See, Symbolize, Act: Grounding VLMs with Spatial Representations for Better Gameplay

Ashish Baghel, Paras Chopra
Lossfunk
ashish.baghel@lossfunk.com, paras@lossfunk.com

Abstract

Vision-Language Models (VLMs) excel at describing visual scenes, yet struggle to translate perception into precise, grounded actions. We investigate whether providing VLMs with both the visual frame and the symbolic representation of the scene can improve their performance on Atari games. We evaluate three state-of-the-art VLMs on games of increasing visual complexity and compare frame-only, frame with self-extracted symbols, frame with ground-truth symbols, and symbol-only pipelines. Our results indicate that all models benefit when the symbolic information is accurate. However, when VLMs extract symbols themselves, performance becomes dependent on model capability and game complexity. We further investigate how accurately VLMs can extract symbolic information from visual inputs and how noise in these symbols affects decision-making and gameplay performance. Our findings reveal that symbolic grounding is beneficial in VLMs only when symbol extraction is reliable, and highlight perception quality as a central bottleneck for future VLM-based agents.

Introduction

Vision-Language Models (VLMs) are increasingly used to build general-purpose AI agents that can both interpret visual scenes and decide how to act within them. When these systems shift from passive perception to interactive decision-making in fields like robotics, embodied AI, or game environments, they encounter tasks that rely on precise spatial understanding, an ability that current VLMs do not yet provide reliably (Waytowich et al. 2024).

Atari gameplay provides a controlled setting for studying these spatial reasoning challenges. Games like Pong, Breakout, and Space Invaders require precise tracking of paddles, balls, and aliens. Under these conditions, current VLMs frequently misidentify objects, repeat ineffective actions, or fail to execute precise controls, lagging behind fine-tuned game-specific agents.

Prior work either fine-tunes VLMs on thousands of game-specific trajectories (Zhai et al. 2024), sacrificing zero-shot generalization, or relies on visual frames alone, as in Atari-GPT (Waytowich et al. 2024), where spatial reasoning remains a challenge. However, no systematic study has evaluated how accurately VLMs identify objects and coordinates, or how the coordinate accuracy affects the models' ability to take better actions as scene complexity increases.

To probe the capabilities of current VLMs, we conduct systematic evaluations across state-of-the-art VLMs (Claude-4-Sonnet, GPT-4o, Gemini-2.5-Pro) on Atari games with varying levels of complexity (Pong, Breakout, and Space Invaders). To isolate the contributing factors, we compare four pipelines: frame-only, frame with self-extracted symbols, frame with ground-truth symbols, and ground-truth symbol-only (no visual context).

In this paper, we analyze VLMs' ability to extract symbolic information from visual input and evaluate their accuracy across multiple games. We also investigate how the quality of extracted symbols influences decision-making, and determine when symbolic information benefits or harms agent performance.

Related Work

VLMs for Game Playing Recent work has explored the use of large Vision-Language Models as zero-shot agents in interactive environments. Atari-GPT (Waytowich et al. 2024) and TextAtari (Li et al. 2025) evaluate multimodal models on Atari games and show that, while VLMs can understand visual scenes, they struggle with consistent control and spatial precision during gameplay. Fine-tuning improves decision-making in VLM-based agents (Zhai et al. 2024), but this requires task-specific data and does not resolve whether VLMs can act effectively in a purely zero-shot setting.

Object-Centric State Representations Object-centric state representations help simplify reasoning and planning in control tasks. OCAtari (Delfosse et al. 2023) extracts objects and their coordinates directly from Atari's RAM, providing ground-truth symbolic information from memory. Prior work has also shown that planning over symbolic or learned structured states benefits decision-making (Dittadi, Drachmann, and Bolander 2021). However, these approaches typically assume access to clean or near-perfect symbolic information. Our work differs by examining symbols extracted directly from vision in a zero-shot manner and studying how their quality impacts downstream actions.

Symbolic Grounding in Agents The symbolic grounding problem (Harnad 1990) highlights the fundamental chal-

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

lenge of connecting high-level symbols to perception. ReAct (Yao et al. 2022) demonstrated that combining reasoning and acting can improve tool-use-inspired language-driven action loops. More recent work focuses on improving spatial reasoning capabilities in VLMs (Chen et al. 2024; Cheng et al. 2024), and studies have analyzed why it remains a persistent challenge for such models (Chen et al. 2025). Robustness under noisy visual conditions has also been explored (Velayuthan and Tavakkoli 2025), but prior work has not examined how perception errors propagate through a symbolic control loop. PoE-World (Piriyakulkij et al. 2025) combines language models with programmatic symbolic world models for Atari tasks, but assumes reliable symbolic input.

Methodology

Our experimental setup evaluates whether symbolic grounding helps VLMs' gameplay performance. We evaluate multiple Vision-Language Models across Atari games.

Experimental Setup

Games. We use Atari games that differ in object count and the spatial reasoning required: *Pong*, *Breakout*, and *Space Invaders*. These games were chosen because their spatial control demands are known to challenge VLMs. These games span different visual complexity levels: Pong (2–4 objects: ball, paddle), Breakout (5–15 objects: ball, paddle, bricks), and Space Invaders (20–50 objects: player, aliens, bullets, shields). These games allow us to test how symbolic grounding behaves as the gameplay becomes more complex.

Models. We evaluate state-of-the-art VLMs: Claude-4-Sonnet, GPT-4o, and Gemini-2.5-Pro. All models were used in a zero-shot setting without any fine-tuning.

Pipelines

Frame + Ground-Truth Symbols. This pipeline reads symbolic information directly from the game RAM using OCAtari (Delfosse et al. 2023), which provides perfect object positions with zero detection error and serves as the upper bound for all other pipelines. The VLM uses both the frame and the symbols to select the next action.

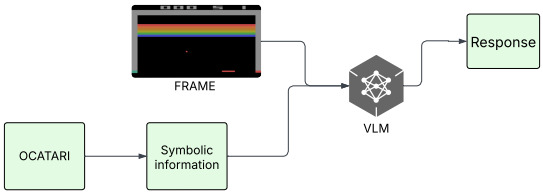

Figure 1: Frame + Ground-Truth Symbols

Frame-only. The VLM receives only the raw game frame and available action mappings, and outputs the action for the next frame. This setup tests whether the model can perform spatial reasoning using only visual input, without any symbolic help.

Frame + Self-Extracted Symbols. This pipeline follows a two-stage process. First, the VLM extracts object-centric symbolic information from the frame as structured data, including object IDs, labels, (x, y) coordinates, and confidence scores. The VLM then uses both the frame and the extracted symbols to select the next action. The detailed steps of the symbol extraction process and its limitations are discussed later in the Symbolic Detection Quality subsection, where we analyze *why* symbolic grounding helps or hurts performance.

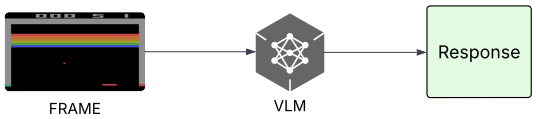

Figure 2: Frame-only pipeline

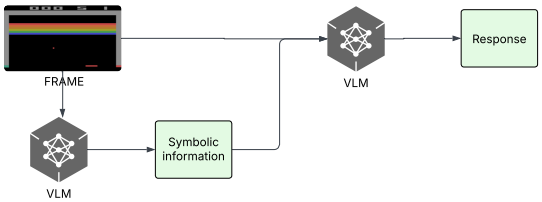

Figure 3: Frame + Self-Extracted Symbols

Symbol-Only. The VLM receives object coordinates from the RAM state, with *no visual frame*. This pipeline isolates the contribution of symbolic information and tests whether VLMs can play Atari by reasoning purely with symbols that provide perfect object detection and no visual context at all. This reveals whether symbols alone are sufficient or whether visual grounding is essential for VLM decision-making.

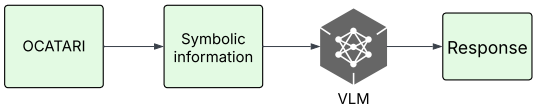

Figure 4: Symbol-only pipeline

Evaluation Metrics

Gameplay Metric. We use cumulative reward as the gameplay metric. The game is paused at every frame, and the VLM decides the next action. Each model plays 600 frames per run, and uses a frame size of 1280×720, across 2 seeds. This setup isolates reasoning quality, since real-time latency is not the focus of this study. To compare models on a common scale, we map raw scores to the 0–100 interval. For each model, we define its Frame + GT Symbols performance as the upper bound (100%). The lower bound is the worst: -17 for Pong, 0 for Breakout, and 0 for Space Invaders over 600 frames.

Detection Metrics. To better understand *why* different pipelines succeed or fail, we measure symbolic extraction quality by comparing VLM-generated symbols with ground-truth annotations across 100 frames per game. We use standard detection metrics:

- **F1 score**: A balanced check of how many objects it identifies correctly (without missing too many or adding extras).
- **IoU (overlap score)**: How much the model's guessed coordinates for objects match up with the real ones.

Prompt Structure We use a generic prompt template for every pipeline, supplying only the actions and the raw inputs (frame, symbols, or both). This ensures that the model receives no task-specific instructions, examples, or strategy hints. The exact wording for the Frame + Self-Extracted Symbols condition is given below; the other variants differ only by the presence or absence of the image or symbol block. Complete prompts for all conditions are listed in Appendix .

Table 1: Example prompt structure for Frame + Self-Extracted Symbols pipeline (abbreviated).

Frame + Self-Extracted Symbols Prompt

```
You are an expert [game] player
analyzing a game frame.

Game controls:
- Action 0: NOOP, Action 1: FIRE, ...

Current frame analysis:
- Total objects detected: 7

Detected objects with coordinates:
- Object 'ball': x=580, y=450, size
15x20
- Object 'paddle': x=850, y=650, size
135x15
- Object 'brick_wall': x=640, y=260,
size 1140x120
- ...

IMPORTANT: Use symbolic information
when reliable,
but prioritize visual reasoning if data
is incomplete.

Choose the optimal action. Return JSON:
{"reasoning": "...", "action": integer}
```

Results

Main Finding: Model-Specific Effects of Symbolic Grounding

We find that symbolic grounding helps only when a model can extract accurate symbols, and this ability breaks down differently across models and game complexity. Table 2 shows this pattern clearly.

The improvement is most pronounced for Claude-4-Sonnet. In the frame-only pipeline, Claude performs poorly in all games. It scores -16.0 in *Pong* and 0.0 in *Breakout*, repeatedly taking the same incorrect actions because it misidentifies the paddle and its position in both games. When we provide self-extracted symbols, Claude's performance improves substantially: It reaches -3.0 in *Pong* and 12.0 in *Breakout*, essentially matching the upper bound. Even in the visually dense *Space Invaders* environment, symbols increase Claude's score from 80.0 to 150.0, as the model is now able to take more informed and effective actions.

GPT-4o and Gemini-2.5-Pro behave differently. In the frame-only pipeline, both achieve stronger baseline scores than Claude, indicating better visual-based reasoning. Adding symbols results in only modest gains in the simpler games: Gemini improves to -3.0 in *Pong* and 10.0 in *Breakout*, while GPT-4o reaches -6.5 in Pong and 8.0 in Breakout. In *Space Invaders*, however, both models show substantial performance degradation. The 20–50 objects on screen overwhelm their detection capabilities, producing many incorrect coordinates that mislead their reasoning and negate potential gains.

Table 2: Cumulative reward over 600 frames across pipelines. Abbreviations: F = Frame-only; S-GT = Symbol-only; F+S-self = Frame + Self-Extracted Symbols; F+S-GT = Frame + Ground-Truth Symbols.

Model	Pipeline	Pong	Breakout	Space Invaders
Claude-4-Sonnet	F+S-GT	-1.0	12.0	175.0
	F	-16.0	0.0	80.0
	S-GT	-14.0	0.0	90.0
	F+S-self	-3.0	12.0	150.0
Gemini-2.5-Pro	F+S-GT	-1.0	12.0	170.0
	F	-7.0	7.0	95.0
	S-GT	-12.0	3.0	95.0
	F+S-self	-3.0	10.0	80.0
GPT-4o	F+S-GT	-3.0	13.0	185.0
	F	-5.0	7.5	130.0
	S-GT	-14.0	0.0	105.0
	F+S-self	-6.5	8.0	65.0

The Symbol-only pipeline shows that visual grounding is essential. Even when models receive perfect symbolic coordinates identical to ground truth, removing the visual frame causes performance to collapse. In Breakout, Claude-4-Sonnet and GPT-4o both score zero points. In Space Invaders, in the best case, GPT-4o drops from 185 points (with Frame-only) to just 105 (57% of the upper bound) without visual context. This proves that symbolic information alone is insufficient. Visual frames act as essential scaffolding, giving VLMs the perceptual context they need to correctly interpret and trust coordinate data.

The Frame + Ground-Truth Symbols results show that when models receive perfect symbols derived from the RAM state and visual frame, their performance consistently improves, confirming that accurate symbolic information is al-

ways beneficial. To evaluate how close each model gets to its upper bound with symbolic grounding, Figure 5 shows detailed model-wise performance relative to the Frame + Ground-Truth Symbols.

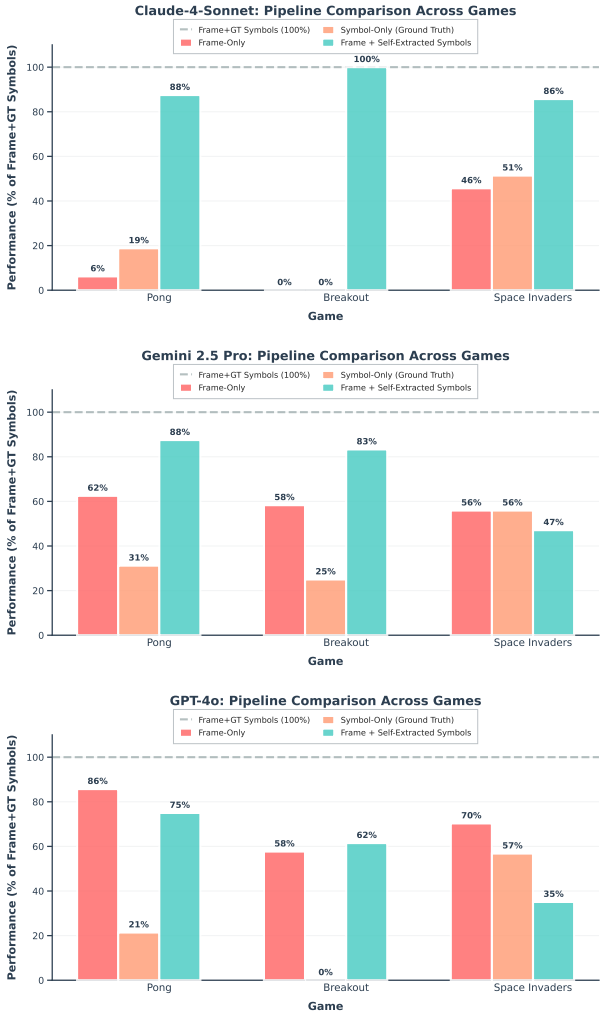

Figure 5: Model-wise performance breakdown across all three games, normalized to each model's Frame+GT Symbols performance. Top: Claude-4-Sonnet shows consistent improvement with symbolic grounding across all games. Middle: Gemini-2.5-Pro benefits in simpler games but struggles in Space Invaders. Bottom: GPT-4o shows mixed results, with symbolic information sometimes harming performance.

Symbolic Detection Quality Across VLMs

We find that the effect of object-centric information on gameplay depends directly on object detection accuracy. Table 3 shows that Claude-4-Sonnet achieves substantially higher extraction accuracy than Gemini-2.5-Pro and GPT-4o, which explains why symbolic grounding helps some models but hurts others.

Claude-4-Sonnet achieves an F1 score of 0.715 and an IoU of 0.533, indicating relatively accurate object detection and localization. This accuracy enables the Frame + Self-Extracted Symbols pipeline to provide useful information that improves model performance. In contrast, Gemini-2.5-Pro and GPT-4o achieve much lower F1 scores (0.189 and 0.124), missing and mislocating the majority of objects. These errors reduce the reliability of the symbolic representations and degrade decision-making performance.

Table 3: Object detection metrics across models, averaged over 100 frames per game.

Model	F1 Score	IoU
Claude-4-Sonnet	0.715	0.533
Gemini-2.5-Pro	0.189	0.202
GPT-4o	0.124	0.128

Ablation Studies

We conduct two ablation studies to examine when symbolic information improves gameplay performance.

How Input Resolution Affects Object Detection Accuracy

Having established that detection quality determines whether symbolic information improves or degrades gameplay performance, we next examine how image resolution influences VLMs' ability to extract accurate object coordinates.

Methodology We evaluate Claude-4-Sonnet's object detection and localization quality across four input resolutions on 100 frames per game:

- **Original** (160×210): Native Atari resolution with no scaling.
- **Uniform** $4\times$ (640×840): Uniformly scaled while preserving the original aspect ratio.
- **Square** (720×720): Square frame that distorts the original 4:3 aspect ratio.
- **Current** (1280×720): The resolution used in our experiments, scaled by $8\times$ in width and $4\times$ in height.

Object extraction quality is measured using F1 score and IoU, and then compared against OCAtari ground-truth annotations.

Table 4: Detection quality across input resolutions for Claude-4-Sonnet (averaged 100 frames per game).

Game	160×210	640×840	720×720	1280×720
F1 Score (higher indicates better performance)				
Pong	0.28	0.52	0.55	**0.58**
Breakout	0.31	0.68	0.70	**0.71**
Space Inv.	0.35	0.71	0.73	**0.75**
Average	0.31	0.64	0.66	**0.68**

Results The native Atari resolution has low detection accuracy (average F1: 0.31), which is below the threshold at which Object-centric information becomes useful, and instead harms gameplay. Increasing resolution improves extraction quality, with 1280×720 achieving an average F1 of 0.68, approximately twice the original value. We observe that results beyond 720px on the shorter side: 720×720 (0.66) and 1280×720 (0.68) perform similarly. This suggests that total pixel count and detail preservation may be more important than preserving the original aspect ratio.

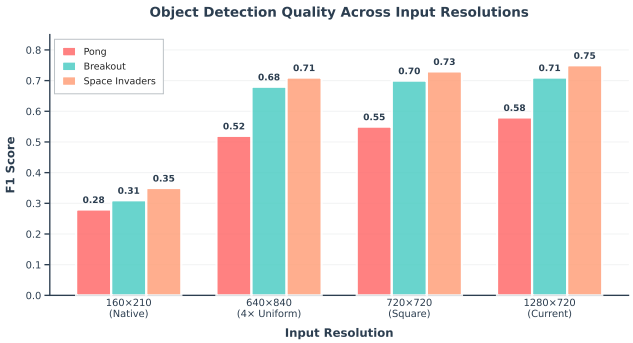

Figure 6: Detection quality (F1 score) across resolutions and games. The native resolution (160×210) produces lower detection accuracy (F1 < 0.35), while higher resolutions enable effective symbolic grounding (F1 > 0.65).

Noise Ablation: Robustness to Detection Errors

We study how different levels of noise in object coordinates affect the usefulness of Object-centric information. This isolates the sensitivity of action selection to detection errors.

Experimental Design We focus on Claude-4-Sonnet and GPT-4o, the two models that differ substantially in object detection quality.

The experiment injects Gaussian noise into ground-truth coordinates to simulate different levels of detection error.

$$x' = x + \mathcal{N}(0, \sigma \times W), \quad y' = y + \mathcal{N}(0, \sigma \times H) \quad (1)$$

Here, W and H denote the frame width and height, respectively. where σ controls noise magnitude. The five noise levels tested are:

- $\sigma = 0.00$: no-noise baseline
- $\sigma = 0.10$: Low noise ($\sim$16–20 pixels of coordinate error)
- $\sigma = 0.20$: Moderate noise ($\sim$32–40px)
- $\sigma = 0.30$: High noise ($\sim$48–60px)
- $\sigma = 0.40$: Severe noise ($\sim$64–80px)

Noise is sampled independently per frame and per object, replicating the randomness present in real-world visual frames.

For each model, game, and noise level, we run 10 independent seeds, each covering 300 frames. This reduces the variance and provides reliable mean performance estimates with confidence intervals.

Table 5: Performance under coordinate noise (mean reward across 10 seeds; standard deviation shown in Fig. 7).

Game	**Model**	σ=0.0	σ=0.1	σ=0.2	σ=0.3	σ=0.4
Pong	Claude-4-Sonnet	−0.6	−2.2	−2.2	−2.2	−2.2
	GPT-4o	−2.7	−3.0	−3.4	−3.8	−2.8
Breakout	Claude-4-Sonnet	5.0	4.3	3.4	3.4	2.8
	GPT-4o	5.0	4.0	3.0	2.3	2.6
Space Inv.	Claude-4-Sonnet	53	57	57	88	52
	GPT-4o	86	43	69	66	62

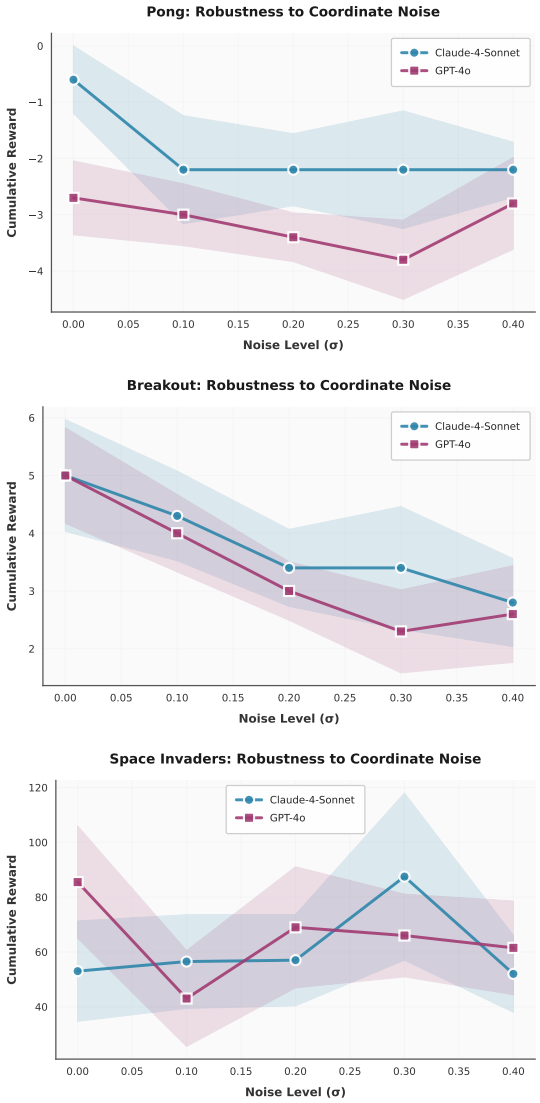

Figure 7: Noise robustness analysis across games for Claude-4-Sonnet and GPT-4o. Pong (top); Breakout (middle); Space Invaders (bottom). Bars mark 95% confidence intervals over 10 seeds.

Results In games like Pong and Breakout, both models show similar degradation: even a low level of noise causes a significant drop in performance. At $\sigma = 0.1$ (just 16–20 pixels of error), scores decline by roughly 30–40% relative to the no-noise baseline. Claude drops from -0.6 to -2.2 in Pong while GPT-4o decreases from -2.7 to -3.0, and a similar decline happens in Breakout. By $\sigma = 0.2$, the symbolic pipeline offers no clear advantage over using the frame alone.

Space Invaders behaves differently. The dense formation of 20–50 aliens means even random firing can hit targets and score points, which partly explains the high variance in results. Claude's scores vary from 53 to 88 points across noise levels, whereas GPT-4o decreases sharply from 86 (no noise) to 43 at $\sigma = 0.1$, losing half its performance from minimal noise.

Limitations and Future Work

While our findings demonstrate when symbolic grounding helps or harms VLMs, several limitations remain that open avenues for future work.

Limited Environment Diversity. Our evaluation focuses on three Atari games. Although they provide controlled settings for spatial reasoning, they represent simple 2D environments. Future work should test symbolic grounding in more diverse and realistic domains, including 3D, physics-based, and complex games.

Symbol Quality Bottleneck. Our results show that the effectiveness of symbolic grounding is highly sensitive to symbol extraction quality. Future work should explore more robust symbol-extraction methods, such as hybrid detectors or lightweight fine-tuning of vision modules, to improve object-centric representations.

Cost and Practicality. Querying VLMs per frame incurs substantial latency and cost, making real-time gameplay infeasible with current models. Scaling symbolic grounding to real-time agents will therefore require fundamental efficiency improvements.

Conclusion

In this work, we set out to understand when symbolic grounding provides measurable benefits to VLMs in gameplay tasks. Our results show that symbolic information is useful only when the extracted symbols are sufficiently accurate. When object positions are reliable, they provide the model with spatial information that Frame-only struggles with. However, when symbols are noisy, they introduce ambiguity into the model's reasoning and lead to degraded decision-making. Even perfect symbolic information is insufficient without visual context. Our ablation further demonstrates that simply increasing the frame resolution significantly boosts object detection accuracy. This shows that reliable perception is the bottleneck, not the idea of symbolic grounding itself. Future work must focus on robust symbol extraction.

References

Chen, B.; Xu, Z.; Kirmani, S.; Ichter, B.; Driess, D.; Florence, P.; Sadigh, D.; Guibas, L.; and Xia, F. 2024. SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, S.; Zhu, T.; Zhou, R.; Zhang, J.; Gao, S.; Niebles, J. C.; Geva, M.; He, J.; Wu, J.; and Li, M. 2025. Why Is Spatial Reasoning Hard for Vision-Language Models? In *International Conference on Machine Learning (ICML)*.

Cheng, A.-C.; Yin, H.; Fu, Y.; Guo, Q.; Yang, R.; Kautz, J.; Wang, X.; and Liu, S. 2024. SpatialRGPT: Grounded Spatial Reasoning in Vision-Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Delfosse, Q.; Blüml, J.; Gregori, B.; Sztwiertnia, S.; and Kersting, K. 2023. OCAtari: Object-Centric Atari 2600 Reinforcement Learning Environments. *arXiv preprint arXiv:2306.08649*.

Dittadi, A.; Drachmann, F. K.; and Bolander, T. 2021. Planning from Pixels in Atari with Learned Symbolic Representations. In *AAAI Conference on Artificial Intelligence*.

Harnad, S. 1990. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 42(1-3): 335–346.

Li, W.; Li, W.; Shen, C.; Sheng, J.; Huang, Z.; Wu, D.; Hua, Y.; Yin, W.; Wang, X.; Zha, H.; and Jin, B. 2025. TextAtari: 100K Frames Game Playing with Language Agents. *arXiv preprint arXiv:2506.04098*.

Piriyakulkij, W. T.; Liang, Y.; Tang, H.; Weller, A.; Kryven, M.; and Ellis, K. 2025. PoE-World: Compositional World Modeling with Products of Programmatic Experts. *arXiv preprint arXiv:2505.10819*.

Velayuthan, P.; and Tavakkoli, A. 2025. Evaluating Robustness of Vision-Language Models Under Noisy Conditions. *arXiv preprint arXiv:2509.12492*.

Waytowich, N. R.; White, D.; Sunbeam, M.; and Goecks, V. G. 2024. Atari-GPT: Benchmarking Multimodal Large Language Models as Low-Level Policies in Atari Games. *arXiv preprint arXiv:2408.15950*.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*.

Zhai, Y.; Bai, H.; Lin, Z.; Pan, J.; Tong, S.; Zhou, Y.; Suhr, A.; Xie, S.; LeCun, Y.; Ma, Y.; and Levine, S. 2024. Fine-Tuning Large Vision-Language Models as Decision-Making Agents via Reinforcement Learning. *arXiv preprint arXiv:2405.10292*.

Appendix

This appendix provides the complete prompts used in our experiments, including the symbolic information extraction prompt and the prompts for each pipeline.

Symbolic Extraction by VLM Prompt

This prompt is used by the Frame + Self-Extracted Symbols pipeline to extract object-centric symbolic information from the visual frame.

```
You are an expert game frame analyzer for
    the game Breakout.

Your task is to detect ALL visible
    objects in the image
with high precision. Detect all distinct,
    visible objects
(like players, enemies, projectiles,
    items, scores). For
each object, provide its label, a tight
    bounding box
[x1, y1, x2, y2], and a confidence score.

Return ONLY valid JSON in the following
    format:
{
    "objects": [
        {
            "id": "unique_id",
            "label":
    "object_type_or_description",
            "coordinates": [x1, y1, x2,
    y2],
            "confidence": 0.95,
            "description": "brief
    description of the object"
        }
    ],
    "image_info": {
        "total_objects": 0,
        "frame_analysis": "brief
    description of what you
                           see in the
    frame"
    }
}
```

Frame-only Pipeline Prompt

```
You are an expert Breakout player
    controlling the
ORANGE PADDLE at the bottom.

IMPORTANT: You are controlling the ORANGE
    PADDLE at
the bottom.

Game controls:
- Action 0: NOOP (do nothing)
- Action 1: FIRE (primary action - often
    shoot/serve/activate)
- Action 2: RIGHT (move right or right
    action)
- Action 3: LEFT (move left or left
    action)
- Action 4: RIGHTFIRE (combination of
    right + fire)
- Action 5: LEFTFIRE (combination of left
    + fire)

As an expert Breakout player controlling
    the ORANGE PADDLE
at the bottom, analyze the visual scene
    and choose the
optimal action.

Think step by step:
1. Observe the current state of the game
2. Predict the trajectory or movement
    patterns
3. Consider your strategic options
4. Choose the optimal action

Return ONLY JSON:
{
    "reasoning": "your expert analysis
    and decision rationale",
    "action": integer_action_code
}
```

Frame + Self-Extracted Symbols Pipeline Prompt

```
You are an expert game player analyzing a
    game frame.

Game controls:
- Action 0: NOOP (do nothing)
- Action 1: FIRE (primary action - often
    shoot/serve/activate)
- Action 2: RIGHT (move right or right
    action)
- Action 3: LEFT (move left or left
    action)
- Action 4: RIGHTFIRE (combination of
    right + fire)
- Action 5: LEFTFIRE (combination of left
    + fire)

Current frame analysis:
- Total objects detected: 7

Detected objects with coordinates and
    positions:
- Object 'score_display': positioned at
    coordinates
  x=337, y=32, size 95x35
- Object 'score_display': positioned at
    coordinates
  x=462, y=32, size 95x35
- Object 'level_display': positioned at
    coordinates
  x=1105, y=32, size 40x35
- Object 'brick_wall': positioned at
    coordinates
  x=640, y=260, size 1140x120
- Object 'player_paddle': positioned at
    coordinates
  x=857, y=657, size 135x15

IMPORTANT: Use the symbolic information
    when available
and reliable, but prioritize visual
    reasoning if objects
```

```
are missing or the symbolic data seems
    incomplete.

As an expert player, analyze the scene
    and choose the
optimal action.

Think step by step:
1. Observe the current state of the game
2. Predict the trajectory or movement
    patterns
3. Consider your strategic options
4. Choose the optimal action

Return ONLY JSON:
{
    "reasoning": "your expert analysis
    with positional awareness",
    "action": integer_action_code
}
```

Frame + Ground-Truth Symbols Pipeline Prompt

```
You are an expert Breakout player.

Game controls:
- Action 0: NOOP
- Action 1: FIRE
- Action 2: RIGHT
- Action 3: LEFT

Current game state (OCAtari ground truth
    coordinates):
Total objects: 110

Detected objects:
- Player: x=792, y=648, size=128x13
- Block: x=64, y=195, size=1152x20
- Block: x=64, y=216, size=1152x20
- Block: x=64, y=236, size=1152x20
- NoObject: x=0, y=0, size=0x0
  ... [many NoObject entries abbreviated]

Analyze the game state and choose the
    optimal action.

Think step by step:
1. Observe the current state of the game
2. Predict the trajectory or movement
    patterns
3. Consider your strategic options
4. Choose the optimal action

Return ONLY JSON:
{
    "reasoning": "your analysis",
    "action": integer_action_code
}
```

Symbol-Only Pipeline Prompt

```
You are playing BREAKOUT using ONLY
    coordinate information
(no visual display).

Rules:
- Hit bricks with the ball to destroy
    them and earn points
- Don’t let the ball fall below the paddle
- Y-axis: 0 (top) to 210 (bottom)
- X-axis: 0 (left) to 160 (right)
- Paddle is at bottom (y ˜= 190)

STRATEGY:
- Calculate horizontal distance between
    ball and paddle
- Predict where ball will land when it
    falls (y ˜= 190)
- If ball is above paddle and falling (dy
    > 0): position
  paddle to intercept
- If ball_x > paddle_x: move RIGHT to
    align
- If ball_x < paddle_x: move LEFT to align
- Account for ball velocity to predict
    landing position

AVAILABLE ACTIONS:
0: NOOP, 1: FIRE, 2: RIGHT, 3: LEFT

CURRENT STATE:
=== GAME STATE (Symbolic Coordinates) ===

YOUR CONTROL (Player/Paddle):
  - Position: x=47, y=189
  - Size: 16x4
BALL:
  - Position: x=53, y=147
OTHER OBJECTS (6 total):
  - BlockRow: x=8, y=57
  - BlockRow: x=8, y=63
  - BlockRow: x=8, y=69
  ... [many Object entries abbreviated]

INSTRUCTIONS:
You must make a decision based ONLY on
    the coordinate
data above. No visual frame is provided.

Perform spatial analysis:
1. Parse all object positions, sizes, and
    velocities
2. Calculate critical distances
    (ball-to-paddle, etc.)
3. Predict future positions using
    velocities
4. Identify immediate threats (enemies
    close, etc.)
5. Select the action that best addresses
    the most
   critical situation

Think step-by-step, then output:
REASONING: [Your spatial analysis and
    calculations]
ACTION: [number only]
```