11.21); compressed index 0.836 (\$5.73). McNemar: keyframe-8 vs. uniform-8 8/8 discordant, $p=1.0$; keyframe-8 vs. compressed index 28/11 discordant, $p=0.0095$.

E Qualitative Examples

#	Bucket	Question (gold)	F	I	C	Why it is illustrative
1	index ✓ / frames ×	“Which action was performed last?” (gold B, <i>added a New button</i>)	C ×	B ✓	B ✓	Temporal/ordering: the index encodes every state and diff and captures the final action; 8 uniform frames miss the last step. The index’s coverage advantage.
2	frames ✓ / index ×	“Which port is the Docker container mapped to locally?” (gold B, 5000:8080)	B ✓	A ×	A ×	Exact on-screen text: OCR mis-reads the port mapping; frames let the model read the pixels directly. The index’s text-fidelity weakness.
3	compression preserved	“What change is made to the GUI elements?” (gold C, button renamed)	C ✓	C ✓	C ✓	Raw and compressed index agree and are both correct; content compression did not drop the answer-bearing text.
4	non-Latin, index ×	“What action followed ‘no text to quick redact’?” (gold B)	C ×	C ×	C ×	Redaction app, 26% non-Latin OCR: noise corrupts the text layer, so both index variants fail where OCR is unreliable.

Table 6: Illustrative win/loss cases (one per bucket), from cached per-item results on the $n=286$ set. F/I/C = predicted option for frames (8) / raw index / compressed index; ✓ correct, × wrong. The pattern matches the aggregate error analysis (§6): the index wins on temporal/coverage questions (1), loses on exact on-screen text (2) and non-Latin/OCR-noisy content (4), and compression is answer-preserving (3). Full option lists and predictions are in the released harness.