RESEARCH MAY 19, 2026

THE PATH TO LARGE SCALE DENSE VIDEO CAPTIONING

By Jade Choghari, Agustin Sansone, Nicolas Pasqualis, Conrado Mader, Aleks Tiupikov, Mouli Sivapurapu

[Video: see online version]

TL;DR:

- Tested dozens of approaches for dense video captioning of hand-object manipulation.
- How you represent the video matters more than fancy prompting.
- Hand-cropped past/present/future collages gave the strongest Flash baseline; raw video performed worst.
- Temporal tokens, visual overlays, smarter sampling, sequential context, and CoT on smaller models mostly added noise.
- Best config (Pro + hand-collage + one targeted Stack rule + lightweight reasoning) reached 63.7% rubric acceptable rate.
- Remaining failures are mostly visual grounding errors: the model understands the action but names the wrong object or destination.

At Scale, we are building the largest corpus of real-world robot manipulation data in the world, with more than 1,000 hours of new demonstrations landing on our platform every day from data factories, homes, and industrial sites. Every episode has to be annotated with precise, action-level captions before it becomes useful training signal. We do this with a mix of human labelers and automated vision-language model pipelines.

In this phase, each episode is already segmented into short subgoals. The task is to generate, for each segment, a precise caption that names the action, the objects, and the destination. Examples include “Stack the green gelatin packet on top of the violet packet” or “Scoop the ice cream from the tub into the cone.” This captioning stage is the foundation for a larger dense video captioning pipeline that will also include temporal clipping and timestamp localization in future phases. These captions support downstream robot training workflows, from fine-tuning foundation models to building reward functions, which means a wrong verb or misidentified object can propagate into everything downstream.

Getting VLMs to caption robot/human manipulation reliably is **harder than it might seem**. Academic benchmarks often give models credit when the generated caption is roughly semantically similar to the reference. Production annotation needs to be correct on *action verb*, *object identity*, *destination*, and *temporal ordering*, simultaneously, on noisy real-world video, at scale. So we built an internal benchmark and ran **ablation experiments** over several weeks. We tested several commonly used temporal-understanding strategies, including temporal-position encoding, fps and motion-guided sampling, marker-based visual prompts, structured reasoning prompts, and context-conditioned caption generation across neighboring events.

Some of it worked, but a lot of it didn't. The failures, however, were instructive. What looks elegant in a paper often becomes brittle in production, and the gap between academic robotics benchmarks and production annotation data shows up as a **fundamentally different distribution of failures, rather than a worse score on the same one.**

What follows documents the surprising failures, the incremental wins, and the simple approach that beats every fancy technique we tried. We also work through **why** several methods from prior work and several recommendations from Apollo's systematic Video-LMM study (Zohar et al., 2024), failed to generalize to our setting, and what that implies for the field.

The Task: Dense Manipulation Captioning

What We're Annotating

[Video: see online version]

Our dataset spans hundreds of unique environments and includes from both robot and human demonstrations, with episodes ranging from 2 to 20 minutes. Each episode is **pre-segmented into subgoals** using human-annotated timestamps. The task: given a video segment `[t_start, t_end]` corresponding to one subgoal, generate a precise natural-language caption describing the action performed.

[Video: see online version]

Captions follow a strict schema:

- **Action verb** from a fixed vocabulary of 30+ verbs including pick up, put, stack, insert, open, close, retrieve, scoop, place, and others.
- **Object** (the primary manipulandum): full name including color/material identifiers
- **Destination or source** (for transfer verbs): where the object ends up

A correct caption looks like: "Stack the green gelatin packet on top of the violet packet." A caption fails on a wrong verb ("Put the green gelatin packet on top of the violet packet."), a wrong object ("Stack the orange packet on top of the violet packet."), or a wrong destination ("Stack the green packet on the table").

This annotation is much **harder** than general video QA. The model must distinguish between Put (on a surface) and Stack (on top of a specific object), between visually similar objects differing only in color, and between

actions that look nearly identical at the frame level. Timestamps are also imperfect: they can be half a second to a second off, which sometimes puts two actions in the same window and leaves the model to figure out which one to caption.

Our Evaluation Methodology

We ran experiments on a held-out benchmark of **~2.3k subgoals** from diverse manipulation tasks. Each prediction was graded by an LLM judge using a default temperature of 0. A caption scores 1.0 only if it gets all three fields right, and 0.0 otherwise:

- **Acceptable** (score 1.0): correct verb, correct object, correct destination/source
- **Not acceptable** (score 0.0): any of the above wrong
- **Egregious** (score 0.0): hallucinated objects, completely wrong scene description

We report **rubric acceptable rate** as our primary metric. We also collect a text similarity score (weighted combination of object, color, action, intent match) for diagnostics and failure analysis, but we optimize for rubric acceptable rate because it directly maps to data quality.

Phase 1: What Do We Feed the Model?

Our earliest experiments explored the most basic question: **how do we show the model a manipulation subgoal?** We tested three representations:

1. **Raw video clip** -> trimmed to the subgoal window, passed as a video file to Gemini's video API
2. **Keyframes only** -> a grid of N frames sampled from the subgoal window, no video

Temporal collage -> three separate collages: a grid from the *previous* subgoal (context), the *current* subgoal, and the *next* subgoal (future context). Plus **hand-cropped** close-ups of the current subgoal.

The results were stark, but highly model-dependent. For the models we tested, simply passing in video was worse than converting the video into structured image inputs. Video-only performed worst (28.7% rubric score), despite containing all the temporal information. Raw frame grids improved substantially (49.2%) but still missed critical context.

Our best-performing format combined past/future collages with explicit present-frame sequences and tight hand-crop keyframes, reaching 61.0%, a 32-point improvement over video-only. One likely reason is that video inputs are internally converted into image tokens using model-specific sampling strategies, giving less control over which details are preserved.

Why Video Hurt

This finding runs counter to the intuition that "more information is better." The raw video experiment (28.7% rubric acceptable rate) was a worst-case, video-only with no collage context. But we later ran a more controlled test: taking our best hand-collage baseline and *adding* video on top. The result was still negative: **2.52pp** (58.5% from 61.0%).

Four distinct failure mechanisms emerged:

Temporal overreach (17 action-flip failures). Sub-goal clips include the approach phase before the action and a release phase after, while static keyframes naturally land at the action peak. Full video lets the model watch every frame and anchor on the wrong temporal moment:

Here, **GT** means **ground truth**: the human-written reference action label from our production annotation pipeline, used as the evaluation target.

```
GT:   "Put the silver knife into the utensil organizer"
Collage: "Put the knife into the cutlery tray."         [PASS -- saw placement frame]
Video:  "Pick up the knife from the table."           [FAIL -- anchored on approach frame]
```

```
GT:   "Retrieve the brown bottle from the ziploc bag"
Collage: "Remove the bottle from the plastic bag."      [PASS]
Video:  "Put the bottle into the plastic bag."         [FAIL -- described return frame]
```

Over-identification (101 of 161 regressions - 63%). Video gives the model richer detail. It now sees brand logos, exact colors, and secondary objects clearly, producing more confident but wrong names. The GT labels are often deliberately generic ("brown pack"), but the model, seeing the video clearly, names the specific brand:

```
GT:   "Retrieve the brown pack from the white grocery bag"
Collage: "Retrieve a packet of chocolate gelatin from the plastic bag."  [PASS -- generic enough]
Video:  "Put the chocolate D'Gari packet into the blue plastic bag."    [FAIL -- brand detail + direction flip]
```

Where Video Genuinely Helps

For directional actions where motion is the only discriminator, video solves cases that static frames cannot:

[Table: see online version]

Video's signal is real for Pour and Turn. The problem is the surrounding temporal noise outweighs it 2:1 (161 regressions vs 103 improvements). Tighter clip trimming (removing the approach and release phases before uploading) is the most promising path to making video net-positive.

The hand-cropped collage addresses this by:

1. **Cropping to the hand region** focuses the model's attention on the manipulation area, where critical object interactions happen

2. **Including temporal neighbors** through past and future collages provide action context without the resolution penalty of video
3. **Showing multiple frames** from each window captures approach, grasp, and release phases

The collage approach was originally developed by our annotation engineers as a human-labeler aid. It turns out VLMs benefit from the same framing that helps humans.

This hand-collage baseline became our reference point for all subsequent ablations: **61.0% rubric acceptable rate** on ~770 subgoals from a diverse set of manipulation tasks.

Temporal Fusion: The Interleaved Token Paradox

Recent video-LMM work suggests that explicit temporal structure can help models reason over video. Apollo (Zohar et al., 2024) finds that adding text or learned separator tokens between video tokens from different frames or clips improves token integration, while Qwen3-VL (Bai et al., 2025) uses explicit textual timestamp alignment for more precise temporal grounding in video.

But we found that this benefit did not transfer to prompt-based image interleaving in frontier APIs. Adding temporal progress tags such as `[t=0.0]` or `[t=0.33]` to Gemini Flash decreased rubric acceptable rate from 61.0% to 59.1%.

Our hypothesis: temporal markers help when they are part of the model's video-specific architecture or training recipe, where they organize dense spatiotemporal token streams. In prompt-based image interleaving, the same markers may behave more like noise: they compete for attention or push the prompt away from the model's pretrained multimodal formatting priors.

The technique helps on bigger models, but hurt on smaller ones. If you aim for cheap, cost-effective solutions, we recommend against using it.

What "Temporal Normalized Progression" Actually Means

To understand why this happens, consider what the model must do with a [t=0.33] token. It must:

1. Parse the numeric value
2. Relate it to the other frames' time positions
3. Infer what *phase* of the action is depicted
4. Use that phase information to disambiguate the action verb

This is a multi-step reasoning chain that requires the model to hold and integrate information across multiple frames. Cheap flash-class models handle single-frame queries well but struggle with cross-frame temporal integration when the integration is explicit (token-form) rather than implicit (spatial layout in a collage).

The collage format itself provides implicit temporal ordering, frames are arranged left-to-right in time, which models have learned to read from web data. Explicit temporal tokens add a second, potentially conflicting encoding that confuses smaller models.

Why Temporal Progress Fails in Production Specifically

There is a second reason normalized temporal progress underperforms in our setting that is specific to production (as opposed to academic benchmarks): **action boundary noise**.

Temporal progress tokens assume the clip boundaries are meaningful. In an ideal clip, `[t=0.0]` shows the start of the interaction, and `[t=1.0]` shows the action's endpoint.

Our production clips are not always that clean. They are segmented by humans watching video at 1× speed, so the boundaries are approximate. Many clips include extra frames before or after the actual manipulation:

- **Pre-contact frames at `[t=0.0]`:** the hand is moving toward the object but has not touched it yet.
- **Post-release frames at `[t=1.0]`:** the hand or arm is retracting after the object has already been released or settled.

This makes the boundary tokens noisy. The model is supposed to use `[t=0.0]` and `[t=1.0]` as anchors for action direction, but in production those anchors can point to the wrong moment. For example, labeling an empty-hand retraction frame as `[t=1.0]` tells the model that the endpoint of the action is the hand moving away, not the actual release.

Sampling Methods

The Premise We Tested

One of the most persistent ideas in video understanding for robotics is that **smarter keyframe selection** should improve action recognition. The reasoning is intuitive: uniform sampling might hit redundant frames, or miss the critical grasp moment during fast motion. A smarter sampler would focus on the informative frames.

We tested six keyframe sampling strategies, all using 6 frames from the current subgoal window:

- **Uniform**: evenly spaced, the naive baseline
- **DP (Dynamic Programming)**: longest-path selection maximizing visual diversity
- **DP-FPS**: FPS-constrained DP variant
- **Middle**: uniform over the middle 60% of the window, trimming the approach/release phases
- **Late-last**: skip first 20%, pin the final frame to the exact last frame (to capture placement state)
- **MG-RGB**: motion-guided sampling using per-frame L1 difference
- **MG-Feature**: motion-guided sampling using learned feature distances

After five distinct sampling strategies, including motion-guided approaches with learned feature representations, not a single one beat naive uniform sampling. The information content for this task is not concentrated in "interesting" frames; it is distributed across the action.

Why Better Sampling Doesn't Help

There are three reasons smarter sampling fails to improve manipulation captioning:

1. The critical signal is often at the boundary, not the peak motion moment. The object's identity is best visible at the very moment it is picked up or placed, these are often the lowest-motion frames. Motion-guided methods preferentially select *high-change* frames, which in manipulation often correspond to the demonstrator's arm swinging through space (not carrying an object), not the critical grasp or place moments.

2. The collage format already compensates for bad sampling. Our baseline includes both past and future collages, giving the model 18 frames total (6 per collage × 3 temporal windows). Within this rich context, the exact 6 frames selected from the current window matter much less. The model can triangulate from the neighboring windows.

3. Object identity doesn't require the "right" frame. For our primary failure mode (wrong object), what matters is seeing enough of the scene to identify which specific item is being manipulated. Any frame showing the hand near the object accomplishes this. Uniform sampling reliably delivers one such frame; sophisticated methods don't do meaningfully better.

Mark-Based Visual Prompting: When Visual Grounding Hurts

The Hypothesis

Recent LMM work introduced **Set-of-Marks (SoM)** prompting to improve visual grounding by overlaying numbered marks on candidate image regions and referring to targets by mark (Yang et al., 2023). Related robotics manipulation work, for example MOKA, adapts mark-based visual prompting for action or affordance grounding by marking candidate points or regions where the robot might act.

Our dominant failure mode is **wrong-target** (37% of errors, the model uses the correct verb but names the wrong object or destination). SoM seemed tailor-made to fix this. The following clip illustrates the exact kind of error we were trying to solve, the model says "Put the book on the table" when the correct action is "Stack the blue book on top of the green book":

We implemented two **mark-based prompting** variants:

- Hand-keypoint / trajectory overlay: We overlaid MediaPipe hand keypoints and connecting lines on each keyframe, optionally preserving motion traces across frames. This was intended to expose fingertip-to-object contact cues, hand identity, and wrist trajectory.
- **Object-label overlay:** a two-pass Gemini Robotics-ER pipeline that detects objects, returns normalized `(x, y)` coordinates with labels, and renders those labels onto each keyframe before captioning.

Mark-based visual prompting did not fix wrong-target errors. Instead, it introduced new errors by adding noisy, incorrect, or ambiguous visual cues that the model sometimes anchored on instead of the underlying visual evidence.

Why the Overlays Backfired

The failure came from a cascade of reliability issues:

Detector errors: In our evaluation, the Gemini Robotics-ER detection pass labeled objects correctly on roughly 70% of frames. The remaining errors propagated downstream. When the detector labeled a blue packet as "green packet," the captioning model often inherited that mistake. The two-pass pipeline introduced an additional source of error rather than removing the original one.

Overlay interference: Both variants added visual marks to the images. For manipulation involving small objects, such as packets, cable connectors, and earphone components, these marks often obscured the exact regions needed for identification. A label or keypoint trace placed over the contact region between two similarly colored objects degraded the visual evidence rather than augmenting it.

There may be a path to making mark-based prompting work for manipulation: use more reliable 3D perception, incorporate depth, or apply labels only when detection confidence exceeds a high threshold. But in our setting, RGB-only ego video at production scale, the overhead and error rate of reliable marking were prohibitive.

The Broader Lesson

Mark-based visual prompting is a meta-strategy: it replaces uncertain implicit visual grounding with explicit visual or symbolic cues, betting that the inserted cues are more reliable than the model's own visual processing. When those cues are reliable, this can help. When they are not, as in our case, the method adds a new failure mode without fixing the original one.

This generalizes beyond mark-based prompting: any technique that inserts an intermediate processing step can compound errors. A two-pass system helps only when the intermediate representation is more reliable than the model behavior it replaces. For production annotation at scale, simpler pipelines may be more robust.

Sequential Context Injection: A Related Failure

We tested **sequential context injection**, an autoregressive approach where subgoal 0 is predicted without context, subgoal 1 receives the model's own prediction for subgoal 0, and subgoal 2 receives predictions for 0 and 1. This implements within-episode coherence: each prediction conditions on prior predictions.

Result: **57.8%**, a 3.2pp drop from the baseline. The breakdown by subgoal index reveals the mechanism:

[Table: see online version]

Sub-goal 0 is essentially unchanged. All damage comes from the context-injected subgoals. The obvious explanation, “pred_0 was wrong, so it poisoned sub-goal 1,” accounts for only **33% of the idx=1 regressions**. The other **67% happened when `pred_0` was correct**:

```
INJECTED (correct): "Put the headphones on the table."
GT for sub-goal 1:  "Put the headphones on the table"
pred:         "Put the headphones on the table."   [PASS]
pred:         "Return to Home"                     [FAIL]
 Model concluded: "Headphones just placed  task finished  RTH"

INJECTED (correct): "Screw the lightbulb into the socket."
GT for sub-goal 1:  "Screw the new light bulb into the socket"
pred:         "Insert the lightbulb into the socket."   [PASS]
pred:         "Turn on the lightbulb."                  [FAIL]
 Model reasoned: "Bulb inserted  next step is turn on", predicted the future action
```

The model shifted from “describe what you see” to “what comes next in a logical task sequence.” Its language priors about task progression overrode the visual evidence. This distraction mechanism, **even correct context causing regressions**, is particularly insidious because it is not fixable by improving prediction accuracy. It requires a different prompt framing: rather than “the previous action was X,” the prompt must say “the previous action X is already **finished** and must NOT appear in your annotation.”

The distraction is amplified by the repetitive structure of robot teleoperation data. Tasks repeat 8 to 20 times per recording session (e.g., “Put the can into the bag” ’ “Adjust the bag” ’ “Put the can into the bag” ’ …). The model’s training prior says “why would you *Put* the same item twice?” so when context shows a *Put*, it predicts *Pick Up* or *RTH* instead of another *Put*.

This echoes findings from multi-turn LLM research: conditioning on the model’s own outputs in a chain increases coherence (the model stays consistent with itself) but not accuracy (the model stays consistently wrong). The distraction failure is subtler, even oracle context causes regressions because the model’s task-completion priors override its visual grounding.

The Lesson

Chain-of-thought (CoT), sequential context, and mini-CoT (a lightweight variant where the model first predicts the past, present, and future frames before writing the final action caption) all share a common failure mode on smaller models: on large, capable models, the reasoning chain catches mistakes and on smaller models, the reasoning chain *creates* mistakes.

For production annotation with Flash-class models, the pragmatic answer is: don’t ask for reasoning. Provide a clean, structured prompt with the relevant context (hand collages, disambiguation rules) and let the model answer directly.

Model Capacity: The Clearest Signal

The Model Ladder

After several experiments on prompt engineering, sampling strategies, and visual augmentation, we ran the simplest possible ablation: swap the model while holding everything else constant.

We compared five models at equivalent configuration (hand-collage, uniform sampling, no CoT, and no visual overlays):

[Table: see online version]

Model Capacity vs Score -- The Clearest Signal in Dozens of Experiments

A +2.7pp gain from upgrading the model dwarfs every single prompt-engineering technique we tried. The best single-technique improvement (stack rule: +0.4pp on Flash) is an order of magnitude smaller.

If you only do one thing to improve VLM annotation quality, upgrade the model. Prompt engineering on a smaller model cannot close the gap with a larger model; it can only extract the last few percentage points once you are on the right model tier.

Targeted Rules: The Exception That Proves the Rule

One prompt engineering technique did provide a reliable, consistent benefit: **targeted disambiguation rules**. The Stack rule (appended to the system prompt) produced +0.4pp on Flash and +0.2pp on Pro:

```
SPECIAL CASE -- Stack:
When one item is being placed ON TOP OF another specific item
(not a flat table/surface), use "Stack", not "Put".
Example: "Stack the green gelatin packet on top of the violet packet."
```

This works because it addresses a *specific, well-characterized* failure mode (Stack/Put confusion) with a *minimal, targeted* intervention. The rule only fires on Stack-relevant scenarios; it cannot degrade performance on non-Stack actions.

The Scoop rule (destination vs source naming) showed a more complicated story. Within Scoop pairs it worked well (+21 improvements, 7 regressions, net +14 Scoop pairs). But the directive “name where the item ends up” leaked into how the model resolves directional ambiguity in other verbs: adding the Scoop rule to the Stack rule (exp22) caused **42 Put regressions** and 19 Pick regressions, the model started second-guessing destination/source in unrelated actions. Net over the Stack-rule-only baseline: +5 pairs — barely measurable.

Lesson: rules must be verb-scoped. A rule written in directional terms bleeds into every verb that involves motion direction. The Stack rule (“when placing ON TOP OF another item, use Stack not Put”) is narrowly visual, it only fires on spatial-layering observations. The Scoop rule as initially written was too semantic and rewired the model’s direction-of-motion prior globally.

How the Failure Mass Actually Decomposes

We analyzed the 890 failing pairs from the **Flash + Stack + Scoop rules** condition. The failures split almost exactly into thirds:

[Table: see online version]

The 332 **correct-verb-wrong-object** failures are the most informative: the model knows *what* is happening but cannot identify *which specific item* is being manipulated. Object identification is the primary failure signal, 82% of all failing pairs involve wrong object, destination, or source, whereas action (verb) errors account for only 63%.

Scoop is the sharpest illustration: **100% of Scoop failures are correct-verb-wrong-destination**, the model always knows it is a scooping action, but consistently names the wrong vessel. Verb rules address the 37% action-recognition failures and leave the 37% object-grounding failures untouched.

This decomposition sets the theoretical ceiling for different intervention types:

- **Perfect prompt rules only**: ~75% (eliminates wrong-verb third, grounding failures remain)
- **Prompt rules + better object grounding** (bounding boxes, higher-res crops, depth): ~85%
- **True ceiling** (best possible with 6 static RGB frames): ~85-90% (minus the ~10–15% structurally ambiguous clips where motion direction cannot be determined from static frames alone)

The 25.9% **always-fail floor**, pairs that fail across every experiment regardless of prompt, is confirmed to be structural. It includes: visually identical objects (honey bottle vs white lid container), actions only determinable from motion direction (Retrieve vs Put from a bag), and temporal contamination from adjacent sub-goals that bleed into the clip window.

The Best Configuration

Our best result is a **63.7% rubric score**, combined:

1. Gemini 3.1 Pro (model capacity)
2. Hand-collage representation (past/present/future frames, hand-cropped)
3. Stack disambiguation rule (targeted fix for dominant verb error)
4. High reasoning effort (Pro thinks before answering)
5. Mini-CoT (structured PREVIOUS/NEXT/CURRENT reasoning)

Notably absent: visual overlays, interleaved tokens, fancy sampling, sequential context. The winning combination is the simplest capable representation plus the most capable model plus one targeted rule.

Ablation Results

All experiments use the Flash hand-collage baseline unless otherwise stated.

What Worked

[Table: see online version]

What Didn't Work (and Hurt)

[Table: see online version]

The Meta-Pattern

Looking across dozens of experiments, a single pattern explains most of the results:

"Complexity taxes smaller models; simpler prompts extract more of their real capability. Larger models can absorb and use complexity productively."

Every technique that added reasoning overhead, intermediate steps, or additional signals *hurt Flash* and *helped Pro*. The practical implication for production:

- **If you have budget for Pro-class models**: use mini-CoT, add temporal grounding, invest in reasoning effort
- **If you need Flash-class economics**: use the cleanest possible representation (hand-collage), add targeted rules for known failure modes, avoid any reasoning overhead

Limitations

This study should be interpreted as a directional engineering evaluation rather than a rigorous statistical analysis.

- Results are reported from single evaluation runs on a fixed 2.3k-pair benchmark without confidence intervals. Small differences (<1–2pp) should not be overinterpreted, and the leaderboard is better viewed as tiers than a strict ranking.
- We do not report latency or inference cost, so higher scores should be considered alongside production tradeoffs.
- Results are scoped to Gemini-family models on one internal manipulation-video benchmark and may not generalize to other distributions or VLM families.

What's Next

This was Phase 1: a single model, no agentic tooling, no multi-step correction. We deliberately stayed in the regime of "one model does everything" to understand the fundamental limits of this approach.

Phase 1 limits are now clear. The 38% failure mass decomposes into three approximately equal thirds:

- **~37% wrong-verb failures**, fixable with better prompt rules, context injection, or middle-clip trimming
- **~37% correct-verb-wrong-object failures**, require better visual grounding (bounding boxes, higher-res crops, depth sensors), prompt engineering cannot touch these
- **~26% egregious failures**, require a stronger model (RTH, directionally ambiguous actions)

The theoretical performance ceilings: ~75% with perfect prompt rules only; ~85% with prompt rules plus visual grounding; ~85–90% true ceiling accounting for structurally ambiguous clips (action direction not determinable

from static RGB frames). The 25.9% always-fail floor is structural, confirmed unchanged across every experiment in Phase 1.

Phase 2 priorities follow directly: (1) visual grounding of the manipulated object in each frame (bounding box overlay from a reliable detector), (2) sub-goal context injection with a negative-framing prompt that tells the model what the *previous* action was and that it is now finished, (3) video input with middle-clip trimming for directional verbs (Pour, Turn, Retrieve).

The findings here generalize beyond manipulation captioning. For any VLM annotation task in production:

- **Representation beats sampling**: how you show frames matters more than which frames you show
- **Rules beat reasoning**: for specific, known failure modes, a targeted rule is more reliable than asking the model to reason its way out
- **Model capacity is the biggest lever**: invest in the right model tier before optimizing prompts
- **Two-pass systems multiply errors**: marker-based visual prompting, object detection pipelines, and autoregressive conditioning on prior predictions (sequential context) all cascade their errors

Acknowledgments

This work was conducted within Scale's Physical AI team. We thank the annotation team, whose domain expertise shaped the evaluation rubric and error taxonomy, and the operators whose data made this possible.

Interested in collaboration or using Scale's Physical AI data engine? Contact us at physical-ai@scale.com.