

Augmenting Vision-Language Agents with Spatial Grounding for Gameplay Automation

Martin Mitrović^{1,2}, Niels Justesen¹, Sebastian Risi², Christoffer Holmgård¹

¹modl.ai

²IT University of Copenhagen

martin@modl.ai, niels@modl.ai, sebr@itu.dk, christoffer@modl.ai

Abstract

Building an autonomous agent capable of playing contemporary video games poses a significant technical challenge, requiring mastery of visual perception, real-time decision-making, long-term memory, and adaptive planning within complex, dynamic environments. While Vision-Language Models (VLMs) demonstrate strong reasoning capabilities, they struggle with spatial grounding, often failing to accurately localize gameplay objects or UI components. This gap between reasoning and perception is a key limitation when VLMs are used as game-playing agents. To address this, we present a framework that augments visual inputs to mitigate spatial grounding errors. These take the form of bounding boxes around game elements, generated using two methods: Segment Identification (SI), which segments each frame and labels the pieces with a VLM online, and a lightweight YOLO detector fine-tuned offline on gameplay footage. We evaluate four VLMs across three casual mobile games unaugmented and under each augmentation, over five trials per configuration. Several models benefit substantially from augmentation, gaining the capability to complete challenges that were out of reach from raw screenshots alone. The benefit is conditional rather than uniform. It improves the models that cannot localize interactive elements on their own, but can interfere with one that already performs well. Our experiments show that the same augmentation substantially improves one model’s efficiency while degrading another’s. Moreover, on localization-limited tasks, augmentation lets a smaller open model match a much larger proprietary one, though this parity does not extend to challenges bottlenecked by planning. These findings position object-level grounding as a promising but situational tool for VLM-based game agents, clarifying when it helps, hurts, or has no measurable effect.

Introduction

Intelligent agents powered by frontier models such as GPT (OpenAI 2025) and Gemini (Google 2026) increasingly automate repetitive digital tasks. Building a generalist agent for video games, however, is constrained by the extreme variability of sensory inputs and the difficulty of mapping them to consistent action policies across very different games. While Vision-Language Models (VLMs) have shown significant potential for gameplay (Wang et al. 2024;

Tan et al. 2025; SIMA Team 2024), their inability to precisely locate objects within a frame remains a major failure point (Rahmanzadehgervi et al. 2024). Since most games rely fundamentally on spatial awareness, this is one of the key bottlenecks for capable VLM-based agents operating from raw pixels. We therefore investigate how augmenting visual input improves action precision, focusing on spatial execution errors: cases where the agent formulates the right action but fails to target the correct screen coordinates.

Our approach supplies bounding boxes around game elements as auxiliary input, generated in two ways. The first, Segment Identification (SI), extracts elements from the screenshot with segmentation algorithms and labels them with the VLM at inference time; the second trains a lightweight, fine-tuned YOLO detector (Jegham et al. 2025) on gameplay footage. These two can also chain: SI’s labeled outputs can serve as the YOLO detector’s training data in place of recorded video, so the system needs no manual annotation and trains its detectors offline. In this work we evaluate them as two separate augmentation options at different operating points rather than as a single pipeline. Our aim is not to detect exhaustively but to locate enough key elements to sustain progress through the game. Decoupling perception from decision-making lets the grounding improve and adapt to new games independently of the reasoning model.

For evaluation, we benchmark agents across permutations of selected VLMs and visual augmentation strategies using three Android mobile game environments: Tavern Rumble (Evrac Studio 2024), Idle Slayer (Leban 2024), and Matchington Mansion (Magic Tavern, Inc. 2024). This selection covers distinct game mechanics, allowing us to evaluate the applicability of our methods. To account for the inherent stochasticity in both game environments and VLM outputs, each configuration is evaluated across five independent trials. Our experiments show that agents autonomously play the selected games and that visual augmentation can be decisive: cold-start SI pushes some models that barely progress from raw screenshots to complete every challenge in one of our games. The effect, however, is not uniform. Grounding helps a model that cannot localize what a task requires, but can interfere with one already competent at it. The augmentation that rescues a weaker model on a spatial task can degrade a stronger model’s efficiency on another, and on visually static interfaces a fine-tuned YOLO detector outper-

forms SI. The sign of the effect is predicted neither by model size nor by the augmentation alone.

The contributions of this paper are: (1) two complementary methods for generating bounding-box augmentations for VLM-based game agents, requiring no manual annotation; (2) a five-trial empirical evaluation across four VLMs and three mobile games with distinct mechanics; (3) the central finding that the benefit of object-level grounding is conditional and model-dependent; and (4) an action-level error analysis that explains this finding mechanistically.

Related Work

Agents that play games from visual input have matured quickly, trading API access for raw pixels. Voyager (Wang et al. 2024) uses an LLM as a central controller that plans and acts by generating executable programs and building a reusable skill library; it performs strongly in Minecraft (Mojang Studios 2011) but relies on the game’s API for internal state. Cradle (Tan et al. 2025) removes this dependency with a vision-based approach, playing complex titles such as Red Dead Redemption 2 from raw pixels and emulated input, though it leans on dedicated auxiliary detectors to compensate for the poor spatial reasoning of VLMs (Rahmanzadehgervi et al. 2024).

A different paradigm invests in the agent model itself: SIMA (SIMA Team 2024) and SIMA2 (SIMA Team 2025) follow instructions across commercial 3D games from vision alone, and NitroGen (Magne et al. 2026) trains a control policy by behavior cloning at scale. We instead leave the reasoning VLM untouched and add only spatial grounding through an external perception module.

VLMs struggle even with simple visual tasks (Rahmanzadehgervi et al. 2024), hallucinating absent objects, missing present ones, and reasoning poorly about spatial relationships; object counting alone remains largely unsolved (Vo et al. 2026). Dedicated detectors offer text-prompted localization, from transformer-based Grounding DINO (Liu et al. 2024) to frameworks such as YOLOE (Wang et al. 2025), but refining prompts to isolate only the desired objects is difficult. Despite their spatial grounding limitations, we favor VLMs for their semantic understanding, which allows them to identify objects of interest based on contextual meaning rather than requiring precisely worded prompts. The combination of dedicated detectors with VLM reasoning has emerged as an active research direction (Sapkota and Karkee 2026), typically using detectors as a perception front-end whose outputs inform VLM-based reasoning. Our work follows this pattern but automates the detector training pipeline rather than relying on separately curated detection datasets. A complementary line of work improves grounding through visual prompting rather than a separate detector: Set-of-Mark (Yang et al. 2023) overlays segment masks and identifiers onto the image so that a frozen VLM can refer to regions directly, a strategy our Segment Identification extends by attaching discovered semantic labels and reusing them through a persistent database. Closer to our application domain, agents for graphical and mobile interfaces treat element localization as the central capability: SeeClick (Cheng

et al. 2024) identifies grounding accuracy as the main bottleneck for visual GUI agents, and Ferret-UI (You et al. 2024) fine-tunes a multimodal model for grounded mobile-UI understanding. These approaches obtain grounding by training or fine-tuning the VLM itself, whereas we leave the VLM frozen and supply grounding through an external, automatically constructed perception module.

At the time of writing, Gemini 3.1 Pro (Google 2026) demonstrates the strongest object detection performance on video game screenshots among the VLMs we evaluated; its bounding boxes are highly accurate, though some objects remain undetected. However, Gemini 3.1 Pro exhibits significant latency during object detection (20–50 seconds per request), rendering it unsuitable for a responsive game agent. In contrast, YOLO variants are well-suited for such applications due to their rapid inference speeds. Despite the widespread availability of pretrained YOLO models, their performance in gaming environments is poor. This is because standard architectures are trained on photographic datasets such as COCO (Lin et al. 2014), Pascal VOC (Everingham et al. 2010), or OpenImages (Kuznetsova et al. 2020), which do not capture the visual properties of video games. Although specialized models like OmniParser (Lu et al. 2024) trained on UI elements exist, the diverse and artistic nature of game interfaces often limits the effectiveness of models trained on conventional web or application datasets.

Regarding the fine-tuning of vision models, the primary challenge remains the acquisition of high-quality annotated data, a process researchers are actively exploring how to automate. SAM2Auto (Rocky and Wu 2025) utilizes the segmentation capabilities of SAM2 (Ravi et al. 2025) combined with the labeling precision of YOLOE (Wang et al. 2025) to automate the annotation pipeline. Similarly, recent research (Griffin et al. 2025) uses VLMs as automated annotation tools, generating labels and boxes to train smaller, more efficient downstream models.

Agent Architecture

A simple yet effective architecture underpins the playing agent used in our experiments. Core components are inspired by Voyager (Wang et al. 2024) and Cradle (Tan et al. 2025) and include:

- **Reasoning** – the central module powered by VLMs. This module determines the next action the agent should take based on the current state, desired goal, and action history. It provides the action that the executor needs to perform.
- **Perception** – the sensory module providing the visual state of the environment. This is the only module modified in our experiments.
- **Executor** – maps high-level actions to concrete environment commands and carries them out.

The agent operates in a continuous loop: observe the environment, select an appropriate action from available skills, and execute. Data obtained from the perception module consists solely of screen recordings of the currently ac-

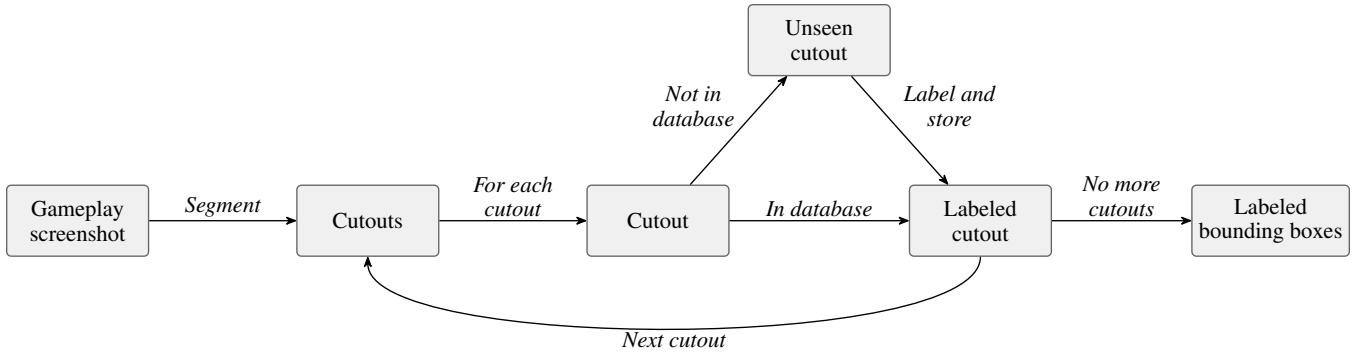


Figure 1: The Segment Identification pipeline: segments are labeled online by a VLM, with caching for recurring elements.



Figure 2: Results of Segment Identification on a Tavern Rumble (Evrac Studio 2024) game screen.

tive game and bounding-box augmentations. As in previous work (Wang et al. 2024; Tan et al. 2025), agent skills are stored as Python code and presented to the agent in the format `skill_name(argument1, argument2, ...)`. Our implementation uses a compact skill set consisting of `tap(x, y)`, `swipe(x1, y1, x2, y2)`, `wait(duration)`, and `type(text)`. Beyond selecting the appropriate skill, the reasoning module must determine the correct arguments.

Prompt Structure

The VLM receives a structured prompt with the following components: (1) a brief description of the current game and its objectives; (2) the current screenshot, with overlay annotations when augmentation is enabled; (3) the list of available skills with parameter signatures; (4) a history of recent actions and their outcomes; and (5) when augmentation is active, a textual list of detected bounding boxes as `(label, x, y, width, height)` tuples. The VLM is instructed to respond with a single skill call. For example, given a detection list including `(start_button, 200, 750, 80, 100)`, the VLM is expected to return `tap(x=240, y=800)` – the coordinate at the approximate center of the detected element. The same prompt template is used across all games, with only the game description changing.

Perception Augmentations

The spatial grounding limitations of VLMs motivate offloading perception to specialized vision models. While simple techniques like Optical Character Recognition (OCR) are

effective for text-heavy interfaces, they fall short in environments that rely on non-textual cues. To achieve a truly scalable solution, an agent must be able to reliably identify diverse gameplay elements, such as enemies, menu icons, or progress bars. In this section, we present two augmentation methods that combine segmentation algorithms with the zero-shot classification capabilities of VLMs to identify and localize relevant elements across varied gaming environments.

Segment Identification

Segment Identification (SI), illustrated in Figure 1, uses segmentation algorithms to detect individual elements on the screen and VLMs to label them. Models such as SAM (Kirillov et al. 2023) and RF-DETR (Robinson et al. 2026) are highly effective at segmenting complex images and extracting meaningful parts, even in video game footage. Depending on the segmentation model’s quality, extracted segments may represent meaningful elements or noise. The algorithm removes the background from each segment and retains only the elements of interest, which we call cutouts. These cutouts are then sent to the labeling model. If it cannot provide a meaningful label, the cutout is flagged as noise. Removing the background helps the labeling model focus on the exact details of the specific cutout and eliminates contextual bias (Vo et al. 2026). Since cutouts are usually small and classification is inherently a simpler task than object detection, processing time is relatively fast. To avoid labeling the same element multiple times, a database matches each new cutout against previously labeled ones and returns the stored label when one is similar enough, rather than reprocessing it.

Figure 2 shows how SI works in the Tavern Rumble (Evrac Studio 2024) video game. Object detection misses many elements of interest, but it provides enough useful information for the agent to advance gameplay. The primary limitation of this augmentation is significant computational overhead, limiting its applicability to turn-based games that await player input or idle games that progress without it.

Fine-tuned YOLO Pipeline

YOLO models (Jegham et al. 2025) offer fast, reliable inference but lack zero-shot capability: they detect only objects

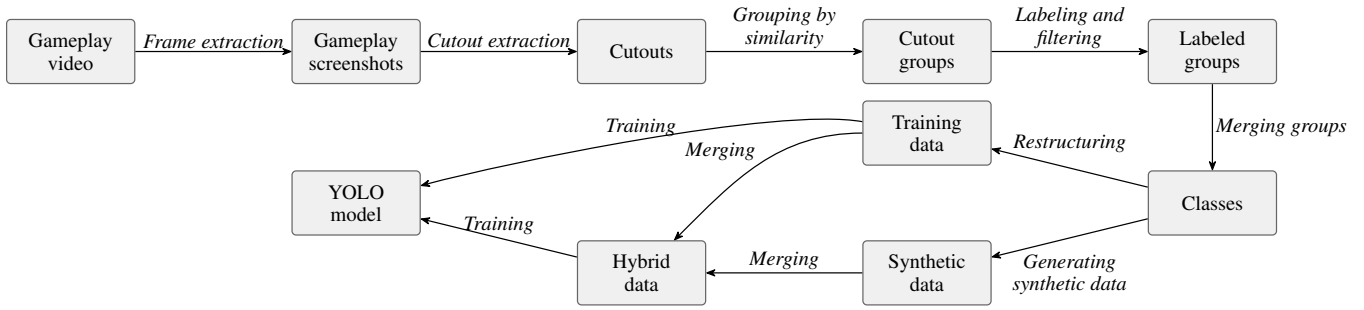


Figure 3: YOLO fine-tuning: an offline pipeline that turns gameplay video into labeled classes and trains a detector.

present in their training data and so require game-specific training. The labeled data for this can come either from SI outputs gathered during exploration or, as in our pipeline (Figure 3), from gameplay videos.

Because few VLMs accept video directly, we sample frames from the gameplay video at regular intervals, trading redundancy against missed events. The extracted frames are processed by segmentation models, producing a list of individual element cutouts. The total count of cutouts can grow significantly based on the sampling frequency and segmentation logic. However, most cutouts are redundant, and labeling each one individually would be highly inefficient. Instead, cutouts are grouped by visual similarity and only a few representative elements from each group are sent to the labeling model. Due to noise in the segmentation and grouping algorithms, cutout groups are rarely perfectly distinct. Multiple different cutouts can end up in the same group, while similar cutouts may be placed in separate groups. In our experiments, we use a high similarity threshold to ensure that all cutouts within a group are visually consistent. Labeling is conducted per group rather than per cutout, reducing the number of labeling requests by over an order of magnitude. Once labeled, cutout groups undergo a merging process that considers both a lower similarity threshold and the assigned labels. If the labels are similar (e.g., *green.button* and *action.button.green*) or identical (e.g., *blue.button* and *blue.button.1*) and the similarity exceeds the threshold, the groups are merged under a unifying label. Omitting this step can result in identical elements receiving multiple different labels.

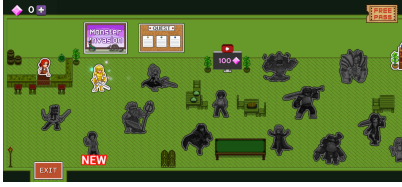
In the obtained data, not all classes are equally represented, making it more difficult for the model to generalize. To balance the dataset, synthetic data is generated in addition to the real data. Synthetic images are generated by placing labeled cutouts at random positions on a blank image, ensuring no overlap between elements. This produces additional training samples with known bounding-box annotations, helping to balance underrepresented classes. The resulting model cannot compete with the accuracy of YOLO models trained on properly curated datasets, but it is sufficient to assist VLMs in playing video games. Unlike Segment Identification, this augmentation requires the entire fine-tuning pipeline to complete before the agent can begin playing. Additionally, if the pipeline encounters errors or the game changes significantly, a new training run is required.

Experiment

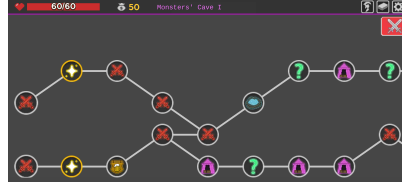
Games used in our experiments are Tavern Rumble (Evrac Studio 2024), Matchington Mansion (Magic Tavern, Inc. 2024), and Idle Slayer (Leban 2024). We chose mobile games for their simple interaction model, as all actions can be performed through taps and swipes. While computer and console games can also be played through our framework, they often require more complex combinations of simultaneous inputs, such as moving the character and camera at the same time. Our selection aims to cover different game mechanics while maintaining simple gameplay complexity.

Tavern Rumble is a roguelike deck-building game where players battle enemies by placing cards on a small grid. Selecting the appropriate cards and their placement demands both tactical selection and spatial reasoning. Matchington Mansion, in contrast, is a casual match-3 puzzle game that merges puzzle solving and decorating mechanics. Properly assessing the board state and selecting the optimal piece to interact with is crucial. Finally, Idle Slayer features a passive play style where monsters are automatically slain and currency is generated over time. Players can purchase upgrades such as sword, shield, and armor improvements, which increase the generated income. In the early stages, the player must collect coins by jumping at the right moment. While our agent cannot react quickly enough to catch every coin, the high spawn rate means that even delayed actions frequently result in successful collection.

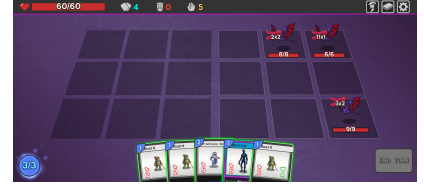
For each game we define three challenges that exercise different demands, illustrated in Figure 4. Three key challenges are encountered while playing Tavern Rumble (Evrac Studio 2024). The first challenge is navigating the game’s startup sequence (Figure 4a), which requires selecting an available character from the hub. Initially the only option is the knight in golden armor, and since no play or start button is visible before tapping the correct hero, the agent may struggle to identify the interaction target. The bartender is also tappable but only opens game settings, which can mislead the agent. We count this challenge as completed when the agent taps the knight and begins a run. After a character is selected, the agent must choose the next event from the run map (Figure 4b): initially the available events are the leftmost nodes, while in later stages they are the nodes adjacent to the current position, indicated by green outline. This challenge is completed when the agent selects any reach-



(a) Tavern Rumble: hub



(b) Tavern Rumble: map



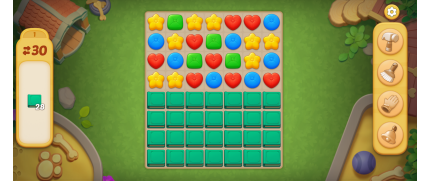
(c) Tavern Rumble: fight



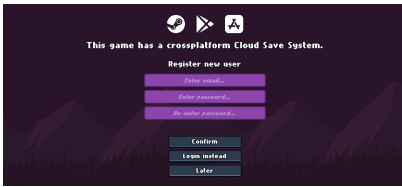
(d) Matchington Mansion: intro



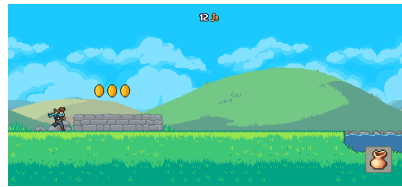
(e) Matchington Mansion: tutorial



(f) Matchington Mansion: 1st level



(g) Idle Slayer: login



(h) Idle Slayer: open panel



(i) Idle Slayer: purchase

Figure 4: The three challenges in each game. Top row: Tavern Rumble (Evrac Studio 2024); middle: Matchington Mansion (Magic Tavern, Inc. 2024); bottom: Idle Slayer (Leban 2024).

able event node. Finally, once a fighting event begins (Figure 4c), play requires placing cards across different rows to defend against monsters and damage the opponent. Since the first turn cannot be ended without playing at least one card, the agent must engage the placement mechanic. If the agent places at least one card successfully, this challenge is completed.

The challenges in Matchington Mansion (Magic Tavern, Inc. 2024) follow a progressive structure. The agent must first choose a repair option for the opening household task (Figure 4d) before the game transitions into a match-3 puzzle; the intro challenge is completed when a repair option is selected. Three guided tutorial puzzles follow, each solvable in two to three matches (Figure 4e), and we count the tutorial challenge as completed when all three are cleared. The agent then advances to the first standard level, which introduces the core mechanic of swapping adjacent tiles to satisfy a level objective (Figure 4f). Completing the first level successfully solves the last challenge.

For Idle Slayer all the challenges are simple localization tasks and only include pressing the right button. The first one is placed on the login/title screen and the agent needs to dismiss the login in order to reach gameplay (Figure 4g). In the game, opening the upgrade panel represents a second challenge (Figure 4h). The final challenge is purchasing an upgrade inside upgrade panel (Figure 4i).

Configurations

To assess the influence of visual augmentations, each agent uses a fixed VLM with one of the following augmentation configurations:

- **No augmentation** – An unmodified screenshot of the game.
- **Fine-tuned YOLO** – Bounding boxes obtained from a YOLO model fine-tuned on 30 minutes of gameplay video are provided to the agent in text format. Additionally, the screenshot is annotated with pink bounding boxes and corresponding text labels.
- **Segment Identification** – Bounding boxes from the SI pipeline are provided in text and overlay format. To preserve real-time responsiveness during gameplay, only segments already present in the database are labeled. Other segments remain unlabeled until they are processed and added to the database asynchronously. In all experiments reported here, the database begins empty. The database can alternatively be warm-started from a previous session or from the labeled cutouts produced during the YOLO pipeline’s data collection, but we leave it empty to isolate the discovery mechanism.

Combining both augmentations provides no additional benefit, as SI detects as many or more elements than YOLO but with higher latency. The two are not competing conditions but different operating points: SI discovers elements online for exploration, while a fine-tuned YOLO detector

Model	Mode	Tavern Rumble				Matchington				Idle Slayer			
		Hub	Map	Fight	Reach	Intro	Tut	Lvl	Reach	Log	Pan	Pur	C/s
GPT-5.2	SC	2	2	2	1	5	0	0	0	5	5	5	5.1
	YOLO	2	2	2	1	4	3	0	1	5	5	5	6.0
	SI	5	5	5	1	2	2	0	1	5	5	5	41.2
Gemini 3.1 Pro	SC	5	5	5	5	5	5	5	3	5	5	5	112.9
	YOLO	5	5	5	5	5	5	5	3	5	5	5	156.2
	SI	5	5	5	4	5	5	5	4	5	3	2	13.0
Kimi K2.5	SC	2	1	1	1	2	0	0	0	5	5	1	0.1
	YOLO	2	2	2	1	2	0	0	0	5	3	1	0.0
	SI	5	5	1	1	3	0	0	0	5	5	4	3.1
Qwen3.5 ¹	SC	1	0	0	0	2	0	0	0	5	5	5	3.5
	YOLO	1	1	1	1	5	5	0	1	5	5	5	12.9
	SI	5	5	5	4	3	0	0	0	5	4	4	0.9

¹ Refers to the Qwen3.5-397B-A17B variant.

Table 1: Challenge completion across the three games (out of five trials), with Idle Slayer’s coins-per-second efficiency. Reach is the deepest progression any single trial achieved. Tavern Rumble: 0 = opening not cleared, 1 = opening cleared, 2–5 = successive roguelike events beyond it. Matchington Mansion: highest level reached (0 = never reached the first level; tutorial progress is given by the Intro and Tut columns). SC = screenshot, YOLO = fine-tuned YOLO, SI = cold-start Segment Identification.

adds no token cost on repetitive tasks that introduce no new visual elements.

Models and Trials

We evaluate proprietary state-of-the-art VLMs including GPT-5.2 (OpenAI 2025) and Gemini 3.1 Pro (Google 2026), as well as open-source models Qwen3.5 (Qwen Team 2026) and Kimi K2.5 (Kimi Team 2026). Although Gemini 3.1 Pro is the most capable detector we tested, labeling is a far simpler task than detection, since it names a single cropped element rather than localizing many, and it runs on every segment at high volume. We therefore label with Gemini 2.5 Flash (Google DeepMind 2025), a fast and inexpensive model that suffices for the short, single-word labels the pipeline requires. Our primary interests are assessing how far each model progresses in the game, how effective its actions are, and which augmentations are most beneficial. All runs are limited to 30 minutes and start from the beginning of the game. Since the early stages of each game are relatively simple, the most significant differences between models are expected to arise from their ability to select accurate coordinates for tapping and swiping.

Implementation details. At runtime, Segment Identification segments each frame with a stock FastSAM-s checkpoint (Zhao et al. 2023). SAM 2 (Ravi et al. 2025) gives higher-quality masks, but SI runs on every frame, so we trade a little quality for FastSAM’s much lower latency. The offline YOLO training pipeline has no per-frame constraint and instead segments with SAM 2 and clusters cutouts with a Perception Encoder (Bolya et al. 2025) embedding before labeling. Each cutout is labeled by Gemini 2.5 Flash (temperature 0.1, at most 200 output tokens, images capped at 128 px). A cutout is matched against the stored library by combining a perceptual-hash similarity with a bounding-box dimension ratio, and is recognized as a cache hit when the

returned match scores at least 0.7. Otherwise it is sent for labeling. The frame given to the reasoning model is down-scaled to 540 px wide while coordinates are requested in the native 2400×1080 device space, so the model extrapolates from a smaller image than it acts on. Remaining settings and thresholds are in the released configuration, alongside full prompts, the challenge rubric, and per-run data.¹

To account for stochasticity in both the games (e.g., randomized card draws in Tavern Rumble) and the VLMs themselves, each model-augmentation configuration is evaluated across five independent trials. For each configuration we report the number of trials (out of five) in which each challenge was completed. Reach in Tavern Rumble is the furthest roguelike event the agent reached in any single run; in Matchington Mansion it is the highest standard level reached, excluding the tutorial. For Idle Slayer we report coins-per-second as a continuous efficiency measure. Segment Identification runs report the final state of the image database (segment count and hit rate) as evidence of online discovery.

Results

In this section, we compare the agent across the four VLMs and three augmentation configurations, reporting per-game findings before analyzing the perception methods and failure modes across games.

Per-Game Findings

Tavern Rumble. Table 1 reveals the clearest positive effect of augmentation in our study. GPT-5.2 and Qwen3.5 are nearly incapable of progressing from raw screenshots: GPT-5.2 completes all three challenges in only 2 of 5 trials, and Qwen3.5 fails to reach the map in any trial. SI transforms both models, taking each to a perfect 5/5 on every

¹<https://github.com/modl-ai/visual-grounding>

challenge. Gemini 3.1 Pro, the strongest baseline, already completes all challenges from screenshots alone and neither benefits nor suffers from augmentation. YOLO augmentation provides little benefit here since the fine-tuned model failed to reliably detect the starting character, indicating the biggest flaw of this method. Kimi K2.5 follows the same pattern partially: cold-start SI lifts it from 2/5 and 1/5 on the hub and map challenges to 5/5 on both, matching the navigation gains of GPT-5.2 and Qwen3.5, though it clears the fight challenge in only one of five trials.

Matchington Mansion. This game (Table 1) exposes a capability ceiling that augmentation cannot overcome. Only Gemini 3.1 Pro completes the first standard level, and it does so in all five trials regardless of augmentation. The match-3 mechanic requires reasoning about which tile swap will clear an objective. This is a planning problem rather than the localization that bounding boxes address. Providing accurate object coordinates does not help a model that cannot determine which object to act on. No other model clears the first level in any configuration.

Augmentation effects here are mixed and, notably, run opposite to Tavern Rumble. For Qwen3.5, YOLO is the strongest configuration, clearing the tutorial in all five trials, while SI clears it in none. We attribute this reversal to the static nature of the puzzle board: a fine-tuned YOLO model produces stable, consistent detections on a fixed layout, whereas SI’s per-frame segmentation introduces more variance on a screen that changes little between turns. This suggests the better augmentation method depends on whether a game’s visual content is dynamic (favoring SI’s discovery) or static and repetitive (favoring a trained detector).

Idle Slayer. Idle Slayer is the clearest case of augmentation cutting both ways (Table 1). Its idle mechanic rewards sustained interaction, so the coins-per-second rate captures how effectively an agent acts over the full run, beyond binary completion, and it exposes effects that point in opposite directions across models. SI’s effect on the coin rate is sharply model-dependent. For GPT-5.2 and Kimi K2.5 it is strongly positive. It raises GPT-5.2 from 5.1 and 6.0 to 41.2 coins per second and Kimi K2.5 from 0.1 to 3.1. For Gemini 3.1 Pro, however, SI is actively harmful. The model generates 113 coins/s from raw screenshots and 156 with YOLO but only 13 with SI. Qwen3.5 regresses in the same direction, from 3.5 to 0.9 coins/s. Run-to-run variance is high (for example, 41.2 ± 30.2 for GPT-5.2 and 13.0 ± 27.4 for Gemini 3.1 Pro under SI, mean $\pm$ standard deviation over five trials), yet these two effects hold across all trials, and we release the per-run values.

Whether SI helps or hurts does not track how capable a model already is. It lifts GPT-5.2 and Kimi K2.5 yet sharply lowers Gemini 3.1 Pro and Qwen3.5. Gemini is the strongest player without it, at 113 and 156 coins/s, but SI cuts it to 13, below even GPT-5.2. The drop is not a speed artifact, since SI labels segments asynchronously and the agent waits only milliseconds per step, so the augmentation never slows its play loop. The pattern instead points to interference. For a model that already localizes and acts well on its own, the extra bounding-box channel seems to distract rather than help, and its repetition rises accordingly (examined in the Error

Game	Classes	P	R	F1	mAP ₅₀
Tavern Rumble	707	0.92	0.93	0.93	0.94
Matchington Mansion	3,197	0.91	0.69	0.78	0.72
Idle Slayer	623	0.92	0.92	0.92	0.94

Table 2: Fine-tuned YOLO11n detector quality per game, measured on each game’s held-out auto-labeled split (P, R, F1 at the default confidence threshold; mAP at IoU 0.50).

Analysis); for GPT-5.2, which cannot localize unaided, the same channel sharpens targeting and lifts its rate. The effect of grounding therefore depends on whether the model needs it, not on model size and not on the augmentation alone.

Perception Methods

Detector quality. Table 2 characterizes the three fine-tuned detectors, each a YOLO11n model fine-tuned from COCO-pretrained weights on the automatically labeled gameplay data. On Tavern Rumble and Idle Slayer they are strong, reaching mAP50 of 0.94 and F1 above 0.92. Matchington Mansion has far more classes (3,197) than the other two games (around 700), reflecting its many distinct tiles, boosters, and level-specific elements, and its recall falls to 0.69 (mAP50 0.72) because rare classes recur too rarely to be learned. These metrics are computed on each game’s auto-labeled validation split, so they measure fidelity to the segmentation-and-labeling signal rather than human ground truth, in line with the dependency noted in the Limitations. The large class vocabularies are themselves a consequence of the unsupervised labeling pipeline, which mints a new label for each visually distinct element it encounters, and the Matchington result shows how that degrades a detector when elements are too varied to recur. Training settings are listed in our public repository.

Cold-start discovery. SI builds its label database online, during play, requiring no prior data. To show this, every SI run started from an empty database. Table 3 reports the resulting state, averaged over the five trials per configuration. Runs accumulated thousands of labeled segments, up to 9,327 for Qwen3.5 in Tavern Rumble, with hit rates between 0.89 and 1.00. Because each database was populated entirely within a single 30-minute run, these hit rates show that SI discovers, labels, and reuses recurring screen elements on the fly rather than relying on any pre-existing store.

Computational cost. The benefit of Segment Identification is bought with a substantial increase in token consumption, summarized in Table 3. Relative to raw screenshots, SI inflates total token use by typically threefold to tenfold, while the fine-tuned YOLO detector leaves token use essentially unchanged. Almost all of SI’s overhead is segment-labeling traffic rather than gameplay reasoning: of the 5.23M tokens used in Qwen3.5’s SI run on Tavern Rumble, 4.08M came from labeling calls, while Qwen3.5’s own playing budget (1.15M) remained comparable to the 1.05M of its screenshot run. The size of the multiplier tracks how many distinct elements a game surfaces for labeling, ranging from a negligible increase for Qwen3.5 in Idle Slayer,

Game	Model	Tokens (M)			SI database	
		SC	YOLO	SI	Segs	Hit
Tavern Rumble	GPT-5.2	0.64	0.73	3.97	6,895	1.00
	Gemini 3.1 Pro	0.65	0.68	2.89	4,882	0.98
	Kimi K2.5	0.43	0.33	1.23	2,002	1.00
	Qwen3.5	1.05	0.93	5.23	9,327	0.99
Matchington	GPT-5.2	0.50	0.40	3.41	4,441	0.94
	Gemini 3.1 Pro	0.43	0.43	4.24	4,883	0.91
	Kimi K2.5	0.21	0.17	0.93	1,042	0.90
	Qwen3.5	0.73	0.65	2.47	3,011	0.98
Idle Slayer	GPT-5.2	1.10	1.16	4.94	6,345	0.89
	Gemini 3.1 Pro	1.44	1.32	3.79	4,882	0.95
	Kimi K2.5	0.34	0.25	1.68	2,196	0.97
	Qwen3.5	1.45	1.31	1.47	1,781	0.96

Table 3: Per-configuration mean token use per 30-minute run (millions) and, for SI, the final cold-start image-database state (segments discovered and hit rate). SC = screenshot, SI = Segment Identification.

where comparatively few new segments arose over the run, to nearly tenfold for Gemini 3.1 Pro in Matchington Mansion. However, training a YOLO model comes with a significant upfront cost both in time and token consumption. This cost profile motivates the intended division of labor: SI serves as a discovery tool for learning a new game’s visual vocabulary, after which a fine-tuned YOLO model supplies the same grounding at near-zero marginal token cost for repeated play.

Perception latency. To measure perception cost we ran each game for ten minutes with the reasoning model fixed to Gemini 3.1 Pro and logged the per-frame vision times. Because segmentation, lookup, and detection are independent of the reasoning model, these figures characterize the pipeline rather than any single agent. VLM reasoning dominates the step: per-move latency ranges from about 11 seconds for GPT-5.2, 13 for Gemini 3.1 Pro, and 17 for Qwen3.5 to 48 seconds for Kimi K2.5, which within the 30-minute cap completes about 53 actions against 135 to 159 for the others. Table 4 isolates the perception pipeline. Segmentation and lookup are the two synchronous stages of Segment Identification and run on the critical path, together a few hundred milliseconds per frame that scale with the number of segments. YOLO inference is 69–102 ms per frame. Both segmentation and YOLO inference are small against per-move reasoning of tens of seconds and are independent of the augmentation. The labeling of newly discovered segments is asynchronous and never blocks a step. Newly segmented cutouts are queued and labeled in the background at roughly two per second across ten workers. Because segments are produced faster than they are labeled, a queue builds up, so a newly seen segment is not retrievable for about 8 to 22 s, most of which is time spent waiting in the queue rather than the labeling call itself. A segment re-encountered within this lag still misses until its label lands.

Error Analysis

By examining the action-level logs, we can characterize the failure reasons. Across every experiment configuration all

issued actions were syntactically valid and within bounds, with Qwen3.5 the sole exception, showing a negligible fraction of invalid actions (less than 1%). Malformed commands and parameters are therefore not a meaningful failure mode. What separates success from failure is which action the agent selects, not whether the action executes. We identify three patterns of failures, one characteristic of each game.

Localization failure. In Tavern Rumble, the bottleneck is finding the correct interaction target. From raw screenshots the weak models issue long streams of valid but largely distinct taps. GPT-5.2 averages about 166 actions per run at 95% action diversity, and Qwen3.5 over 200, yet GPT-5.2 clears each challenge in only 2 of 5 runs and Qwen3.5 never reaches the map. The actions are almost all valid and rarely repeated, so the failure is neither an execution error nor a stuck loop. The models simply probe the screen broadly without locating the correct element to interact with. Segment Identification, which overlays the positions of detected elements, lifts both models to 5/5 on every challenge. Their actions were already valid and varied, so the improvement comes from supplying the spatial targets they could not find on their own, not from correcting how they act.

Capability ceiling. Matchington Mansion shows a different failure. The board is fully visible, so locating tiles is not the problem, yet only Gemini 3.1 Pro clears the first standard level under any configuration. Much of what the other models do changes nothing on the board. They swipe tiles that do not match or tap inert regions, and the legal matches they manage are too few to satisfy the level objective. The actions are valid and seldom repeated, so the failure is not execution or a stuck loop but an inability to reliably form matches that progress the game. Bounding boxes cannot help here, since they supply where the tiles are, not which swap produces a useful match. This is why the augmentation that rescues the weaker models in Tavern Rumble leaves Matchington unchanged.

Interference. In Idle Slayer, augmentation can push an otherwise capable model toward a worse strategy. The coin rate is driven by upgrades, which raise passive income, far

Game	Segment (ms)	Lookup (ms)	YOLO (ms)	Label rate (/s)	Label lat. (s)
Idle Slayer	143	106	69	2.16	7.9
Matchington Mansion	434	227	102	1.98	22.2
Tavern Rumble	236	130	69	1.51	13.9

Table 4: Per-frame perception latency (median ms) and asynchronous labeling, per game. Segment and Lookup are the two synchronous stages of Segment Identification and run on the critical path; YOLO is single-frame detector inference. Label rate is measured labeling throughput and Label lat. is the queued-to-retrievable time, on ten Gemini 2.5 Flash workers.

more than by manual jumping, which collects only a few coins per tap. Without augmentation Gemini 3.1 Pro buys upgrades and reaches 156 coins per second. Under SI it largely stops, completing the upgrade challenges in only 3 and 2 of 5 runs, and falls back on repeated tapping to jump, so its rate collapses to 13. GPT-5.2 moves the other way. It reaches the upgrades in every mode but earns little unaided, and under SI it becomes less repetitive and roughly eight-fold more productive. The same overlay sharpens GPT-5.2’s play on the upgrade economy while diverting Gemini 3.1 Pro from it.

Limitations

Evaluation scope. We chose games for diverse mechanics while keeping complexity within the agent’s reasoning capacity, deliberately excluding titles that require intricate real-time maneuvers such as 3D character navigation. We report five trials per configuration to account for stochasticity, but both the games and the VLMs are variable enough that firmer causal claims about specific augmentation effects would require more. Because five samples cannot support meaningful confidence intervals or significance tests, our conclusions rely on effects that hold across all five trials rather than on small differences, and we release the per-run values behind each reported rate and maximum in our public repository.

Latency and metrics. Per-move latency varies widely across models, from a few seconds to over a minute, and every run is capped at 30 minutes. This cap can itself limit what a slow model achieves: Kimi K2.5 completes only about 53 actions per run against 135 to 159 for the others, so some of its failures may reflect too few actions within the cap rather than an inability to succeed given more time. Completion results should therefore be read as performance under a fixed time budget, not an upper bound on capability. The augmentation comparisons are more robust to this, since a model faces the same cap across all of its conditions, but a longer detection list does lengthen the prompt, so we do not treat latency as identical across conditions.

Augmentation quality. The segmentation and labeling pipeline is imprecise: VLM-generated labels vary across identical segments, and a correctly labeled object is still lost if segmentation fails to detect it in a later frame. The fine-tuned YOLO detector is especially exposed, as it is trained on automatically generated labels rather than human annotations and inherits any systematic mislabeling or missed element from the upstream pipeline, which cannot be cor-

rected without revising that pipeline. The visual overlay, the semantic labels, and the coordinates are also supplied to the agent together, and their individual contributions to the reported effects have not been separated.

Visual complexity. The games we study are visually simple, with bright, cartoon-like sprites on uncluttered backgrounds, which favors segmentation and labeling. As the number of distinct on-screen elements grows, the unsupervised pipeline produces ever more classes, many of them rare, and detector quality falls. This is already visible in Matchington Mansion, where a 3,197-class vocabulary drives recall to 0.69 (Table 2). Photorealistic or cluttered scenes, heavy occlusion, fast action, and 3D layouts would strain the approach further. Developers with native access to on-screen labels could sidestep these issues; for third-party titles, human annotation remains the most reliable path to high-quality grounding.

Conclusion

Our work investigates whether the spatial grounding that VLMs lack can be supplied externally. Two methods generate this grounding without manual annotation. Segment Identification segments each frame and labels the pieces with a VLM online, while a fine-tuned YOLO detector learns the same elements offline from gameplay video. Both operate within a modular agent that adapts to new games through prompt changes alone.

Across four VLMs, three mobile games, and five trials each, augmentation let GPT-5.2 and Qwen3.5 clear challenges they could not solve from raw screenshots. The benefit is conditional: grounding helps when a model’s bottleneck is perceptual, does nothing when it is strategic, and can hurt when it diverts a capable model from a strategy that already worked. This pattern, not model size, also governs when a smaller open model reaches parity with much larger proprietary ones.

Several directions remain open. Visually richer, more cluttered games would test the perception pipeline harder, and lower-latency models and detectors are needed before such agents can act in real time. Grounding might also be applied selectively, only when a model is likely to benefit, and an oracle with perfectly labeled boxes would separate grounding quality from a model’s ability to use it.

Acknowledgments

This project was supported by the Innovation Fund Denmark (Grant ID 3194-00030B).

References

- Bolya, D.; Huang, P.-Y.; Sun, P.; Cho, J. H.; Madotto, A.; Wei, C.; Ma, T.; Zhi, J.; Rajasegaran, J.; Bangalath, H.; Wang, J.; Monteiro, M.; Xu, H.; Dong, S.; Ravi, N.; Li, S.-W.; Dollar, P.; and Feichtenhofer, C. 2025. Perception Encoder: The best visual embeddings are not at the output of the network. In Belgrave, D.; Zhang, C.; Lin, H.; Pascanu, R.; Koniusz, P.; Ghassemi, M.; and Chen, N., eds., *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 60884–60937. Curran Associates, Inc.
- Cheng, K.; Sun, Q.; Chu, Y.; Xu, F.; Li, Y.; Zhang, J.; and Wu, Z. 2024. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. In *Annual Meeting of the Association for Computational Linguistics*.
- Everingham, M.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2): 303–338.
- Evrac Studio. 2024. Tavern Rumble - Roguelike Card.
- Google. 2026. Gemini 3.1 Pro: A Smarter Model for Your Most Complex Tasks. <https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/>. Accessed: Mar. 13, 2026.
- Google DeepMind. 2025. Gemini 2.5 Flash. <https://deepmind.google/models/gemini/flash/>. Model card.
- Griffin, B. A.; Gangwar, M.; Sela, J.; and Corso, J. J. 2025. Auto-Labeling Data for Object Detection. *arXiv preprint arXiv:2506.02359*.
- Jegham, N.; Koh, C. Y.; Abdelatti, M.; and Hendawi, A. 2025. YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions. *arXiv preprint arXiv:2411.00201*.
- Kimi Team. 2026. Kimi K2.5: Visual Agentic Intelligence. *arXiv preprint arXiv:2602.02276*.
- Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollar, P.; and Girshick, R. 2023. Segment Anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3992–4003. Los Alamitos, CA, USA: IEEE Computer Society.
- Kuznetsova, A.; Rom, H.; Alldrin, N.; Uijlings, J.; Krasin, I.; Pont-Tuset, J.; Kamali, S.; Popov, S.; Mallocci, M.; Kolesnikov, A.; et al. 2020. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7): 1956–1981.
- Leban, P. 2024. Idle Slayer.
- Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, 740–755. Cham: Springer International Publishing. ISBN 978-3-319-10602-1.
- Liu, S.; Zeng, Z.; Ren, T.; Li, F.; Zhang, H.; Yang, J.; Jiang, Q.; Li, C.; Yang, J.; Su, H.; Zhu, J.; and Zhang, L. 2024. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. In *European Conference on Computer Vision (ECCV)*.
- Lu, Y.; Yang, J.; Shen, Y.; and Awadallah, A. 2024. OmniParser for Pure Vision Based GUI Agent. *arXiv preprint arXiv:2408.00203*.
- Magic Tavern, Inc. 2024. Matchington Mansion.
- Magne, L.; Awadalla, A.; Wang, G.; Xu, Y.; Belofsky, J.; Hu, F.; Kim, J.; Schmidt, L.; Gkioxari, G.; Kautz, J.; Yue, Y.; Choi, Y.; Zhu, Y.; and Fan, L. J. 2026. NitroGen: An Open Foundation Model for Generalist Gaming Agents. *arXiv preprint arXiv:2601.02427*.
- Mojang Studios. 2011. Minecraft. Video game.
- OpenAI. 2025. Introducing GPT-5.2. <https://openai.com/index/introducing-gpt-5-2/>. Accessed: Mar. 13, 2026.
- Qwen Team. 2026. Qwen3.5: Towards Native Multimodal Agents. Accessed: Mar. 16, 2026.
- Rahmanzadehgervi, P.; Bolton, L.; Taesiri, M. R.; and Nguyen, A. T. 2024. Vision Language Models are Blind: Failing to Translate Detailed Visual Features into Words. In *Proceedings of the 18th Asian Conference on Computer Vision (ACCV)*.
- Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; Mintun, E.; Pan, J.; Alwala, K. V.; Carion, N.; Wu, C.-Y.; Girshick, R.; Dollar, P.; and Feichtenhofer, C. 2025. SAM 2: Segment Anything in Images and Videos. In *The Thirteenth International Conference on Learning Representations*.
- Robinson, I.; Robicheaux, P.; Popov, M.; Ramanan, D.; and Peri, N. 2026. RF-DETR: Neural Architecture Search for Real-Time Detection Transformers. In Vondrick, C.; Hariharan, B.; Raffel, C.; Pinto, L.; Yang, D.; and Faust, A., eds., *International Conference on Learning Representations*, volume 2026, 56087–56100.
- Rocky, A.; and Wu, Q. M. J. 2025. SAM2Auto: Auto Annotation Using FLASH. *arXiv preprint arXiv:2506.07850*.
- Sapkota, R.; and Karkee, M. 2026. Object detection with multimodal large vision-language models: An in-depth review. *Information Fusion*, 126: 103575.
- SIMA Team. 2024. Scaling Instructable Agents Across Many Simulated Worlds. *arXiv preprint arXiv:2404.10179*.
- SIMA Team. 2025. SIMA 2: A Generalist Embodied Agent for Virtual Worlds. *arXiv preprint arXiv:2512.04797*.
- Tan, W.; Zhang, W.; Xu, X.; Xia, H.; Ding, Z.; Li, B.; Zhou, B.; Yue, J.; Jiang, J.; Li, Y.; An, R.; Qin, M.; Zong, C.; Zheng, L.; Wu, Y.; Chai, X.; Bi, Y.; Xie, T.; Gu, P.; Li, X.; Zhang, C.; Tian, L.; Wang, C.; Wang, X.; Karlsson, B. F.; An, B.; Yan, S.; and Lu, Z. 2025. Cradle: Empowering Foundation Agents towards General Computer Control. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 58658–58725. PMLR.
- Vo, A.; Nguyen, K.-N.; Taesiri, M. R.; Dang, V. T.; Nguyen, A. T.; and Kim, D. 2026. Vision Language Models are Biased. In *The Fourteenth International Conference on Learning Representations*.
- Wang, A.; Liu, L.; Chen, H.; Lin, Z.; Han, J.; and Ding, G. 2025. YOLOE: Real-Time Seeing Anything. In *Proceedings*

of the *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2024. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*.

Yang, J.; Zhang, H.; Li, F.; Zou, X.; Li, C.; and Gao, J. 2023. Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*.

You, K.; Zhang, H.; Schoop, E.; Weers, F.; Swearngin, A.; Nichols, J.; Yang, Y.; and Gan, Z. 2024. Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. In *European Conference on Computer Vision (ECCV)*.

Zhao, X.; Ding, W.; An, Y.; Du, Y.; Yu, T.; Li, M.; Tang, M.; and Wang, J. 2023. Fast Segment Anything. *arXiv preprint arXiv:2306.12156*.