SABRE: SCALABLE AND AUTOMATED BENCHMARKING OF VLMS UNDER STRESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) advance rapidly, but fixed benchmarks can saturate and constructing targeted stress tests remains labor-intensive. Collecting and annotating existing images does not ensure that they satisfy task-specific visual conditions or challenge current models. To solve this, we present SABRE, a scalable, semi-automated pipeline that converts Test Primers (Markdown Task Designs with Data Schemas) into structured specifications, generated or edited images, and question–answer pairs. A Filtering VLM screens candidates for obvious generation defects and prunes readily solved cases, focusing human review on challenging candidates. Reviewers verify task validity, correct annotations, and perform localized image repair when needed. As our main case study, we instantiate SABRE-Prior to test whether VLMs follow visual evidence over world priors—learned expectations about familiar objects and scenes. Its 600 images and 1,000 questions span four subsets: **Context**, **Texture**, **Attribute**, and **Language Elicitation**. Across six VLMs, macro-average accuracy ranges from 17.8% to 31.3% (22.6% mean). Pilot instantiations on **Counting** and **Spatial** reasoning, aided by procedural scene extensions, further demonstrate how the shared workflow generalizes to settings requiring precise quantitative and geometric control. Together, these components provide a reusable framework for constructing and refreshing targeted VLM stress tests as model capabilities evolve.

1 INTRODUCTION

Vision-language models (VLMs) are advancing rapidly, while reliable benchmarks remain slow to design, construct, and validate(Fu et al., 2024; Yang et al., 2025). As fixed benchmarks saturate, they reveal less about the weaknesses that remain and provide less guidance for further model development. This paper asks a direct question: *Can we construct targeted and reliable VLM stress tests at scale as models evolve?*

Large-scale visual benchmarks have traditionally been built by collecting real images and annotating them with human labor. ImageNet (Deng et al., 2009) and MS COCO (Lin et al., 2015) are landmark examples of this approach. This approach is especially costly for stress tests, which evaluate a target capability under deliberately demanding, distracting, or counterintuitive conditions while keeping the task valid and answerable. **Building stress tests from real-image collections is slow and labor-intensive.** Designers must search for images that satisfy the specific visual conditions required to probe a target capability. Finding relevant images does not ensure that these conditions are met, and even a suitable image may be readily solved by current VLMs. A scene that appears difficult to a human therefore does not necessarily constitute a stress test. To reduce annotation costs, several approaches automatically derive questions or labels from existing image collections. For example, POPE constructs object-existence questions from images and object annotations, while AutoConverter automatically turns existing visual questions into challenging multiple-choice questions (Li et al., 2023; Zhang et al., 2025). These approaches show that much of the annotation process can be automated. However, they remain constrained by the visual content of the source images. Image generation and editing offer a way to construct the required visual conditions directly from test specifications. Image generation alone does not produce a valid benchmark: generators may omit requested objects, alter unrelated regions, or produce incorrect counts (Hu et al., 2023; Huang et al., 2025; Ghosh et al., 2023), causing questions and reference answers derived from the

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.

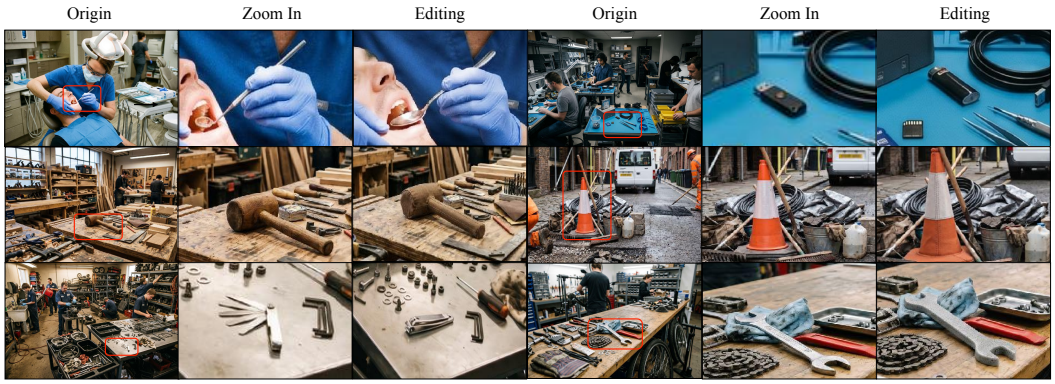


Figure 1: Samples of SABRE-Prior. Each group of three columns shows the source image, a magnified view of the selected region, and the edited result. With SABRE, we can create artificial images that violates world prior with precise visual evidence change.

benchmark specification to disagree with the actual image. Benchmark construction therefore requires specification-faithful generation, model-based difficulty assessment, and explicit validity verification.

To address these challenges, we introduce SABRE, a modular, semi-automated pipeline in which a benchmark designer provides a Test Primer consisting of a natural-language Markdown Task Design, a Data Schema that defines task-specific fields and validation rules, and a question format, such as open-ended generation or multiple choice. The pipeline converts the primer into structured sample specifications. It then generates the required images, applies task-oriented edits when needed, and builds the corresponding question–answer pairs. A Filtering VLM serves as a preliminary quality and difficulty gate: it screens out obvious generation defects and discards cases that it readily solves. This automated screening focuses human inspection and repair efforts on challenging candidates that may reveal model weaknesses. Because automated screening cannot establish task validity, every retained candidate is subsequently verified by human reviewers. For tasks that require a specified image edit, reviewers compare the base image I^b with the edited image I^e to verify that the requested change is present and unrelated content is preserved. Reviewers can revise questions, reference answers, and region annotations, or repair localized image defects without regenerating the entire image. The same interface also accepts uploaded real images, which pass through the same screening and curation stages. This modular design supports scalable construction: additional candidates can be generated from the same Test Primer, while new primers specify different test topics that reuse the generation, screening, verification, and repair workflow.

As our main case study, we instantiate the pipeline as SABRE-Prior,, which tests whether VLMs follow visual evidence when it conflicts with world priors, namely expectations learned from training data about how familiar objects, scenes, and situations normally appear(Liu et al., 2025; Lee et al., 2025; Luo et al., 2025; Vo et al., 2026; Guan et al., 2024). Such conflicts are underrepresented in conventional image collections because they are intentionally unusual and often require precise image edits. Nevertheless, these atypical scenes, edited content, and noncanonical object configurations can test a basic requirement of visual grounding: a model should answer from the image supplied by the user rather than from what it expects to see. Using the pipeline, we construct SABRE-Prior across four complementary subsets: unexpected objects in familiar scenes (**Context**), answers suggested by question wording but unsupported by the image (**Language Elicitation**), counterfactual object materials (**Texture**), and noncanonical component counts (**Attribute**). To examine the pipeline in different stress-test settings, we also apply it to **Counting** and **Spatial** reasoning using separate Test Primers. These small-scale pilots examine whether new task specifications can be used to construct additional stress tests without redesigning the shared construction workflow.

We evaluate SABRE-Prior on six recent frontier VLMs. Across its four world-prior subsets, macro accuracy ranges from 17.8% to 31.3%, with an average of 22.6% across models. The distinct performance profiles across subsets reveal complementary failures in following visual evidence and rejecting unsupported prior expectations. Together, SABRE-Prior and the two pilot instantiations

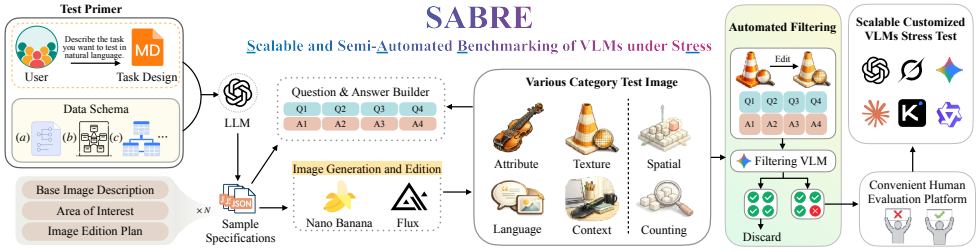


Figure 2: Overview of the SABRE benchmark-construction pipeline.

demonstrate how SABRE supports different stress-test settings through a shared, semi-automated workflow. Our central contribution is a reusable framework for scalable construction, verification, and renewal of targeted VLM stress tests as models evolve.

2 Related Work

VLM Evaluation and Stress Testing. VLM evaluation has expanded from task-specific VQA to broad suites covering perception, reasoning, instruction following, and hallucination, such as MME, MMBench, and MM-Vet, alongside targeted stress tests such as POPE and MMVP (Fu et al., 2025; Liu et al., 2024; Yu et al., 2024; Li et al., 2023; Tong et al., 2024). These benchmarks show that strong aggregate performance can coexist with systematic failures on carefully selected visual cases. However, most are fixed evaluation sets focused on what to measure, whereas our work studies how targeted stress tests can be constructed, screened, and maintained as models and evaluation needs evolve.

Automated and Generative Benchmark Construction. Prior work reduces benchmark construction costs by automatically generating questions, answers, and distractors from existing image collections. For example, POPE derives object-existence questions from image annotations, while AutoConverter transforms existing VQA questions into challenging multiple-choice evaluations (Li et al., 2023; Zhang et al., 2025). Although these methods automate substantial annotation work, they cannot control the visual evidence contained in the source images. Image generation, editing, and procedural rendering address this limitation by directly constructing evaluation images. ImageNet-D and JourneyBench use generated images to introduce controlled or unusual visual conditions, while Vision-Language Bootstrapping, Auto-Comp, and InfiniBench use dynamic image editing, synthetic image pairs, and customizable 3D scenes (Zhang et al., 2024; Wang et al., 2024; Yang et al., 2025; Sbrolli et al., 2026; Wang et al., 2026). However, generative construction makes sample validity dependent on whether each image realizes its intended specification. Our pipeline therefore integrates structured sample specifications, image generation and targeted editing, pressure screening, human verification, annotation revision, and localized repair. It also supports generated images and uploaded real images within the same curation workflow.

World Priors and Visual Evidence Conflicts. A growing body of work studies whether VLMs follow visual evidence when it conflicts with language or world knowledge. The PhD benchmark includes a counter-common-sense subset, PhD-ccs, that uses AI-generated images violating everyday expectations to evaluate visual hallucination (Liu et al., 2025). VLind-Bench measures language priors using DALL-E 3-generated counterfactual images together with auxiliary tests of commonsense knowledge, visual perception, and commonsense bias, thereby reducing alternative explanations for a model's failure (Lee et al., 2025). ViLP constructs out-of-distribution image–question–answer triplets in which one condition can be answered from textual priors while the others require evidence from generated images (Luo et al., 2025). VLMBias makes controlled counterfactual modifications to familiar subjects, such as well-known logos, animals, flags, and game pieces, to test whether memorized canonical knowledge overrides visual counting and identification (Vo et al., 2026). HallusionBench is broader than a world-prior benchmark: it uses human-collected and human-edited control pairs to diagnose both language hallucination and visual illusion, although some edited cases explicitly place visual content in conflict with common knowledge (Guan et al., 2024).



Figure 3: Samples of SABRE on attribute modification(left) and counting(right).

These benchmarks directly motivate our world-prior case study, and we include them as comparison benchmarks in our experiments. Our primary goal, however, is not to introduce another isolated test of prior reliance, but to use this demanding setting to instantiate and evaluate a general pipeline for scalable stress-test construction.

3 METHOD

3.1 OVERVIEW AND PROBLEM FORMULATION

Our goal is to provide a scalable pipeline for constructing targeted VLM stress tests from task specifications. The pipeline takes a `Test Primer`, denoted by S, which consists of a natural-language Markdown `Task Design`, a `Data Schema` that defines task-specific fields and validation rules, and a question format, such as open-ended generation or multiple choice. Given S, the pipeline automatically constructs a set of `Candidate Samples` before human verification:

$$\mathcal{C}(S) = \{x_i\}_{i=1}^{N}.$$

Each candidate $x_i = (I_i, q_i, y_i)$ consists of an image I_i, a question q_i, and its reference answer y_i. Related image–question pairs derived from the same underlying image construction or visual intervention form a case.

As illustrated in Figure 2, an LLM first converts the test primer S into `Sample Specifications`. Each sample specification is a JSON file containing the information required by the subsequent steps, such as image generation and editing prompts. Let z_i denote the i-th sample specification. An image generation model, denoted by G, generates the required images. When a design requires an image modification, an image editing model, denoted by E, edits the generated image. The candidates then enter `Automated Filtering`, where a VLM, denoted by M and called the `Filtering VLM`, screens for obvious generation defects and evaluates the task questions. The pipeline uses these responses to filter out readily solved cases according to task-specific screening criteria, focusing human review on challenging candidates. Candidates retained by automated screening undergo human verification through our curation platform. Human reviewers assess visual quality and task validity, correct annotations, and perform localized image repair when necessary. Human verification determines whether the screened candidates are valid and answerable and can be included in the final benchmark. Accepted candidates become `Samples` in the final benchmark.

3.2 MODULAR TASK RECIPES

A test primer S defines one stress-test topic. It contains a natural-language Markdown task design, an data schema, and a selected question format. The Markdown file states what to test and how to construct the images. It can specify the scene, the visual content to add, remove, or modify, and the content that must remain unchanged. The question format specifies how the VLM should answer, such as open generation or multiple choice.

The data schema controls the structure of the JSON output. It defines the data types (e.g., text or lists) and other constraints. The pipeline uses the default data schema when the task recipe does not provide a custom one.

A design LLM reads S and produces sample specifications in JSON. Each sample specification contains the information needed to construct one candidate sample, including an image generation or editing prompt, a question, and its reference answer.

New stress-test topics are introduced through new test primers. The pipeline reuses a shared image generation, pressure screening, human verification, image repair, and export workflow. This separation provides extensibility without requiring a new benchmark-construction pipeline for each topic.

3.3 Automatic Candidate Sample Construction

Each sample specification z_i contains an image prompt, a question q_i, and a reference answer y_i. For a single-image task, the image generation model G generates an image I_i from the prompt p_i:

$$I_i = G(p_i).$$

Some stress-test topics test whether a VLM responds to a specific visual change, such as removing an object or changing an object attribute. These topics require image editing. The edit aims to change the target visual evidence while preserving the rest of the scene. This setup tests whether the VLM updates its answer based on the intended change.

For an editing task, z_i contains a base-generation prompt p_i^b and an editing prompt p_i^e. The pipeline uses G to generate the base image I_i^b and the image editing model E to produce the edited image I_i^e:

$$I_i^b = G(p_i^b), \qquad I_i^e = E(I_i^b, p_i^e).$$

The editing prompt states what to change and what to preserve. A real image can replace I_i^b, which skips the image generation step.

The pipeline pairs each applicable image with one question and its reference answer. Each sample specification produces exactly one candidate sample:

$$x = (I, q, y).$$

After human verification, each accepted sample forms or joins a case. A case contains one sample for one-to-one tasks and multiple samples when several image–question pairs derive from the same underlying image construction or visual intervention. The pipeline stores the prompts, model identifiers, and generation records as metadata.

3.4 Model-in-the-Loop Pressure Screening

A Filtering VLM M serves as a preliminary quality and difficulty gate. It first screens candidates for obvious generation defects. Candidates that pass this initial check undergo task-based evaluation to identify challenging instances for human verification.

For each candidate sample $x_i = (I_i, q_i, y_i)$, M produces a response:

$$r_i = M(I_i, q_i).$$

The evaluator compares r_i with the reference answer y_i using the scoring rule for the selected question format:

$$e_i = \mathrm{Eval}(r_i, y_i), \qquad e_i \in \{0, 1\},$$

where $e_i = 1$ indicates a correct response and $e_i = 0$ indicates an incorrect response.

Following the preliminary quality check, pressure screening retains candidates with incorrect task responses ($e_i = 0$) for human verification and excludes correctly answered candidates ($e_i = 1$) from the pressure-selected pool. The retained set is denoted by

$$\mathcal{C}_{\mathrm{pressure}}(S; M) \subseteq \mathcal{C}(S).$$

This selection focuses human inspection and repair efforts on candidates that may reveal model weaknesses. The screening scores are computed before human revisions or localized image repair.

Automated screening does not establish task validity. An incorrect response may reflect a genuine model failure, but it may also result from a generation or editing defect missed during the initial check, an ambiguous question, or an incorrect reference answer. Every retained candidate must therefore undergo human verification to confirm that the visual evidence supports an unambiguous question and a correct reference answer before final inclusion.

3.5 Human Verification and Localized Image Repair

Candidate samples selected for review based on the filtering results undergo human verification through our curation platform. Reviewers inspect the image I_i, question q_i, reference answer y_i, sample specification z_i. A candidate sample passes verification only when the required visual evidence is present, the requested edit is correctly applied when needed, the question is unambiguous, and the reference answer matches the image. Reviewers can accept or reject a candidate sample, revise its question or reference answer, or repair a local image defect.

For image repair, the reviewer selects a bounding box b_i and provides a repair instruction u_i. The platform expands the selected region to include its surrounding visual context and crops the resulting patch P_i from the image. It marks the target region inside the patch and sends the marked patch, rather than the full image, to Gemini 3.1 Flash Image Preview with u_i. Restricting the model input to a local patch focuses the edit on the selected defect and prevents unintended changes to the rest of the image.

The repaired patch $\widetilde{P}_i$ cannot be directly pasted into the image because a hard boundary may introduce visible color and texture discontinuities. We therefore construct a soft mask A_i by expanding the target region and applying Gaussian blur to its boundary. The platform blends the repaired and original patches as

$$P_i^\star = A_i \odot \widetilde{P}_i + (1 - A_i) \odot P_i.$$

It then places $P_i^\star$ back into the original image. The center of the mask preserves the repaired content, while the smooth boundary gradually transitions to the original pixels. This soft blending reduces rectangular seams and integrates the repaired region more naturally into the surrounding scene.

The reviewer compares the repaired image with the original and may accept it, repair it again, or reject the candidate sample. Our questionnaire study shows that participants consistently preferred our localized repair results for removing the target entity naturally while preserving unrelated image content, compared with direct whole-image Gemini editing and the other evaluated methods.

Let $x_i^\star$ denote the candidate sample after human verification and any reviewer revisions or image repairs. The reviewer assigns a validity decision $v_i \in \{0, 1\}$, where $v_i = 1$ indicates acceptance. The final benchmark is

$$\mathcal{D}(S; M) = \{x_i^\star \mid x_i \in \mathcal{C}_{\text{pressure}}(S; M),\ v_i = 1\}.$$

Each item in $\mathcal{D}(S; M)$ originates from a pressure-selected candidate sample and passes human verification. We call each retained item a sample.

The curation platform also supports real-image authoring. Users can upload a real image, provide or revise its question and reference answer, and apply localized image repair when needed.

4 Benchmark Instantiation

As our main case study, we use our pipeline to construct SABRE-Prior, a benchmark that tests whether VLMs answer according to the image when its content conflicts with their learned expectations about familiar objects and scenes. We call these expectations *world priors*.

SABRE-PRIOR **Stress-Test Subsets.** The benchmark contains four complementary subsets. **Context** places unexpected objects in familiar scenes, such as replacing a microscope in a laboratory with a toaster. **Language Elicitation** uses question wording to suggest an answer that is unsupported by the image, such as asking what is inside a closed box. **Texture** assigns familiar objects counterfactual materials, such as giving a ceramic mug a furry surface. **Attribute** modifies canonical component counts while preserving object identity, such as showing a chair with five legs.

Context and Texture use paired base–edit images with four complementary yes/no probes. Language Elicitation uses single-image, four-option questions whose correct answer is *unknown*, while Attribute uses open-ended counting questions over edited images.

SABRE-PRIOR **Construction, Scale, and Evaluation.** All four subsets use the shared workflow in Section 3. For each subset, GPT-5.4 reads its task primer S and produces sample specifications. FLUX.2 [flex], FLUX.2 [pro] (Black Forest Labs, 2025), and Gemini 3.1 Flash Image (Nano Banana

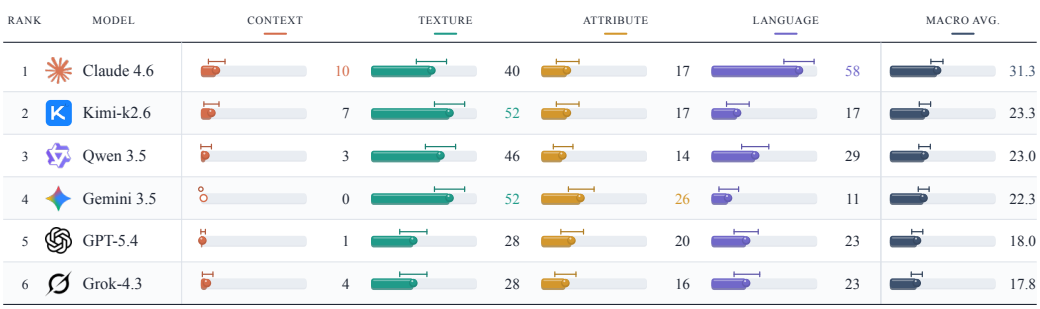


Figure 4: **Accuracy (%) of six frontier VLMs on SABRE-Prior.** Bars show point estimates, and whiskers indicate the corresponding confidence intervals. Context and Texture report strict All4 accuracy: a case is correct only when all four Base–Edited yes/no probes are answered correctly. Attribute uses normalized exact-match counting accuracy, while Language Elicitation uses four-option multiple-choice accuracy. Macro Average is computed across the four subset accuracies.

2) (Raisinghani, 2026) generate or edit the images specified by these specifications. Gemini 3.5 Flash serves as the filtering VLM M. Candidate samples retained by the automated filtering undergo human verification and localized repair when needed. Each subset contains 100 cases. The final benchmark contains 400 cases, 600 images, and 1,000 samples, with one question per sample.

Context and Texture are evaluated using paired Base and Edited images. **Context** asks whether the source and target objects are visible in each image, with expected answers *yes/no* for Base and *no/yes* for Edited. **Texture** uses the same four-probe structure for the expected and counterfactual materials, again producing *yes/no* for Base and *no/yes* for Edited. A case is counted as correct only when all four yes/no responses are correct (All4). This criterion jointly assesses recognition of present evidence and rejection of absent evidence before and after the visual intervention. Individual-probe accuracies and the accuracy of answering both Edited-image probes correctly (Edit2) are reported in Table 8. **Attribute** presents a single edited image with an open-ended component-counting question; the extracted numeric response must exactly match the annotated visible count after normalizing digits and number words. **Language Elicitation** tests whether a VLM gives a language-suggested answer when the image does not provide sufficient evidence to determine the requested information. Each sample uses a single image and a four-option question. The correct option is *unknown*, with its position balanced across the subset. Selecting a specific answer in this setting constitutes an unsupported inference; the evaluation measures this observable failure without attributing it to a single underlying mechanism. Each question is evaluated independently, without cross-question history or any instruction indicating that all correct answers in this subset are *unknown*. We report each subset's primary accuracy and their macro-average as the overall score.

Additional Pipeline Instantiations To evaluate the extensibility of our pipeline, we use the same workflow to construct two additional pilots: Counting and Spatial. Counting presents dense scenes containing dozens of target objects and visually similar distractors, and asks the model to report the number of visible target objects. Responses are scored by normalized exact count. Spatial presents dense multi-layer 3D lattices and includes multiple question formats, such as identifying the color and shape at a specified coordinate or counting objects that satisfy spatial and visual constraints, for example red cubes on the top layer. Responses are scored by normalized exact attribute matching or exact count, depending on the question type. We further provide two procedural generators for counting and spatial reasoning, with explicit control over object counts, spatial layouts, and reference answers. Their construction procedures and evaluation results are presented in Appendix F.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTING

We evaluate six frontier VLMs on SABRE-Prior: Gemini 3.5 Flash (Google DeepMind, 2026), GPT-5.4 (OpenAI, 2026), Claude 4.6 Sonnet (Anthropic, 2026), Kimi-k2.6 (Moonshot AI, 2026), Qwen

3.5 27B (Qwen Team, 2026), and Grok-4.3 (xAI, 2026). These models represent the current state of the art, and our stress tests are designed to identify their blindspots for future improvement. All models are evaluated with zero-shot prompts using greedy decoding (i.e., temperature $= 0$) when supported. We estimate 95% confidence intervals using a percentile bootstrap with 20,000 resamples, resampling cases within each subset. For each resample, we recompute every SABRE-Prior subset accuracy and their macro-average, defined as the unweighted mean of the subset accuracies. Detailed model identifiers, access dates, prompts, and response-parsing rules are provided in the Appendix. We additionally evaluate VCD (Leng et al., 2024) and SoM (Yang et al., 2023) on the open-source Qwen 3.5 27B model to examine whether visual-enhancement methods improve performance. We also construct a real-image variant of Attribute subset to test whether model failures persist when the edited source images are real rather than generated. We also show more examples shown in 6 and 7.

5.2 MAIN RESULTS

Figure 4 reports the primary subset scores of the six evaluated VLMs. Macro-average accuracy ranges from 17.8% to 31.3%, with a mean of 22.6% across models. Because SABRE-Prior uses pressure screening and evaluates different behaviors with different question formats, the macro-average summarizes performance on this benchmark rather than providing a general ranking of model capability. Table 2 provides complementary evidence. Gemini 3.5 Flash achieves 57.3%–90.0% accuracy on five existing benchmarks but only 22.3% on SABRE-Prior. Its low score on SABRE-Prior partly reflects its role as the Filtering VLM. However, the other five models, which were not used during pressure screening, also score below 32%. These results show that the constructed cases expose failures across frontier VLMs rather than only the Filtering VLM.

Context obtains the lowest scores under the criterion: no model exceeds 10%, and the mean is 4.2%. This reflects the difficulty of consistently recognizing both the original and counterfactual objects across the Base–Edit intervention. Texture scores are higher, ranging from 28% to 52%, suggesting that visible material evidence is more accessible than context-incongruent object identity, although canonical material associations still exert substantial pressure. Attribute ranges from 14% to 26%, combining fine-grained structural perception and exact counting with the need to override canonical component counts. Language Elicitation shows the widest variation, from 11% to 58%. This subset should not be interpreted as a pure measure of visual recognition: the queried information is intentionally unavailable in the image. Instead, it tests whether a model recognizes insufficient visual evidence and rejects a plausible answer induced by the question wording. The variation reflects differences in models' willingness to abstain under suggestive but insufficient evidence; performance on answerable questions is outside the scope of this subset. Models perform very differently across subsets. Claude 4.6 achieves the highest macro-average, largely because of its strong Language Elicitation score. Gemini 3.5 Flash and Kimi-k2.6 perform best on Texture, while Gemini 3.5 Flash leads on Attribute but answers no Context cases correctly. The model ranking therefore changes across subsets, and no model performs best on all of them. These results show that SABRE-Prior exposes distinct model weaknesses that a single aggregate score would hide.

Method	**Ctx**	**Tex**	**Att**	**Lang**	**Avg.**
Qwen 3.5 27B	3	46	14	29	23.0
- VCD	1	34	16	27	19.5
- SoM	6	24	12	25	16.8

Table 1: **Effect of visual-enhancement methods on Qwen 3.5 27B.** Results use the same metrics as Figure 4; Avg. denotes their macro-average.

5.3 COMPARISON WITH EXISTING BENCHMARKS

We compare SABRE-Prior with five related benchmarks. As shown in Table 2, all three models obtain substantially lower scores on SABRE-Prior, indicating that the constructed cases remain challenging across models rather than only for the Filtering VLM. The benchmarks use different question formats and scoring rules, so their scores are not directly equivalent. Nevertheless, these results highlight remaining weaknesses that motivate the construction of new, targeted stress tests as models improve.

5.4 Effect of Visual-Enhancement Methods

A useful stress-test benchmark should not only expose model failures but also measure whether proposed methods fix them. We use SABRE-Prior to evaluate two visual-enhancement methods on Qwen 3.5 27B. Visual Contrastive Decoding (VCD) contrasts predictions from original and perturbed visual inputs, while Set-of-Mark (SoM) adds region markers to improve visual grounding. We evaluate both methods using the same questions and subset metrics as the standard Qwen 3.5 27B setting.As shown in Table 1, neither method consistently improves performance. VCD and SoM improve selected subsets but reduce accuracy on others, resulting in lower macro-averages than the standard model. These results indicate that the failures exposed by SABRE-Prior cannot be fixed simply by highlighting visual regions or changing the decoding signal. This experiment also demonstrates the diagnostic value of SABRE-Prior. It shows where a mitigation method helps, where it introduces regressions, and which model weaknesses remain unresolved. The benchmark can therefore support the development and evaluation of methods designed to address specific VLM failures.

Benchmark	Gemini 3.5 Flash	GPT-5.4	Claude Sonnet 4.6
PhD-CCS	82.3	79.2	80.6
VLind-Bench	90.0	81.1	85.5
ViLP	70.3	62.4	69.8
HallusionBench	81.6	74.8	75.6
VLMBias	57.3	45.2	47.1
SABRE-Prior	**22.3**	**18.0**	**31.3**

Table 2: Accuracy (%) of three VLMs on the evaluated benchmark subsets. Results use benchmark-specific metrics; SABRE-Prior reports the macro-average over its four subsets.

5.5 Additional Stress-Test Instantiations

We use two different test primers, to construct additional stress tests for counting and spatial reasoning. Each stress test contains 20 samples and uses the same generation, pressure screening, human verification, and repair workflow. Figure 5 shows that both stress tests are challenging for all six VLMs. Each model answers at most one of the 20 Counting samples correctly, and none answers any Spatial sample correctly. These results contrast with the strong performance of frontier VLMs on many established benchmarks, which can become saturated as models improve. They also demonstrate the extensibility of our pipeline: different task specifications can be used to construct challenging stress tests for new visual capabilities without changing the shared workflow. We further evaluate counting and spatial reasoning using procedurally generated test sets with controlled difficulty levels. Appendix F describes their construction, evaluation protocols, and results.

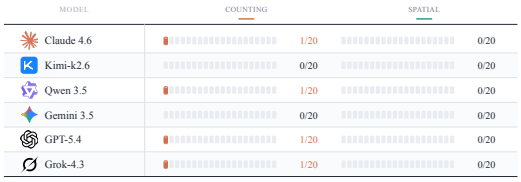


Figure 5: **Results on two additional stress-test instantiations.** Each cell represents one sample, and colored cells indicate correct model responses.

6 Conclusion and Limitations

We introduced SABRE, a scalable and modular pipeline for constructing VLM stress tests from natural-language task specifications. The pipeline converts each task specification into structured case specifications, constructs the required images and question–answer pairs, and uses filtering-model feedback to guide candidate selection, followed by human verification and localized repair. Using this pipeline, we constructed SABRE-Prior, covering four forms of conflict between visual evidence and world priors. Six recent VLMs achieve only 17.8%–31.3% macro accuracy and exhibit distinct failure profiles. We also apply the same pipeline to Counting and Spatial reasoning through separate task specifications, demonstrating its use across different stress-test settings. A limitation shared by any fixed benchmark suite is that its coverage reflects a finite set of task specifications at the time of release. The modular design of SABRE mitigates this limitation by allowing new task specifications and construction modules to be added as VLM capabilities evolve.

AI USE STATEMENT

Generative AI tools were used solely for English language polishing and grammar correction. All AI-assisted text was thoroughly reviewed and verified by the authors to ensure accuracy. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

This work complies with the ICLR Code of Ethics. We do not foresee any direct negative societal impacts or ethical concerns resulting from this work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, all methodological workflows are thoroughly elaborated in the main text and the appendix. The complete source code and experimental environments will be made publicly available upon acceptance.

REFERENCES

Anthropic. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, February 2026. Accessed: 2026-07-20.

Black Forest Labs. FLUX.2: Frontier visual intelligence. https://bfl.ai/blog/flux-2, November 2025. Accessed: 2026-07-20.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Chaoyou Fu, Yi-Fan Zhang, Shukang Yin, Bo Li, Xinyu Fang, Sirui Zhao, Haodong Duan, Xing Sun, Ziwei Liu, Liang Wang, Caifeng Shan, and Ran He. Mme-survey: A comprehensive survey on evaluation of multimodal llms, 2024. URL https://arxiv.org/abs/2411.15296.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2025. URL https://arxiv.org/abs/2306.13394.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment, 2023. URL https://arxiv.org/abs/2310.11513.

Google DeepMind. Gemini 3.5 flash: Model card. https://deepmind.google/models/model-cards/gemini-3-5-flash/, May 2026. Accessed: 2026-07-20.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models, 2024. URL https://arxiv.org/abs/2310.14566.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20349–20360, 2023. doi: 10.1109/ICCV51070.2023.01866.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation, 2025. URL https://arxiv.org/abs/2307.06350.

Kang-il Lee, Minbeom Kim, Seunghyun Yoon, Minsung Kim, Dongryeol Lee, Hyukhun Koh, and Kyomin Jung. VLind-bench: Measuring language priors in large vision-language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational*

Linguistics: NAACL 2025, pp. 4129–4144, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.231. URL https://aclanthology.org/2025.findings-naacl.231/.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13872–13882, 2024. doi: 10.1109/CVPR52733.2024.01316.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20/.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Jiazhen Liu, Yuhan Fu, Ruobing Xie, Runquan Xie, Xingwu Sun, Fengzong Lian, Zhanhui Kang, and Xirong Li. Phd: A chatgpt-prompted visual hallucination evaluation dataset, 2025. URL https://arxiv.org/abs/2403.11116.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player?, 2024. URL https://arxiv.org/abs/2307.06281.

Tiange Luo, Ang Cao, Gunhee Lee, Justin Johnson, and Honglak Lee. Probing visual language priors in VLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41120–41156. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/luo25b.html.

Moonshot AI. Kimi K2.6: Advancing open-source coding. https://www.kimi.com/blog/kimi-k2-6, April 2026. Accessed: 2026-07-20.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, March 2026. Accessed: 2026-07-20.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026. Accessed: 2026-07-20.

Naina Raisinghani. Nano banana 2: Combining pro capabilities with lightning-fast speed. https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/, February 2026. Accessed: 2026-07-20.

Cristian Sbrolli, Matteo Matteucci, and Toshihiko Yamasaki. Auto-comp: An automated pipeline for scalable compositional probing of contrastive vision-language models, 2026. URL https://arxiv.org/abs/2602.02043.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9568–9578, June 2024.

An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased, 2026. URL https://arxiv.org/abs/2505.23941.

Haoming Wang, Qiyao Xue, and Wei Gao. Infinibench: Infinite benchmarking for visual spatial reasoning with customizable scene complexity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21594–21605, June 2026.

Zhecan Wang, Junzhang Liu, Chia-Wei Tang, Hani Alomari, Anushka Sivakumar, Rui Sun, Wenhao Li, Md. Atabuzzaman, Hammad Ayyubi, Haoxuan You, Alvi Ishmam, Kai-Wei Chang, Shih-Fu Chang, and Chris Thomas. Journeybench: A challenging one-stop vision-language understanding benchmark of generated images. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 63110–63123. Curran Associates, Inc., 2024. doi: 10.52202/079017-2015. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/734abb86d3caa949f44da8a093717f61-Paper-Datasets_and_Benchmarks_Track.pdf.

xAI. Grok 4.3. https://docs.x.ai/developers/models/grok-4.3, 2026. Accessed: 2026-07-20.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.

Yue Yang, Shuibo Zhang, Kaipeng Zhang, Yi Bin, Yu Wang, Ping Luo, and Wenqi Shao. Dynamic multimodal evaluation with flexible complexity by vision-language bootstrapping. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21399–21420, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/36d9468ebdb76b9b229fbd343fff84d5-Paper-Conference.pdf.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-vet: Evaluating large multimodal models for integrated capabilities. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57730–57754. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yu24o.html.

Chenshuang Zhang, Fei Pan, Junmo Kim, In So Kweon, and Chengzhi Mao. Imagenet-d: Benchmarking neural network robustness on diffusion synthetic object. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21752–21762, June 2024.

Yuhui Zhang, Yuchang Su, Yiming Liu, Xiaohan Wang, James Burgess, Elaine Sui, Chenyu Wang, Josiah Aklilu, Alejandro Lozano, Anjiang Wei, Ludwig Schmidt, and Serena Yeung-Levy. Automated generation of challenging multiple-choice questions for vision language model evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29580–29590, June 2025.

A Detailed Experimental Setup

This section describes the evaluation settings for SABRE-Prior and the two 20-sample stress tests reported in the main text. The filtering-model ablation and the procedural counting and spatial experiments use the settings specified separately in Appendices E and F, respectively.

Evaluated Models. Table 3 lists the exact model identifiers and inference interfaces used in our experiments. All models were evaluated using the versions exposed under these identifiers on that date. Each model received one image–question sample at a time without in-context examples.

Inference Prompts. We use zero-shot, format-specific prompts and do not request explanations or chain-of-thought reasoning. Context and Texture use the following prompt:

```
Question: {question}
Answer with only one word: yes or no.
```

Attribute and Counting use:

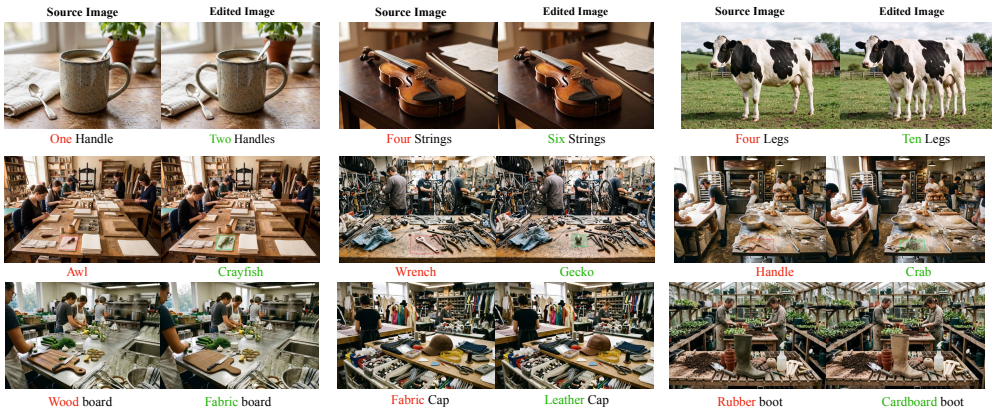


Figure 6: Examples of generated images from SABRE

```
Question: {question}
Answer with only the count or shortest possible count phrase.  Do not
explain.
```

Language Elicitation presents the question followed by four options labeled A–D and uses:

```
Question: {question and options}
Answer with only the option letter. Do not explain.
```

Spatial uses the same short-answer instruction as Counting. Depending on the question, the expected response is either a count or a short combination of attributes such as a color and shape. For the local Qwen evaluation, these instructions are wrapped in the model's standard chat template with a format-only system message requesting no reasoning or additional text.

Decoding and Image Input. The maximum output length is restricted to 8–16 tokens for API models, depending on the response format. Qwen uses at most 32 newly generated tokens. Images are supplied at their original stored resolution to API models. GPT-5.4 uses the *high* image-detail setting. For local Qwen inference, images are converted to RGB and processed using the checkpoint's *AutoProcessor*. Qwen is evaluated with *do_sample=false*, *bfloat16* precision, batch size 1, and automatic device placement. The four subsets are distributed across four CUDA devices, with the subsets evaluated independently.

Response Parsing. Yes/no responses are lowercased and mapped to *yes* or *no* when the response begins with the corresponding word. If necessary, we accept the answer when exactly one of the two words occurs in the response; ambiguous or unparseable outputs are marked incorrect. Multiple-choice responses are parsed as the first standalone letter from A to D.

For open-count questions, responses are lowercased, punctuation is removed, and number words from zero to twenty are mapped to digits. We then extract the first valid numeric count. Equivalent forms such as "seven" and "7" are therefore treated identically, while explanations or other text do not receive partial credit. Spatial attribute responses are normalized for capitalization, spacing, and punctuation before exact matching. An output that cannot be normalized to the required response format is counted as incorrect.

Scoring. Attribute, Language Elicitation, Counting, and Spatial are scored independently at the sample level. Context and Texture contain four samples for each Base–Edited case: two questions are asked about the Base image and two about the Edited image. A case is correct only if all four responses are correct, which we refer to as strict All4 accuracy. Language Elicitation is evaluated using four-option multiple-choice accuracy, with the position of the *unknown* answer balanced across the dataset. Attribute and Counting use normalized exact-count accuracy. Spatial uses normalized exact count or exact attribute matching according to the question type. The overall SABRE-Prior

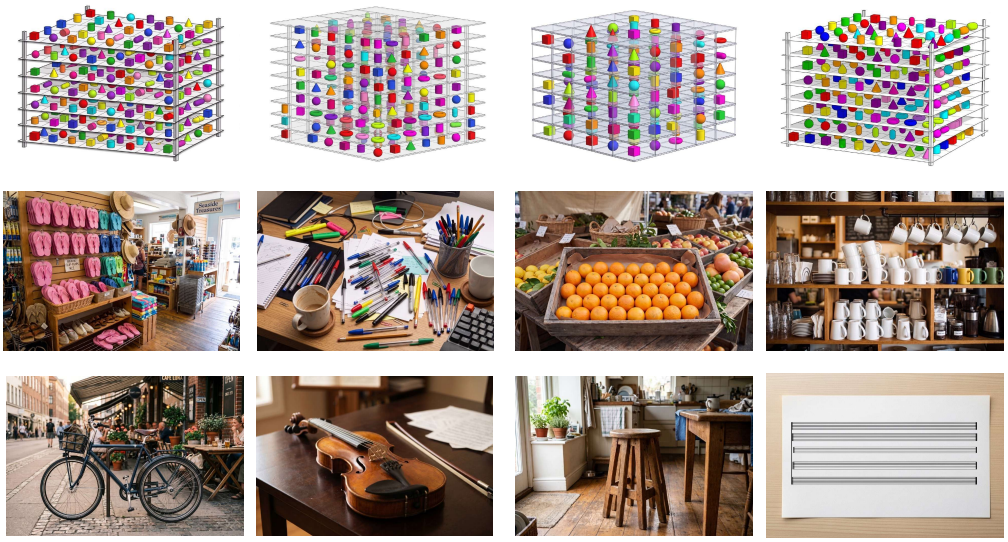


Figure 7: Additional examples of generated images from SABRE. The first row shows Spatial QA instances, where the question-answer format follows the same structure as illustrated in the main paper. The second row presents Counting instances, including examples that require counting pink slippers, pens, oranges, and mugs from left to right. The third row provides additional Attribute instances.

Reported Name	**Model Identifier**	**Interface**	**Inference Configuration**
Gemini 3.5 Flash	*gemini-3.5-flash*	Google Gen AI API	Temperature 0; thinking budget 0
GPT-5.4	*gpt-5.4*	OpenAI Responses API	Temperature 0; reasoning effort *none*; image detail *high*
Claude 4.6	*claude-sonnet-4-6*	OpenRouter	Temperature 0
Kimi-k2.6	*kimi-k2.6*	Official Moonshot API	Thinking disabled; provider-supported deterministic configuration
Qwen 3.5 27B	*Qwen3.5-27B*	Local Hugging Face Transformers	Greedy decoding; thinking disabled; bfloat16
Grok-4.3	*grok-4.3*	OpenRouter	Temperature 0

Table 3: **Models and inference configurations used in our experiments.** Greedy decoding is used whenever supported by the corresponding inference interface.

score is the unweighted macro-average of the Context, Texture, Attribute, and Language Elicitation subset accuracies.

Confidence Intervals. We compute 95% confidence intervals using a percentile bootstrap with 20,000 resamples and random seed 42. Resampling is performed over cases within each subset. Thus, the four samples belonging to a Context or Texture case remain grouped and are resampled together. For every bootstrap replicate, we recompute the four subset accuracies and their unweighted macro-average. The reported interval endpoints are the 2.5th and 97.5th percentiles of the resulting bootstrap distributions.

B TEST PRIMERS AND SAMPLE SPECIFICATIONS

A benchmark designer begins by providing a *Test Primer*, which consists of a natural-language Task Design written in Markdown and a Data Schema that defines the required output fields and validation rules. The Task Design describes the capability or failure mode to test, the visual construction

Field	Type	Purpose
concept_id	String	Unique identifier used to reject duplicate specifications and connect subsequent construction records.
pressure_test_type	String	Name of the stress-test topic, such as *context_prior*.
objective	String	Capability or failure mode that the specification is intended to test.
generation_mode	Enum	Either *paired_edit* or *single_image*.
base_prompt	String	Prompt used to construct the Base image for a paired intervention.
edit_prompt	String	Local transformation applied to the Base image to produce the Edited image.
image_prompt	String	Prompt used for tasks that require only one generated image.
task_attributes	List	Task-specific structured values, such as source entity, target entity, material, count, or spatial relation.
questions	List	Question specifications containing the probe identifier, image role, question type, prompt, reference answer, evaluation rule, and optional multiple-choice options.
review_targets	List	Instructions identifying the visual evidence and image region that a human reviewer should verify.
generation_notes	List	Additional constraints used during image generation, editing, and review.

Table 4: **Principal fields in the default Sample Specification schema.**

requirements, and the desired question format. The schema converts these requirements into a machine-verifiable contract. A designer may use the default schema provided by SABRE or replace it with a task-specific Pydantic schema when additional fields or validation rules are required.

In our implementation, GPT-5.4 receives the Task Design together with the schema and generates one structured Sample Specification at a time. The output is accepted only if it satisfies the schema; invalid outputs and duplicate concept identifiers are returned to the design model with validation feedback for regeneration. Valid specifications are stored as JSONL records and passed to the image-construction and question-building modules.

Default Data Schema. Table 4 summarizes the principal fields in the default Sample Specification schema. The schema supports paired image editing and single-image generation, as well as yes/no, multiple-choice, and open-generation questions.

The schema also enforces cross-field constraints. A *paired_edit* specification must contain both a Base prompt and an edit prompt, and its questions may refer only to the Base or Edited image. A *single_image* specification must instead contain an image prompt, and all questions must refer to that image. Yes/no questions require *yes_no_exact* evaluation and a reference answer of either *yes* or *no*; multiple-choice answers must identify one of the provided options; and open-generation questions must specify an exact-match, counting, or containment-based evaluation rule.

Context Task Design. We use the Context subset as a running example. Its Task Design instructs the design model to construct a natural Base scene containing one contextually expected source entity and no target entity. The Edited image must replace only the source entity with a visually compatible but contextually unexpected target while preserving the rest of the scene. The Task Design also defines four complementary yes/no probes:

1. Source entity visible in Base: *yes*.
2. Target entity visible in Base: *no*.
3. Source entity visible in Edited: *no*.
4. Target entity visible in Edited: *yes*.

The Markdown further requires both entities to remain human-recognizable while occupying a minor region of a visually dense scene. It prohibits duplicate source or target instances, readable labels, large isolated objects, obvious editing artifacts, and changes outside the specified region.

Example Context Sample Specification. The following abbreviated record is an actual schema-compatible specification generated for the Context task. Long image prompts are shortened only for presentation 8.

Sample Specification `Context / paired_edit`


```
{
  "concept_id": "ctx01_repack_tape_bagel",
  "pressure_test_type": "context_prior",
  "generation_mode": "paired_edit",
  "objective": "Test whether a VLM follows visible evidence when an expected object is
    replaced by an unexpected object.",

  "base_prompt": "Generate a documentary photograph of a busy warehouse repacking floor.
     Include exactly one tan roll of masking tape on the lower shelf. No bagel or
    additional tape roll is visible. ...",

  "edit_prompt": "Replace only the masking tape with one plain bagel at the same
    location. Preserve all other objects, people, lighting, camera properties, and
    scene content. ...",

  "task_attributes": [
    {
      "name": "source_entity",
      "value": "roll of masking tape"
    },
    {
      "name": "target_entity",
      "value": "plain bagel"
    },
    {
      "name": "review_location",
      "value": "lower shelf of the central repacking bench"
    }
  ],

  "questions": [
    {
      "probe_id": "base_source",
      "image_role": "base",
      "question_type": "yes_no",
      "prompt": "Is a roll of masking tape visible?",
      "answer": "yes",
      "eval_type": "yes_no_exact"
    },
    {
      "probe_id": "base_target",
      "image_role": "base",
      "question_type": "yes_no",
      "prompt": "Is a plain bagel visible?",
      "answer": "no",
      "eval_type": "yes_no_exact"
    },
    {
      "probe_id": "edited_source",
      "image_role": "edited",
      "question_type": "yes_no",
      "prompt": "Is a roll of masking tape visible?",
      "answer": "no",
      "eval_type": "yes_no_exact"
    },
    {
      "probe_id": "edited_target",
      "image_role": "edited",
      "question_type": "yes_no",
```

```
      "prompt": "Is a plain bagel visible?",
      "answer": "yes",
      "eval_type": "yes_no_exact"
    }
  ],

  "review_targets": [
    {
      "label": "source object",
      "image_role": "base",
      "description": "Verify one tape roll at the specified location."
    },
    {
      "label": "target object",
      "image_role": "edited",
      "description": "Verify that one bagel replaces the tape roll."
    }
  ]
}
```

Figure 8: **Abbreviated Context Sample Specification.** The record connects visual construction, question definition, and human verification within a schema-validated specification. Long prompts are shortened for presentation.

Question Instantiation. The questions are specified before image construction; the Question Builder does not use another model to generate them. Once the required images are available, it associates each question with the appropriate image through its *image_role*, assigns a unique identifier, and exports the canonical records used for filtering, human review, and final evaluation.
For Context, the four probes share the same case identifier and Base–Edited assets and are evaluated jointly using All4. Other subsets use the same question-instantiation interface with their task-specific question formats and scoring rules. Complete Test Primers and schemas will be released with the benchmark-construction code.

C TASK-SPECIFIC INSTANTIATIONS

We instantiate the pipeline on the four SABRE-Prior subsets and two additional pilots. Each instantiation uses a different Test Primer, visual construction requirement, question format, and validity criterion, while retaining the same construction, filtering, verification, repair, and export workflow. Table 5 summarizes these settings. Context and Texture use paired Base–Edited cases, whereas Attribute and Language Elicitation use a single final image. Counting and Spatial are evaluated at the individual-image level and do not use the All4 case-level metric.

Context. A Context specification defines a dense, coherent Base scene containing exactly one source entity that is natural for the scene and no instance of the target entity. The source is placed in a precisely described review location and remains identifiable without becoming the focal object. The target is selected to have a similar apparent size, orientation, color, or silhouette so that it can occupy the same location after editing while remaining contextually unexpected.
The edit instruction replaces only the source with the target and preserves the camera, people, lighting, composition, surrounding objects, and local occlusion pattern. The specification contains four question records: *base_source*, *base_target*, *edited_source*, and *edited_target*, with reference answers *yes*, *no*, *no*, and *yes*, respectively. A candidate is visually valid only when both entities are human-recognizable in their intended images, neither entity appears in the wrong image, and no unrelated scene content changes in a way that affects the questions.

Texture. A Texture specification begins with a familiar entity displaying its expected surface, such as a plastic watering can or a wooden rolling pin. The counterfactual material is chosen to conflict with the entity's canonical appearance while providing recognizable visual evidence, such as visible fabric fibers, foam structure, glass transparency, or a paper-like surface. The intervention changes the visible material of the target entity but preserves its category, geometry, scale, pose, and location.
The four probes ask whether the entity has the canonical or counterfactual surface in the Base and Edited images. Their answer pattern is again *yes/no* for Base and *no/yes* for Edited. Human validation

Instantiation	Image Mode	Task-Specific Construction	Question Format	Primary Validity Requirement
Context	Paired Base–Edited	Replace one contextually expected source entity with a visually compatible but contextually unexpected target.	Four yes/no probes	The source appears only in Base, the target appears only in Edited, and all non-target content is preserved.
Texture	Paired Base–Edited	Change the visible surface of a familiar entity from its canonical material to a counterfactual material while preserving its identity and geometry.	Four yes/no probes	Canonical and counterfactual surface cues are visually distinguishable, and the intervention does not change the entity or surrounding scene.
Attribute	Single final image	Construct a familiar entity with a noncanonical number of visible components while preserving its category identity.	Open-ended count	Every counted component is visible and individually distinguishable, and the annotated count matches the image exactly.
Language Elicitation	Single image	Construct a scene in which the question suggests a plausible answer although the queried evidence is hidden, occluded, unreadable, or outside the frame.	Four-option MCQ	The requested information cannot be determined from the image, making *unknown* the uniquely supported answer.
Counting	Independently scored images	Construct dense scenes containing many instances of a precisely defined target category together with visually similar distractors.	Open-ended count	Every target instance has a visible boundary, distractors are distinguishable, and the reference count is exact.
Spatial	Independently scored images	Construct dense multi-layer 3D lattices with controlled object colors, shapes, coordinates, and spatial relations.	Open-ended attribute or count	Each question has one visually verifiable answer under an explicitly defined coordinate system or spatial constraint.

Table 5: **Task-specific instantiations of the shared SABRE pipeline.** The main SABRE-Prior benchmark contains Context, Texture, Attribute, and Language Elicitation. Counting and Spatial are reported separately as additional pipeline instantiations.

checks that the surface change covers the intended visible region, that the material cues are sufficiently clear for direct visual judgment, and that the edit does not deform the entity or alter neighboring objects.

Attribute. Attribute specifications target familiar objects whose components normally follow a canonical count, including animal legs, chair legs, shirt buttons, instrument holes, and repeated symbols on playing cards. The image is constructed or edited to contain a noncanonical number of visible components while retaining the object's recognizable identity. The intervention must produce complete components rather than duplicated fragments, merged structures, shadows, or ambiguous partial instances.
Each exported image is associated with one open-ended question asking for the visible component count. The reference answer is derived from the verified image rather than assumed from the construction prompt. During validation, the reviewer confirms the object identity, checks each visible component, and ensures that occlusion or malformed geometry does not make the count ambiguous.

Language Elicitation. Language Elicitation specifications construct scenes in which contextual wording makes one answer appear plausible although the necessary evidence is not available in the image. Examples include asking about the contents of a closed opaque container, text on a face-down object, information hidden by glare or blur, or content outside the image boundary. The image prompt

must ensure that the queried information is genuinely unavailable and that no incidental visual cue reveals the intended distractor.
Each question contains three plausible content answers and one *unknown* option. The plausible alternatives are selected to reflect answers that may be suggested by the scene or question wording, rather than visually supported answers. The position of *unknown* is shuffled and balanced across the subset. A candidate is valid only when none of the three content options can be confirmed from the image and *unknown* is therefore the uniquely supported response.

Counting. Counting specifications define a target category using visible properties such as object type, color, pattern, or location. The scene contains dozens of target instances distributed across natural clusters, together with distractors sharing selected properties with the target. For example, a scene may contain yellow lemons among other yellow or round objects, or four-hole blue buttons among buttons and beads with similar colors and shapes. Targets are not arranged in regular rows and may touch or overlap slightly, but each instance must retain a visible boundary.
The construction process may produce Base and Edited images with different target counts. The edit adds or removes a specified number of targets while preserving the scene and distractors. Unlike Context and Texture, each image is exported as an independent open-count sample. Human verification determines the final visible count and rejects images containing merged, duplicated, heavily occluded, or otherwise uncountable target instances.

Spatial. Spatial specifications construct dense multi-layer 3D lattices containing repeated colored shapes at controlled cells. A fixed coordinate convention defines the front-left-bottom cell as the origin cell, with the horizontal axis increasing from left to right, depth increasing from front to back, and height increasing from bottom to top. The same convention is stated explicitly in every coordinate question to avoid dependence on an unstated viewpoint.
The task supports coordinate lookup questions, such as identifying the color and shape at a specified cell, as well as constrained counting questions, such as counting red cubes on the top layer. Specifications may also move a target object between cells while preserving the lattice, camera, and all other objects. As with Counting, Base and Edited images are evaluated independently. A candidate is valid only when the lattice orientation is visually recoverable, the queried cells or layers remain visible, the target objects are distinguishable from distractors, and the reference answer can be verified without relying on the generation prompt.

D Human Verification and Curation Platform

D.1 Annotation Platform

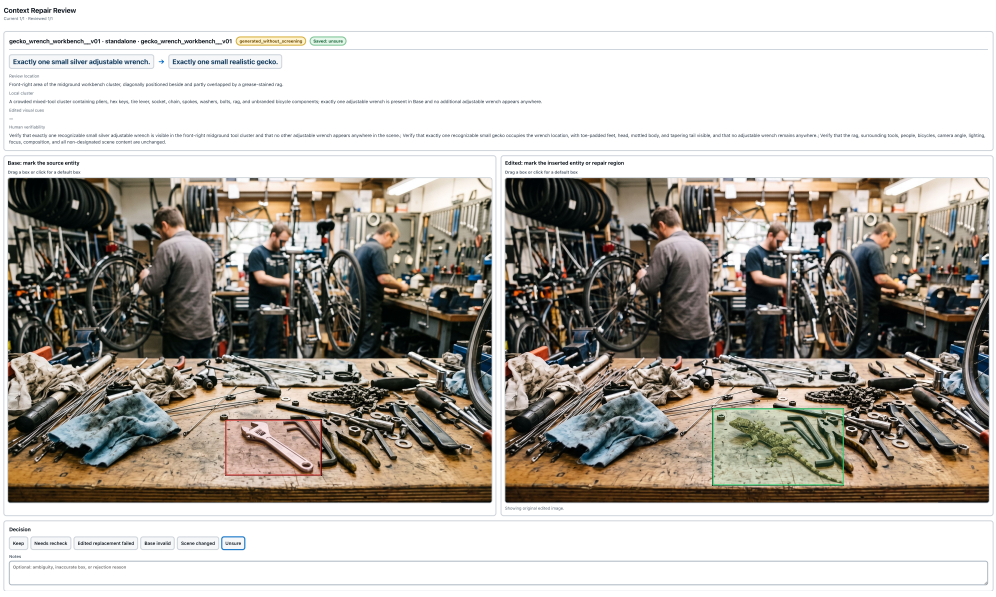


Figure 9: Overview of the SABRE annotation platform.

Figure 9 shows the platform used to verify candidates retained by automated pressure filtering. For paired-image cases, the Base and Edited images are displayed side by side together with the intended source-to- target transformation and a natural-language description of the review location. This location cue helps reviewers find the relevant evidence in visually dense scenes. Reviewers mark the source entity in the Base image and the inserted entity or edited region in the Edited image using bounding boxes. The normalized box coordinates are stored as curation metadata, allowing the regions to be highlighted during later inspection and reused for localized repair. These annotations are not provided to the evaluated VLMs. Reviewers inspect whether the Base image is valid, whether the intended transformation is correctly realized, and whether unrelated scene content is preserved. The interface also presents the four pressure-screening probes, including their reference answers, filtering-model predictions, and correctness. A valid case is marked as *Keep*; otherwise, it can be marked as requiring reinspection, having a failed edit, having an invalid Base image, containing unintended scene changes, or being uncertain. Reviewers may also record notes explaining ambiguous cases or rejection decisions. The platform additionally provides localized image repair and real-image authoring tools. Reviewers can select a defective region in an Edited image, generate a repaired candidate, compare it with the current image, and either accept the repair or restore the original version. The real-image authoring interface supports importing photographs, specifying source and target entities, constructing edited candidates, defining question–answer pairs, and exporting the resulting annotations in the same structured format. Across all subsets of SABRE-Prior, each case was first reviewed by one reviewer and then checked by a second reviewer. Disagreements were resolved through discussion between the two reviewers before the final inclusion decision.

D.2 Localized Image Repair

Fine-grained full-image editing is frequently unreliable, especially when the requested change affects only a small object in a visually complex scene. A generation model may ignore the requested change, leave visible remnants of the original entity, or modify unrelated objects and scene content. Repeatedly regenerating the complete image can therefore destroy otherwise valid visual evidence and weaken the controlled relationship between the Base and Edited images. To address this problem, our curation platform provides localized patch repair, as illustrated in Figure 10. When a reviewer identifies a defective region, they draw a bounding box around the failed edit or local artifact and select a repair operation. The system expands the selected region slightly to retain its surrounding visual context, crops the resulting patch, and sends only this marked patch to the image generation model together with a task-specific repair instruction. The repaired patch is resized when necessary and blended back into the original image using a soft mask. This procedure preserves the image outside the local neighborhood and reduces unintended changes to unrelated scene content.

The platform supports four repair operations: removing an unwanted entity, swapping one entity for another, cleaning local visual artifacts, and restoring the background. Figure 10 presents the two entity-level operations. For *Swap Entity*, the reviewer specifies both the source and target entities; for *Remove Entity*, the selected object is removed and the exposed region is completed using its local context. Reviewers may further refine the repair instruction to specify expected appearance, scale, lighting, or content preservation constraints.

A generated repair is treated as a candidate rather than being accepted automatically. The platform displays the original Edited image and repair preview side by side, marks the case as requiring reinspection, and allows the reviewer to accept the repair or return to the previous image. When a repair is accepted, the previous Edited image is retained as a backup and the repair operation is recorded. The repaired case must then pass human verification before it is included in the benchmark.

D.3 Blind User Study of Localized Repair Quality

We conduct a randomized, method-anonymous user study to assess participant preferences among image repair methods. The survey interface is shown in Figure 11. The study contains 20 image-repair cases. Each question presents an original image with a red bounding box indicating an unwanted entity, followed by four candidate repair results. Participants select the result that best fulfills the requested edit while maintaining visual naturalness. The instructions require complete removal of the marked entity and preservation of unrelated image content.

We compare four methods: Gemini prompt-only full-image editing, OpenCV inpainting, Gemini patch generation with hard pasting, and our Gemini patch generation with soft-mask blending. The hard-paste baseline and our method use the same generated patch, isolating the effect of the compositing strategy. Candidate results are displayed only as *Method 1–Method 4*, without revealing their underlying methods. Their display positions are independently shuffled for every question, and

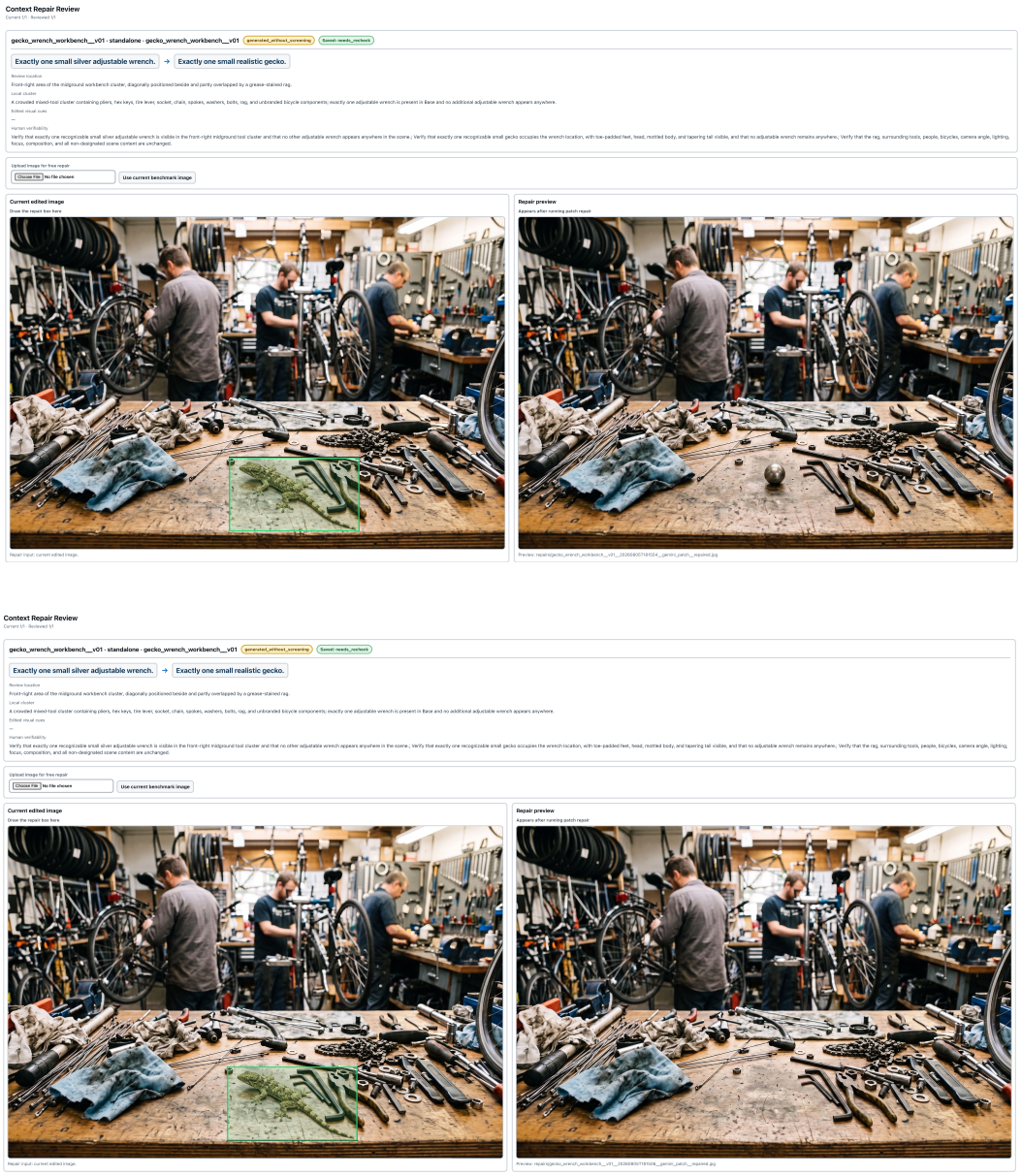


Figure 10: **Localized image repair in the curation platform.** Top: *Swap Entity* replaces the gecko inside the selected region with a ball. Bottom: *Remove Entity* removes the gecko and reconstructs the surrounding tabletop. The current Edited image is shown on the left and the repair preview on the right. Reviewers can accept the repaired image or revert to the original Edited image.

the system records both the randomized order and the selected method. Each question requires one selection and cannot be skipped.

Participants were recruited through a university computer science community group chat. Participation was voluntary and unpaid. Forty survey sessions were initiated, of which 20 were completed by 20 distinct participants. Each participant completed all 20 comparisons, yielding 400 valid selections. As shown in Table 6, our method is selected in 374 of the 400 comparisons, corresponding to an overall preference rate of 93.5%. Because every participant completes the same 20 questions, this is also the mean participant-level preference rate. The standard deviation across participants is 8.1 percentage points, with individual preference rates ranging from 70% to 100%. On these repair cases, participants preferred localized patch generation with soft-mask blending over full-image editing, OpenCV inpainting, and hard patch compositing.

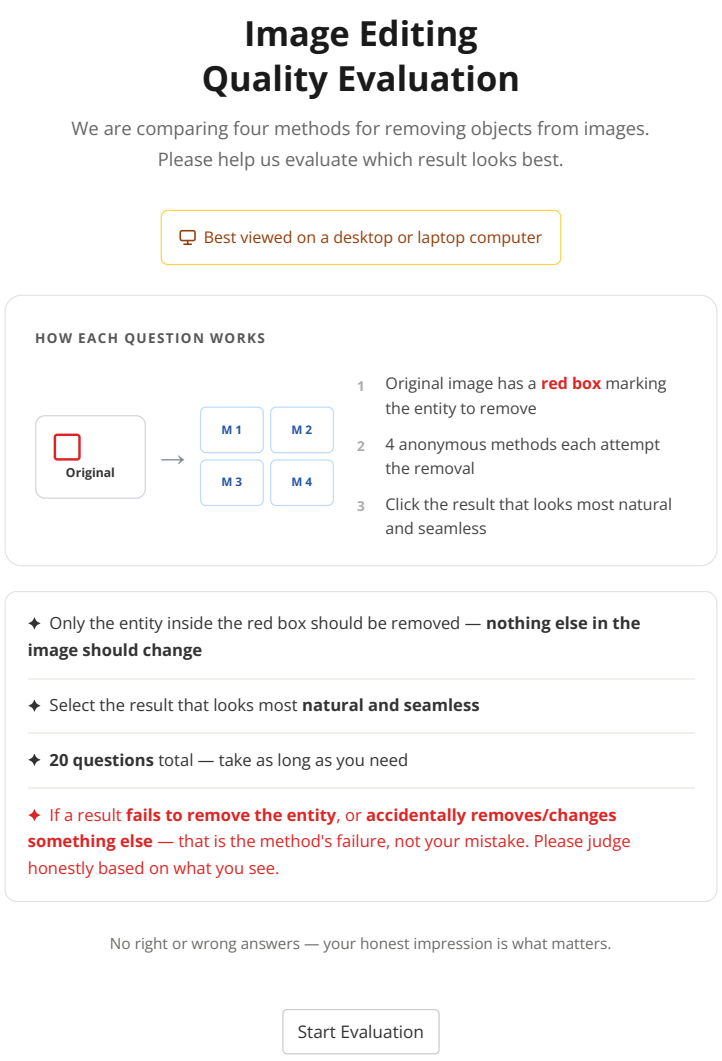


Figure 11: **Instructions for the method-anonymous repair-quality study.** Participants select the most natural result that removes the marked entity while preserving unrelated image content.

D.4 Context Construction and Cost Analysis

We use the Context subset to illustrate the resources required for benchmark construction. Construction effort depends on the task specification, the quality of the generation and editing prompts, and the image model's ability to realize the requested visual conditions. The following statistics therefore describe the Context construction process rather than a uniform cost across all tasks.

Construction and Retention. The candidate pool comprised 220 cases, each containing a Base image, an Edited image, and four question–answer pairs. Filtering-model feedback guided candidate selection, while human review assessed visual quality and task validity. The final subset contains 100 cases, comprising 200 images and 400 question–answer pairs, for an overall retention rate of

Repair Method	Votes	Preference (%)
Gemini full-image prompt	16	4.0
OpenCV inpainting	0	0.0
Gemini patch + hard paste	10	2.5
Ours: patch + soft blend	**374**	**93.5**

Table 6: Method preferences from 20 participants over 20 image-repair cases, yielding 400 selections in total.

Item	Value
Initial candidate cases	220
Final retained cases	100
Overall retention rate	45.5%
Final Base / Edited images	100 / 100
Final question–answer pairs	400
Retained Edited images receiving repair	≈ 21
Final cases verified by two reviewers	100
Estimated Base-image output cost	$22.22
Estimated Edited-image output cost	$22.22
Estimated initial image-output subtotal	$44.44

Table 7: Context construction statistics and estimated initial image-output costs. Each case comprises one Base–Edited pair and four question–answer pairs. Costs assume one 2K output per Base image and per initial Edited image for all 220 candidates, at approximately $0.101 per output. The subtotal covers image outputs only, excluding localized repair, additional attempts, input tokens, text outputs, specification generation, and VLM filtering.

45.5%. This rate reflects the combined construction and curation process rather than the acceptance rate of automated filtering alone. Approximately 21 Edited images in the retained subset underwent localized repair. All 100 final Base–Edited pairs were verified by two reviewers.

Human Curation Effort. For the Context subset, pressure screening reduced the initial pool of 220 candidates to 147 candidates for human review, reducing the number requiring manual inspection by 33.2%. Both reviewers examined all 147 candidates. Following discussion, 47 cases were excluded because of unsuitable images, defects judged impractical to repair, or redundancy. The remaining 100 cases formed the final subset.

The review interface displays the images and associated annotations together and provides controls for retaining or rejecting each case. Based on the reviewers' practical estimates, inspecting a candidate and recording an initial decision typically took less than five seconds. This estimate does not include subsequent discussion or image repair.

Among the retained cases, 21 underwent localized repair, with one repair attempt per case. The interface allows a reviewer to mark the affected region and enter a repair instruction. These manual operations typically took less than 15 seconds per case. The repair tool typically returned a result in approximately five seconds under normal network conditions, although this waiting time depended on the Gemini API and network availability. Repaired candidates were inspected before final acceptance. These durations are experience-based estimates rather than systematically timed measurements. They describe individual review and repair operations, rather than the total elapsed time of the curation process.

Scope of construction effort. Constructing targeted stress tests requires more than collecting images and annotating answers. Candidates must satisfy the specified visual conditions, support valid questions and reference answers, and challenge the models being tested. An image found through manual search may meet the visual requirements yet be readily solved by the model. Comparing construction effort solely by the number of annotated images or time spent reviewing them therefore overlooks the effort required to obtain valid, challenging cases. A meaningful efficiency comparison must account for these requirements under a common task specification. Since these issues cannot be quantified, we have provided a sufficient basis for comparison.

Subset	Metric	Gemini 3.5 Flash	GPT-5.4	Claude 4.6	Kimi-k2.6	Qwen 3.5 27B	Grok-4.3
Context	**All4 (P)** ↑	0	1	**10**	7	3	4
	Edit2 ↑	0	2	**13**	9	7	6
	Q1 Base+ ↑	97	**99**	72	85	93	97
	Q2 Base- ↑	82	67	**94**	84	85	62
	Q3 Edit- ↑	0	3	**33**	17	10	7
	Q4 Edit+ ↑	86	82	52	58	68	**88**
Texture	**All4 (P)** ↑	**52**	28	40	**52**	46	28
	Edit2 ↑	52	28	41	**54**	47	29
	Q1 Base+ ↑	99	**100**	98	98	99	99
	Q2 Base- ↑	**100**	99	**100**	99	**100**	**100**
	Q3 Edit- ↑	**81**	30	54	69	63	38
	Q4 Edit+ ↑	58	**82**	65	67	65	65
Attribute	**Count Acc. (P)** ↑	**26**	20	17	17	14	16
Language	**MCQ Acc. (P)** ↑	11	23	**58**	17	29	23
Overall	**Macro Avg.** ↑	22.3	18.0	**31.3**	23.3	23.0	17.8

Table 8: **Detailed results on SABRE-Prior.** Rows marked with (P) are the primary subset metrics used to compute the macro-average. For Context Prior and Texture, All4 requires all four Base–Edited probes to be correct, while Edit2 requires both Edited-image probes to be correct. Q1 and Q2 respectively test the expected and counterfactual evidence in the Base image; Q3 and Q4 test the same evidence after editing, with expected answers *yes/no/no/yes*. Attribute reports normalized exact-match counting accuracy, and Language Elicitation reports four-option multiple-choice accuracy. All values are percentages. Bold values indicate the highest score in each row.

Image-Output Cost Estimate. The Gemini-based image pipeline used `gemini-3.1-flash-image-preview` with a 2K resolution setting and a 3:2 aspect ratio. The final stored images have a resolution of 2528×1696 pixels. Using the published Standard price of approximately \$0.101 per 2K image output, one Base generation and one initial edit for each of the 220 candidates correspond to 440 image outputs, or approximately \$44.44. This is a price-based estimate for the specified outputs, not a reconstruction of the complete historical API bill. The number of repaired images does not determine the number of repair attempts, so repair expenditure is not inferred from the approximately 21 repaired images.

Human Review and Repair Workflow. The curation platform presents each Base–Edited pair side by side, together with the intended transformation, review-location cues, questions, reference answers, and filtering-model predictions. Preparing these materials in advance allows reviewers to focus on the relevant visual evidence and record their decisions within the same interface.
When an Edited image requires correction, the reviewer marks the affected region and supplies a short repair instruction. The integrated tool generates a replacement patch and blends it into the image for reinspection. Reviewers can accept the result or request further correction without switching to a separate editing application or manually compositing the patch. We describe these operations qualitatively because active review time and API waiting time were not systematically recorded.

E ADDITIONAL RESULTS

Table 8 localizes the failures within the paired-image subsets. Base-image accuracy is generally high, whereas performance drops after editing. Indicating that models may recognize the inserted target while still reporting the replaced source. These results motivate the strict All4 metric over individual-probe accuracy.

E.1 INDEPENDENT HUMAN EVALUATION

We conduct an independent human evaluation with two participants who were not involved in sample construction or curation. Participants answer the questions using the evaluation images and, where applicable, the provided answer options. Reference answers and curation information, including review-location cues and intended transformations, are not provided. We report the mean accuracy of the two participants for each subset.

Subset	Human accuracy (%)
Context	93.0
Texture	90.0
Attribute	100.0
Language Elicitation	100.0
Macro-average	95.8

Table 9: Independent human evaluation on SABRE-Prior. Each subset score is the mean accuracy of two participants; the overall score is the macro-average across the four subsets.

Filter	N	Gemini	GPT	Sonnet
None	102	17.65	30.39	9.80
Gemini	84	—	25.00	8.33
GPT	71	11.27	—	8.45
Sonnet	92	16.30	29.35	—

Table 10: Accuracy (%) on the shared candidate pool and the subsets selected by each Filtering VLM. N is the number of samples in each set. Selection and scoring reuse the same responses; diagonal entries are therefore zero by construction and are omitted.

As shown in Table 9, the participants achieve high accuracy across all four subsets. These results provide evidence that the tasks are generally answerable without access to the annotations used during curation.

E.2 Ablation Study of the Filtering VLM

We test how the choice of Filtering VLM affects sample selection and cross-model performance on a shared cable-tracing task. Each image asks a model to follow a cable from a specified lettered endpoint to its numbered endpoint. The candidate pool contains 102 samples The additional samples cover 8, 12, 16, 20, and 24 cables, with varied endpoint layouts and crossing patterns. Reference answers follow the visible cable connections rather than the intended mappings in the generation prompts.

We compare Gemini 3.5 Flash, GPT-5.4, and Claude Sonnet 4.6 as Filtering VLMs. All three models receive the same images and questions. Images are encoded as JPEGs at their native resolution, and responses use temperature zero and a 256-token output limit. For this controlled comparison, each filter selects the candidates it answers incorrectly. We report the size of each selected set and the accuracy of the other models on that set, alongside accuracy on the full candidate pool.

As shown in Table 10, the three filters select 84, 71, and 92 samples, respectively. Their pairwise Jaccard overlaps range from 66.33% to 77.78%, indicating shared but nonidentical failure sets. On each selected set, the other two models achieve accuracies between 8.33% and 29.35%, below their corresponding accuracies on the full pool. For example, GPT-5.4 scores 25.00% on the Gemini-selected set, compared with 30.39% on the full pool; Gemini scores 11.27% on the GPT-selected set, compared with 17.65% on the full pool. Thus, filter choice changes which samples are selected, while the selected samples remain difficult for models other than the filter itself. This analysis concerns filter selection on the shared cable-tracing pool and does not isolate the effects of the specification-design or image-generation models.

F Procedural Extensions for Counting and Spatial Reasoning

Counting and spatial reasoning tasks require precise control over object quantities and arrangements. In constructing the additional stress tests, we observed that image generation models could produce scenes whose object counts or spatial layouts did not match the sample specifications, particularly when many repeated elements were present. Reference answers derived from those specifications therefore required verification against the generated images. Dense scenes made this verification more labor-intensive and complicated the construction of test sets with controlled difficulty levels.

To address this construction issue, we introduce procedural extensions for counting and spatial reasoning. Each extension defines a scene programmatically, renders the corresponding image, and

Image Source	Correct / Total	Accuracy
Generated (Attribute)	26 / 100	26%
Real-Image Control	6 / 20	30%

Table 11: **Real-image control.** Exact-match counting accuracy of Gemini 3.5 Flash on generated and real-image Attribute samples.

derives reference answers from the underlying scene specification. This construction allows task parameters to be varied systematically and supports evaluations across defined difficulty levels. It complements the image-generation workflow: the original workflow supports naturalistic scenes and varied visual content, while procedural rendering provides explicit control over the quantities and arrangements required by these tasks. The following subsections describe the task definitions, construction procedures, evaluation protocols, and results.

F.1 Real-Image Control

We examine whether the low accuracy observed on Attribute also occurs when the edited images are constructed from real photographs. We create 20 control cases using real photographs as the source images[1] and apply the same editing and evaluation procedures as in the Attribute subset. As shown in Table 11, Gemini 3.5 Flash achieves 30% accuracy on these cases, compared with 26% on the generated-source cases. The low accuracy on both sets indicates that the observed difficulty extends to cases constructed from real photographs. This control complements the generated-source evaluation and illustrates the pipeline's applicability to both real and generated source images.

F.2 Counting

We construct counting scenes by placing three-dimensional object instances in a fixed kitchen environment and rendering them from a fixed viewpoint. Each image is paired with a question asking for the number of instances belonging to a specified category. The scene specification determines both the rendered objects and the reference answer. The construction procedure consists of asset preparation, inventory sampling, constrained placement, visibility verification, and rendering.

Environment and object assets. We use a kitchen environment assembled from RoboCasa assets (Nasiriany et al., 2026), with a fixed room layout, camera, and lighting configuration. The counting region comprises three support surfaces: the foreground kitchen island and the countertops to the left and right of the sink. Placement regions are defined within these surfaces, with margins around their boundaries and exclusions for the sink opening and faucet. Fixed kitchen fixtures provide the scene context but are excluded from the counting task.

The object library contains 150 models, of which 147 are enabled for scene construction. These models form 27 counting categories, including fruits, vegetables, tableware, cookware, and packaged food. Multiple models may belong to the same category, allowing instances within a category to vary in appearance. Categories with overlapping visual identities are grouped where appropriate. For example, food and beverage cans share one counting category, and the question wording explicitly includes both.

Each asset is imported using its visual mesh and source textures. During preparation, its coordinate frame is normalized, its base is aligned with zero height, and its mesh is uniformly scaled using a category-specific size convention. The aspect ratio is preserved. The multi-counter configuration uses a diameter of approximately 22 cm for plates and an overall length of approximately 38 cm for pans; other objects retain their prepared dimensions. These are modeling conventions rather than measurements of the source objects. Object sizes do not change with scene density or target status.

Inventory sampling. We use four scene sizes, $N \in \{20, 50, 100, 150\}$, where N denotes the total number of countable object instances rather than the reference answer. For each candidate scene, a seeded random generator first samples a target category uniformly from the available categories. The target count K is then sampled uniformly from the integers in

$$[\lceil 0.10N \rceil , \lfloor 0.75N \rfloor] .$$

The corresponding intervals are 2–15, 5–37, 10–75, and 15–112 for the four scene sizes.

[1] All source photographs were obtained from Pexels: `https://www.pexels.com/license/`.

The inventory contains exactly K target instances and $N - K$ distractors. Each target instance independently draws a model from the target category. Each distractor first draws a non-target category uniformly and then draws a model from that category. Models may be reused within a scene. The inventory is shuffled, and each instance receives a unique identifier. Target instances receive no special highlighting or target-dependent scaling. This sampling defines the candidate distribution; the retained scenes are additionally conditioned on the placement and visibility checks described below.

Constrained placement. The placement procedure assigns each instance a support surface, a planar position, and a rotation about the vertical axis. Positions are computed directly under geometric constraints; we do not use a physics rollout to settle objects on the countertops. Instances are processed in descending order of their horizontal bounding-box area so that larger objects are placed before smaller ones.

For each placement attempt, the procedure samples one of the three support surfaces, using weights of 0.78 for the island, 0.12 for the left countertop, and 0.10 for the right countertop. It then samples a position uniformly within the selected region and a rotation uniformly over a full turn. These weights govern proposal sampling, rather than imposing fixed proportions on the final arrangement.

A proposed placement must keep the object's horizontal bounding box within the allowed support region and its projected bounds within the camera view. Physical separation is checked using conservative polygonal footprints derived from the object meshes. A separating-axis test requires a gap of at least 8 mm between the footprints of different instances. This prevents physical interpenetration while allowing their projections to overlap in the rendered image.

The procedure also applies a preliminary occlusion constraint. It projects each object's three-dimensional bounding box into the image plane and uses camera depth to estimate which objects are in front. For each object, the accumulated overlap from foreground projected boxes must not exceed 35% of its own projected-box area. A new placement must satisfy this constraint for both the new object and previously placed objects. This is a placement heuristic: projected boxes include empty space, and their overlap is not an exact measure of visible object area.

Each object receives at most 2,500 placement proposals per layout attempt, and the procedure permits up to four layout attempts for a candidate inventory. If it cannot place every instance, the candidate is rejected. Objects are neither removed nor reduced in size to make a failed inventory fit.

Mesh-based support and visibility checks. After placement, the scene is instantiated in Blender using the actual object and kitchen meshes. We verify that every object's base is aligned with the support height and use downward ray casts to check support against the imported countertop mesh. We also require the transformed object bounds to lie within the camera view.

Visibility is checked separately from the projected-box heuristic. For each object, we sample a 32×32 grid within its projected bounding box and cast rays from the camera. We first intersect these rays with the object in isolation, retaining the samples that hit its mesh. We then cast the corresponding rays through the complete scene. A sample is considered visible if the first intersection belongs to the same object. Its estimated visible fraction is

$$\widehat{v}_i = \frac{n_i^{\text{visible}}}{n_i^{\text{isolated}}},$$

where n_i^{isolated} is the number of samples that hit object i in isolation and n_i^{visible} is the number that remain visible in the complete scene.

Every instance must have at least 20 isolated-object samples and satisfy $\widehat{v}_i \geq 0.5$. The check allows partial occlusion while rejecting instances that are largely hidden by other objects or kitchen fixtures. It estimates geometric visibility from sampled rays; it does not establish that every retained instance is semantically unambiguous to a human observer.

Rendering. Scenes that pass these checks are rendered with Blender Cycles at a resolution of 1920×1600 pixels, using 32 samples per pixel and denoising. We use a fixed perspective camera, environment illumination, and two area lights. The kitchen layout, camera, and lighting remain unchanged across scene sizes. Variation comes from the sampled object inventory, asset choices, positions, and orientations.

Object appearance uses the source textures with material settings adapted for the renderer. Shading and texture-coordinate corrections preserve mesh geometry and instance transforms. The output is a PNG image without instance identifiers, target markers, or answer annotations.

Question and reference-answer construction. Each scene specification stores the category, asset identifier, position, orientation, scale, and support surface of every instance. The reference answer is



Figure 12: **Examples of procedurally generated counting scenes.** From left to right, the scenes contain 20, 50, 100, and 150 countable objects, respectively.

Total Objects	Questions	Correct	Acc. (%)
20	50	18	36.0
50	50	0	0.0
100	50	0	0.0
150	50	0	0.0

Table 12: Exact-match counting accuracy of Qwen 3.5 27B on the procedurally generated kitchen scenes.

recomputed from this inventory:

$$y = \sum_{i=1}^{N} \mathbb{1}[c_i = c_{\text{target}}],$$

where c_i is the category of instance i and c_{target} is the queried category. The implementation checks that the inventory contains exactly N instances and that the recomputed answer matches the intended target count.

Questions explicitly define the counting region and exclude fixed fixtures. For example, an apple-counting question reads:

> Count the loose objects on the kitchen island and the countertops beside the sink. How many apples are there? Do not count fixed kitchen fixtures. Return the integer count.

The evaluation prompt additionally requests a single integer without an object list or explanation. Only the rendered image and question are supplied to the evaluated model. Scene inventories, coordinates, seeds, and reference answers are retained separately.

Test-set assembly. We retain 50 scenes at each of the four scene sizes, giving 200 images and 200 counting questions. Before construction, each sample slot is assigned an ordered list of 16 distinct candidate seeds. Candidates are processed in that order, and the first scene passing the placement, support, framing, and visibility checks is retained. Rejected candidates and their failure reasons are recorded.

All construction checks occur before model evaluation. Retained scenes are not replaced according to whether the evaluated model answers correctly. For reproducibility, we preserve the candidate seeds, scene specifications, construction code snapshot, rendering reports, and image hashes.

Evaluation setup. We evaluate Qwen 3.5 27B on the 200 image–question pairs, with 50 examples at each scene size. The model receives only the rendered image and the counting question and is instructed to return a single integer. We use a temperature of 0, disable thinking, and set the maximum output length to 8192 tokens. A prediction is correct only if the extracted integer exactly matches the reference count. Examples are shown in 12

Results. As shown in Table 12, Qwen 3.5 27B achieves only 36.0% accuracy even in the smallest scenes, which contain 20 objects. Accuracy falls to 0% in the larger settings containing 50, 100, and 150 objects. These results do not imply that the model lacks counting ability altogether, but they indicate limitations in its ability to count precisely in more complex visual scenes.

F.3 SPATIAL REASONING

We construct spatial reasoning scenes using fully occupied three-dimensional grids of colored geometric objects. Each scene supports two questions about the same target: identifying its color and shape from its coordinates, and recovering its coordinates from its appearance. The scene specification defines the object attributes, coordinate system, target selection, and reference answers.

Grid and object construction. We use grids of size $n \times n \times n$, where $n \in \{2, 4, 7, 10\}$. Every cell contains exactly one object, giving 8, 64, 343, and 1000 objects per scene, respectively. Interior objects remain present even when occluded from the camera. Grid coordinates are integers from 1 to n along each axis.

Objects are drawn from four shapes—sphere, cube, cone, and cylinder—and eight colors: green, blue, red, yellow, orange, purple, white, and pink. Shapes are constructed procedurally as three-dimensional meshes and assigned materials from a fixed palette. Objects have a nominal extent of 0.22 scene units, with a cell spacing of 0.44 units. For a cell (x, y, z), the horizontal center and support height are

$$p_x = 0.44\left(x - \frac{n+1}{2}\right), \qquad p_y = 0.44\left(y - \frac{n+1}{2}\right), \qquad h_z = 0.08 + 0.44(z-1).$$

The object is positioned with its base at h_z. These dimensions remain unchanged across grid sizes.

Support structure and coordinate frame. The grid is represented by a physical support structure with a solid base, corner posts, and open upper levels. Thin rails and crossed support strips hold the objects at their assigned cells without introducing broad opaque shelves between levels. Horizontal grid lines and vertical boundaries on the front and right faces indicate cell spacing.

Coordinate labels mark only the endpoints of each axis: X1 and Xn, Y1 and Yn, and Z1 and Zn. Individual objects and intermediate cells are not numbered. The question text states the coordinate convention: X follows the X ruler, Y follows the Y ruler from front to back, and Z indexes shelf levels from bottom to top. The same convention is used in scene construction and reference-answer generation.

Target and appearance sampling. Each scene is generated from a recorded random seed. Questions are restricted to objects on the top face and the two exterior side faces facing the camera. Under our camera configuration, these cells satisfy

$$x = n \ \vee \ y = 1 \ \vee \ z = n.$$

A target is sampled uniformly from this union before rendering. Interior objects remain in the scene but are not selected as question targets.

The target receives a randomly selected color–shape pair. Its color is reserved exclusively for that instance, while all other objects use the remaining seven colors. The target can therefore be referred to unambiguously by its color. Both questions concern this same exterior object: one asks for its color and shape given its coordinates, and the other asks for its coordinates given its unique color.

Rendering. We use a fixed viewing direction from the positive-X, negative-Y, and positive-Z side of the grid. Camera distance is adjusted according to grid size to include the apparatus and coordinate labels within the frame. For a given grid size, framing does not depend on the sampled target. The scenes use a neutral off-white background, consistent object materials, and lighting scaled with the apparatus.

The rendering pipeline first constructs the meshes in Blender and bakes their surface appearance using Cycles with 96 samples. It then samples the mesh surfaces to construct a Gaussian-splat representation. Gaussian centers are placed on the mesh surfaces, their orientations follow the surface normals, and their colors are sampled from the baked appearance. The representation is constructed directly from the meshes without a learned reconstruction stage. Objects, support structures, grid lines, and axis labels are included in the rendered representation.

The model-facing image is a PNG exported from the Gaussian renderer at the prescribed camera pose. Image resolutions are 1600×1000 for the 2^3 and 4^3 grids, 2240×1400 for the 7^3 grid, and 3200×2000 for the 10^3 grid.

Construction checks. The scene is checked against its deterministic specification to verify full occupancy, object attributes, target uniqueness, and reference answers. Mesh-based checks examine support contacts, and framing checks verify that the coordinate labels lie within the image.

Ray-based visibility measurements and image-space diagnostics are retained for inspection. In this experiment, their thresholds are advisory and do not filter the preselected exterior targets. Scenes are

Grid Size	Images	Coordinate	Color + Shape	Overall
$2 \times 2 \times 2$	50	32/50 (64%)	33/50 (66%)	65/100 (65%)
$4 \times 4 \times 4$	50	10/50 (20%)	14/50 (28%)	24/100 (24%)
$7 \times 7 \times 7$	50	0/50 (0%)	3/50 (6%)	3/100 (3%)
$10 \times 10 \times 10$	50	0/50 (0%)	1/50 (2%)	1/100 (1%)

Table 13: Exact-match accuracy of Qwen 3.5 27B on the procedural spatial reasoning tasks. Coordinate denotes entity-to-coordinate questions; Color + Shape denotes coordinate-to-entity questions. Each grid size contains 50 images and 100 questions. Scores use all scheduled questions, with failed or incomplete requests receiving no credit.

not resampled according to these measurements or model accuracy. The scene specification, rendered asset, and exported image are linked by hashes to verify that each question is evaluated against its corresponding image.

Question and reference-answer construction. Each image has two questions concerning the same target. The first, *coordinate-to-entity*, supplies the target's coordinates and asks for its color and shape. For example:

> What are the color and shape of the object at (2, 1, 1)? Return only two English words in color-shape order.

The reference answer is the color–shape pair stored at that cell.
The second, *entity-to-coordinate*, identifies the target by its unique color and asks for its grid coordinates:

> What are the (x, y, z) coordinates of the unique yellow object? Return only an integer array.

Its reference answer is the target's coordinate tuple. The prompt explicitly states that exactly one object has the specified color in the entire grid, including hidden cells. Both question types explain the coordinate convention and state that every cell is occupied.
The two questions are submitted as independent requests with no shared conversation history. Each request contains only the image and its own question. The scene specification, reference answer, and answer to the paired question are not supplied to the model.

Test-set assembly. We generate 50 scenes for each grid size, yielding 200 images and 400 questions. Each grid size uses 50 distinct random seeds. The complete seed schedule and paired questions are recorded before rendering begins. Scenes and targets are not replaced based on model correctness. We retain the scene specifications, rendering reports, exported images, and raw model responses for each scheduled example.

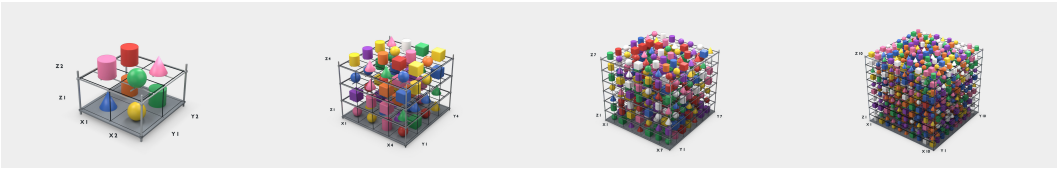


Figure 13: **Examples of procedurally generated spatial reasoning scenes.** From left to right, the grids have dimensions $2 \times 2 \times 2$, $4 \times 4 \times 4$, $7 \times 7 \times 7$, and $10 \times 10 \times 10$, containing 8, 64, 343, and 1000 objects, respectively.

Evaluation setup. We evaluate Qwen 3.5 27B with a temperature of 0, thinking disabled, and a maximum output length of 512 tokens. All questions concern the preselected target on the top face or either of the two camera-facing side faces; no question requires identifying a hidden interior object. For coordinate-to-entity questions, both the predicted color and shape must match the reference answer. For entity-to-coordinate questions, all three coordinates must match exactly. No partial credit or coordinate-distance tolerance is used. The answer parser accepts the requested short format or a supported explicit final-answer format following an explanation.

We report accuracy over all scheduled questions: 50 questions per task at each grid size. Failed or incomplete requests receive no credit. Overall accuracy is computed over the 100 questions across both tasks at each grid size. Example scenes are shown in Figure 13.

Results. As shown in Table 13, Qwen 3.5 27B achieves an overall accuracy of 65% on the smallest $2 \times 2 \times 2$ grids, with 64% accuracy on coordinate questions and 66% on color-and-shape questions. Overall accuracy decreases to 24%, 3%, and 1% as grid size increases to 4^3, 7^3, and 10^3, respectively. At the two largest sizes, no coordinate question receives a correct answer, while color-and-shape accuracy falls to 6% and 2%. These results do not imply that the model lacks spatial reasoning ability altogether, but they indicate limitations in accurately identifying objects and recovering their coordinates in more complex three-dimensional scenes.