Video-Grounded Verification for Long-Horizon Visual Imitation Learning

Quan Chen, Chenrui Shi, Qi Chen, Yuwei Wu, *Member, IEEE*, Xintong Zhang, Rui Gao, Haoyu Zhang, Zhi Gao, Che Sun, Kun Wu, and Yunde Jia

***Abstract*—Visual imitation learning aims to translate a human demonstration video into an action plan or robot-executable code that enables a robot to faithfully reproduce the demonstrated behavior. Recent approaches based on vision-language models (VLMs) rely on these models' ability to translate visual observations into action plans or codes. However, in long-horizon visual imitation learning, errors in temporal ordering, spatial grounding, or plan–code consistency can accumulate over time, *i.e.*, compounding errors, resulting in significant deviations from the intended behavior. To address this problem, we propose LongVIL, a framework that introduces video-grounded verification into both plan and code generation. LongVIL first generates a structured action plan and verifies each step against the demonstration video through segment-level temporal consistency and end-frame spatial grounding. It then translates the verified plan into robot-executable code and verifies whether each code step faithfully implements the verified action plan. We further introduce LongVILBench, a benchmark for long-horizon demonstration-to-code imitation with 150 tabletop manipulation tasks and 300 human demonstration videos recorded under two visual conditions, featuring sequences of up to 18 steps, six spatial relation types, and three difficulty levels. Experiments on LongVILBench and SeeDo show that LongVIL consistently improves long-horizon visual imitation by reducing temporal, spatial, and plan–code errors.**

***Index Terms*—Imitation learning, learning from demonstration, data sets for robot learning.**

Received 2 April 2026; accepted 16 July 2026. Date of publication 12 August 2026; date of current version 28 August 2026. This article was recommended for publication by Associate Editor David Fridovich-Keil and Editor Wei Pan upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62406009, in part by Shenzhen Science and Technology Program under Grant JCYJ20241202130548062, in part by the Natural Science Foundation of Shenzhen under Grant JCYJ20230807142703006, and in part by the Key Research Platforms and Projects of the Guangdong Provincial Department of Education under Grant 2023ZDZX1034. *(Corresponding authors: Zhi Gao; Che Sun.)*

Quan Chen, Qi Chen, and Rui Gao are with the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Guangdong Laboratory of Machine Perception and Intelligent Computing, Shenzhen MSU-BIT University, Shenzhen 518172, China.

Chenrui Shi, Yuwei Wu, Xintong Zhang, Haoyu Zhang, and Zhi Gao are with the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: gaozhibit@bit.edu.cn).

Che Sun and Yunde Jia are with the Guangdong Laboratory of Machine Perception and Intelligent Computing, Shenzhen MSU-BIT University, Shenzhen 518172, China (e-mail: sunche@smbu.edu.cn).

Kun Wu is with the Beijing Innovation Center of Humanoid Robotics, Beijing 101111, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2026.3723308, provided by the authors.

Digital Object Identifier 10.1109/LRA.2026.3723308

I. Introduction

VISUAL imitation learning (VIL) aims to translate human demonstration videos into structured action plans or robot-executable code, so that a robot can faithfully reproduce the demonstrated task, providing a promising paradigm for teaching robots manipulation skills from human videos [1], [2], [3], [4], [5], [6], [7], [8], [9]. Recent advances in vision-language models (VLMs) have significantly expanded this paradigm. Existing VLM-based VIL systems typically take sparse visual observations from a demonstration, such as sampled frames or selected key moments, to interpret the procedure and generate a structured plan, action description, or robot-executable program [10], [11], [12]. This formulation improves modularity and interpretability, and naturally supports reusable low-level skills or code-based execution. It has shown promising performance on short demonstrations with a few atomic actions, where the procedure is relatively simple and the visual evidence is easier to recover.

However, many real-world manipulation tasks are inherently long-horizon. Successfully imitating them requires accurate action ordering over many steps, precise object-centric spatial grounding, and faithful alignment between the recovered plan and the final executable code [13], [14], [15], [16]. A central challenge is compounding error: early mistakes (such as errors in temporal ordering, spatial grounding, or plan–code consistency) may accumulate over plan and code generation, causing deviations from the intended behavior. This problem may arise in many current VLM-based VIL systems because they rely on direct generation from demonstrations without explicitly checking whether each predicted step is supported by the video evidence before execution [17].

To address this problem, we propose LongVIL, a framework for long-horizon visual imitation with video-grounded verification in both plan and code generation. LongVIL first generates a structured action plan and verifies each step against the demonstration video by checking temporal consistency and spatial grounding. It then translates the verified plan into robot-executable code and performs a second verification stage to ensure that the generated program faithfully implements the verified action sequence, reducing plan–code mismatch before execution. By explicitly verifying both the recovered plan and its executable implementation, LongVIL mitigates error accumulation in long-horizon generation.

To support systematic evaluation, we introduce LongVILBench, a benchmark for long-horizon imitation learning. LongVILBench contains 150 tabletop manipulation tasks and 300 human demonstration videos recorded under two visual conditions, with action sequences of up to 18 steps, six spatial

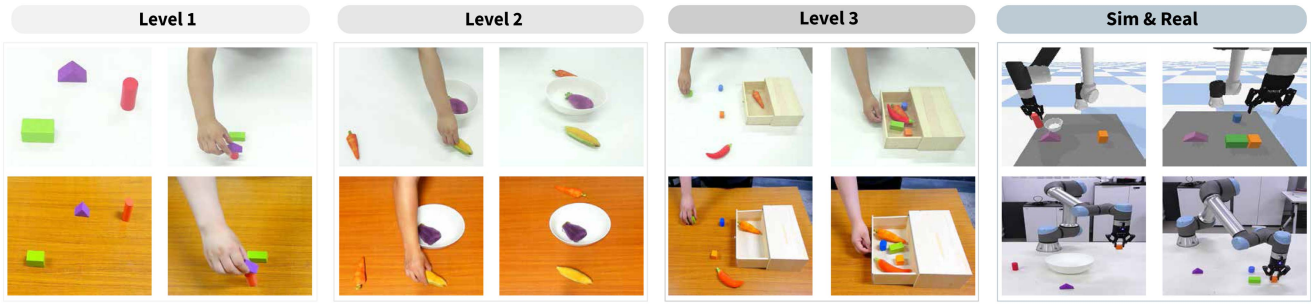

Fig. 1. Example images from LongVILBench. LongVILBench contains 150 tabletop manipulation tasks and 300 human demonstration videos recorded under two visual conditions, with tasks grouped into three difficulty levels according to action-sequence length.

relation types, and three difficulty levels. Rather than merely increasing task length, it is designed to stress long temporal dependencies and object-centric spatial grounding. Fig. 1 illustrates representative tasks from LongVILBench across different task categories and difficulty levels. We evaluate LongVIL on both LongVILBench and the SeeDo benchmark [18], together with ablation studies, cost analysis, error analysis, and real-robot experiments. Results show that the video-grounded verification consistently improves long-horizon visual imitation by reducing temporal, spatial, and plan–code errors. Our contributions are summarized as follows:

- We propose LongVIL, a framework for long-horizon visual imitation from human demonstration videos that introduces video-grounded verification in both plan and code generation.
- We introduce LongVILBench, a benchmark for systematic evaluation of long-horizon imitation learning, designed to analyze long temporal dependencies and object-centric spatial grounding.
- We conduct extensive experiments on LongVILBench and SeeDo across multiple baselines and model backbones, showing that video-grounded verification consistently improves long-horizon visual imitation.

II. Related Work

A. VLMs for Visual Imitation Learning

Recent work has incorporated vision-language models (VLMs) into visual imitation learning. One line uses VLMs as high-level reasoning modules to translate demonstration videos into symbolic steps, structured plans, or robot-executable code [10], [11], [12], [18], [19], [20], [21], [22], [23], [24]. This demo-to-plan or demo-to-code paradigm improves modularity and interpretability, and naturally supports reusable low-level skills or code-based execution. However, many existing methods in this line rely on direct generation, without checking whether the generated actions are actually supported by the demonstration evidence. This becomes increasingly problematic in long-horizon visual imitation learning, where temporal inconsistency, spatial mis-grounding, and plan–code mismatch can accumulate across steps. Another line uses VLMs and videos more directly for low-level control, including VLM-based action grounding and vision-language-action (VLA) methods that leverage human demonstration videos [14], [25], [26], [27]. These approaches target visuomotor policy learning rather than directly translating demonstration videos into symbolic plans or executable code, and typically rely on large-scale annotated training data.

Our work is most closely related to the first line, i.e., long-horizon demonstration-to-code generation from human videos. Unlike prior direct demonstration-to-code pipelines, LongVIL introduces video-grounded verification to verify and correct generated plans and code using the demonstration itself.

TABLE I
Comparison of Existing Benchmarks by Data Type (Data), Task Horizon (Length), Number of Spatial Relation Types (Spatial), Number of Tasks (Tasks), and Explicit Difficulty Levels (Diff.)

Benchmark	Data	Length	Spatial	Tasks	Diff.
Imitrob [28]	videos	1	0	56	✗
FetchBench [29]	images	3–5	5	6156	✗
SeeDo [18]	videos	2–8	4	106	✗
Ours	videos	1–18	6	150	✓

B. Benchmarks for Visual Imitation Learning

Existing VIL benchmarks still focus on short-horizon tasks with limited structural complexity. As summarized in Table I, Imitrob [28] and FetchBench [29] mainly consist of short sequences, offering limited temporal dependency and spatial reasoning. SeeDo [18] introduces multi-step sequences and spatial relations, but still provides limited coverage of long action chains, diverse spatial interactions, and does not explicitly stratify tasks by difficulty level.

In contrast, LongVILBench supports long action sequences together with richer temporal dependencies, more diverse spatial relations, and explicit difficulty levels. This enables fine-grained evaluation of long temporal dependencies and object-centric spatial grounding in long-horizon imitation.

C. Verification and Correction Mechanisms

Verification, correction, and self-refinement are widely studied for improving generated plans, programs, and agent behaviors. Existing methods often perform verification with respect to an external reference: LLM agents use textual instructions, task goals, or feedback for replanning [30], [31], [32], [33], [34], while robot program-synthesis methods use formal specifications or simulation-based validation to check the program [35], [36]. These works improve reliability, but their verification reference is usually provided outside the demonstration.

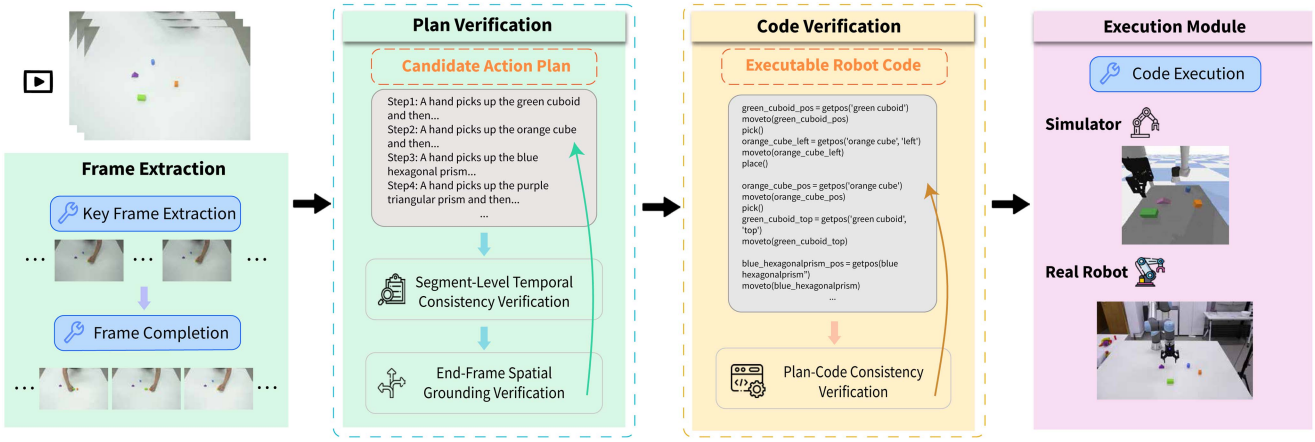

Fig. 2. Overview of LongVIL. The framework verifies a candidate action plan from a demonstration video through segment-level temporal consistency and end-frame spatial grounding, and then verifies plan–code consistency before execution in simulation or the real world.

Video-based verification has also been studied in video understanding, where temporal action segmentation, procedure recognition, and neuro-symbolic reasoning verify whether observed events match action labels, procedural steps, or temporal specifications [37], [38], [39], [40]. These methods focus on recognizing, segmenting, or verifying events in videos rather than producing executable robot code. LongVIL instead uses the human demonstration video itself as the task-defining reference to verify both the recovered action plan and the generated code for long-horizon demonstration-to-code imitation.

III. Method

A. Overview

Given a demonstration video V, our goal is to generate a robot-executable program Π^* that reproduces the demonstrated behavior in this video. LongVIL factorizes this process into two verification stages. It first derives a candidate action plan A from V and refines it into a verified plan A^* through video-grounded verification. The structured natural-language plan makes action order, objects, and spatial relations explicit and associates each action with video evidence, enabling temporal and spatial verification before code generation. It then translates A^* into a candidate program Π and further verifies plan–code consistency to obtain the final executable program Π^*. Fig. 2 illustrates the overall LongVIL pipeline. By verifying the recovered plan against the demonstration video and then checking whether the generated code preserves the verified plan, LongVIL reduces error accumulation in long-horizon imitation.

B. Plan Verification

Plan verification checks whether each predicted action is consistent with the video evidence, including both the observed motion and the final object relation after the action. We first construct a candidate plan from the demonstration video. For each frame f_t in V, we estimate the 3D hand position h_t using MediaPipe [41], [42] and compute hand velocity:

$$v_t = \|h_{t+1} - h_t\|_2. \tag{1}$$

Local minima in the smoothed velocity curve are selected as initial keyframes, as they often correspond to interaction transitions such as grasping or releasing. Let the resulting keyframe sequence be $K = \{f_{i_1}, f_{i_2}, \ldots, f_{i_m}\}$. To improve temporal coverage in long demonstrations, we further densify the sequence when two neighboring keyframes are too far apart. Specifically, for an adjacent pair $(f_i, f_j) \in K$ with $j - i > \Delta$, we insert two candidate positions:

$$k_1 = \left\lfloor i + \frac{1}{3}(j-i) \right\rfloor, \qquad k_2 = \left\lfloor i + \frac{2}{3}(j-i) \right\rfloor, \tag{2}$$

and map them to the nearest valid hand-visible frames. Given the resulting keyframes and the detected object set O, a VLM generates a candidate action plan $A = \{\alpha_1, \alpha_2, \ldots, \alpha_T\}$, where each action is represented as $\alpha_i = \langle a_i, S_i \rangle$. For each predicted action α_i, the VLM returns a natural-language action description a_i together with the original frame IDs S_i that define the local video segment supporting the action. In practice, each segment contains 2.66 frames on average, with 83% of segments containing 2–3 frames.

The resulting plan provides action-level predictions aligned with local visual evidence, but some actions may still be unsupported or incorrectly grounded. We therefore verify each action in two complementary steps: segment-level temporal consistency and end-frame spatial grounding.

Segment-Level Temporal Consistency Verification: We first verify whether each predicted action is temporally consistent with the motion observed in its assigned segment. For each candidate action $\alpha_i = \langle a_i, S_i \rangle$, the VLM-based temporal verifier takes the ordered keyframes S_i, the queried action description a_i, and the detected object set O as input and is prompted to infer the demonstrated motion from the ordered keyframes and compare it with the queried action:

$$(l_i^{\text{temp}}, e_i^{\text{temp}}) = \mathcal{V}_{\text{temp}}(S_i, a_i, O), \tag{3}$$

where $l_i^{\text{temp}} \in \{\text{Yes}, \text{No}, \text{Unclear}\}$ denotes whether the motion in S_i supports the action semantics in a_i, and e_i^{temp} is a textual explanation of the judgment. The Yes label indicates that the motion is consistent with the queried action, so the action is kept unchanged. The No label indicates an explicit mismatch, and the action is revised using the verifier explanation. The Unclear label indicates insufficient or ambiguous visual evidence, and

the action is kept as a provisional result and resolved together with the subsequent spatial grounding result. The action after temporal verification is defined as follows:

$$\tilde{a}_i = \begin{cases} a_i, & l_i^{\text{temp}} = \text{Yes}, \\ \mathcal{C}_{\text{temp}}(a_i, e_i^{\text{temp}}, O), & \text{otherwise}, \end{cases} \tag{4}$$

where $\mathcal{C}_{\text{temp}}$ denotes a VLM-based temporal correction step, rather than an additional trained model. It revises No cases according to the verifier explanation and passes Unclear cases as provisional results to the subsequent spatial grounding step.

End-Frame Spatial Grounding Verification: Temporal consistency alone does not guarantee that an action is spatially grounded correctly. An action may have the right coarse type but still imply an incorrect relation, such as *into*, *on top of*, or *to the left of*. We therefore examine the end frame of each aligned segment, denoted by f_i^{end}, which captures the object configuration after the action. We apply a VLM-based spatial grounding verifier to infer the final relation supported by the end-frame object configuration:

$$r_i^{\text{spa}} = \mathcal{V}_{\text{spa}}(f_i^{\text{end}}, \tilde{a}_i, O), \tag{5}$$

where r_i^{spa} denotes the grounded spatial relation between the manipulated object and the reference object. The VLM-based spatial correction step, denoted by $\mathcal{C}_{\text{spa}}$, then compares the relation described in $\tilde{a}_i$ with r_i^{spa}. If they are inconsistent, it revises the spatial relation or the reference object in the action; otherwise, it keeps the temporally verified action unchanged:

$$a_i^* = \begin{cases} \tilde{a}_i, & \text{if consistent with } r_i^{\text{spa}}, \\ \mathcal{C}_{\text{spa}}(\tilde{a}_i, r_i^{\text{spa}}, e_i^{\text{temp}}), & \text{otherwise}. \end{cases} \tag{6}$$

For actions with the `Unclear` label, $\tilde{a}_i$ remains provisional. After r_i^{spa} is obtained, $\mathcal{C}_{\text{spa}}$ uses both e_i^{temp} and r_i^{spa} to decide whether to keep $\tilde{a}_i$ or revise it. This step grounds the final spatial state of actions involving explicit spatial relations, while temporal verification remains responsible for determining whether the manipulation itself is supported by the video segment. After temporal and spatial verification, we obtain a verified action plan $A^* = \{\alpha_1^*, \alpha_2^*, \ldots, \alpha_T^*\}$, where $\alpha_i^* = \langle a_i^*, S_i \rangle$, and each action is locally supported by the demonstration video.

C. Code Verification

We translate the verified plan A^* into executable robot code. Given A^*, a VLM generates a candidate program $\Pi = [\pi_1, \pi_2, \ldots, \pi_T]$, where each code step π_i corresponds to an action α_i^*. The candidate program is composed of a predefined set of robot primitives, summarized in Table II. The primitive set is extensible: new primitives can be added by specifying their textual descriptions, required inputs, and executable control modules, making them available throughout the LongVIL pipeline.

Plan–Code Consistency Verification: Even if the verified plan is correct, code generation may still introduce wrong objects, spatial parameters, primitive calls, or operation order. We therefore check whether each generated code step faithfully implements the verified action before execution. For each action–code pair (α_i^*, π_i), we apply a VLM-based plan–code verifier:

$$(l_i^{\text{pc}}, e_i^{\text{pc}}) = \mathcal{V}_{\text{pc}}(\alpha_i^*, \pi_i), \tag{7}$$

where $l_i^{\text{pc}} \in \{\text{Yes}, \text{No}\}$ indicates whether π_i faithfully implements the semantics of α_i^*, and e_i^{pc} provides an explanation of the inconsistency. Typical mismatches include incorrect primitive calls, wrong object arguments, and invalid spatial parameters.

TABLE II
ROBOT PRIMITIVES USED IN CANDIDATE CODE GENERATION

Primitive	Description
`getpos(obj)`	Returns the 3D position of object `obj`.
`getpos(obj, dir)`	Returns the position in direction `dir` (e.g., left, right) relative to object `obj`.
`moveto(pos)`	Moves the robot end-effector to position `pos`.
`pick()`	Grasps the object at the current end-effector position.
`place()`	Releases the currently held object.
`open()`	Opens the drawer.
`close()`	Closes the drawer.

When a mismatch is detected, we revise the code using a VLM-based correction step, denoted by $\mathcal{C}_{\text{pc}}$:

$$\pi_i^* = \begin{cases} \pi_i, & l_i^{\text{pc}} = \text{Yes}, \\ \mathcal{C}_{\text{pc}}(\alpha_i^*, \pi_i, e_i^{\text{pc}}), & \text{otherwise}, \end{cases} \tag{8}$$

where $\mathcal{C}_{\text{pc}}$ denotes the code correction step implemented by the VLM. This correction keeps the verified action plan fixed and only revises the generated code. This verification stage ensures that the executable program remains faithful to the verified action sequence. The final executable program is $\Pi^* = [\pi_1^*, \pi_2^*, \ldots, \pi_T^*]$.

Finally, the verified program Π^* is executed on a simulator or a real robot platform. Each primitive is mapped to the corresponding low-level controller of the target platform to reproduce the demonstrated behavior.

IV. LONGVILBENCH

We introduce LongVILBench, a benchmark designed to assess long temporal dependencies and object-centric spatial grounding in long-horizon demonstration-to-code imitation from human demonstration videos.

A. Formulation

Each task instance in LongVILBench is represented as

$$\mathcal{D} = (\mathcal{O}, \mathcal{P}, \mathcal{V}, \bar{\mathcal{A}}, \mathbf{\Pi}), \tag{9}$$

where

- $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ is the set of physical objects involved in the task;
- $\mathcal{P} = \{\text{pos}(o_1), \ldots, \text{pos}(o_n)\}$ denotes their initial 3D positions;
- $\mathcal{V} = \{f_1, f_2, \ldots, f_{T'}\}$ is a human demonstration video with T' RGB frames;
- $\bar{\mathcal{A}} = [a_1, a_2, \ldots, a_T]$ is the temporally ordered action sequence, and each action is represented as $a_t = [u_t, o_t, d_t]$, with u_t the atomic operation, o_t the manipulated object, and d_t the target spatial relation;
- $\mathbf{\Pi} = [\pi_1, \pi_2, \ldots, \pi_T]$ is the program, where each π_t is the executable code corresponding to action a_t.

For actions such as *open* and *close* that do not require a target spatial relation, unused fields are set to null.

B. Task Design

LongVILBench contains three representative tabletop manipulation categories: *block manipulation*, *tabletop cleanup*, and

vegetable sorting. Tasks are composed of four atomic operations: *pick*, *place*, *open*, and *close*. The benchmark involves 14 unique objects and six spatial relation types: *left*, *right*, *front*, *behind*, *on top of*, and *into*. To support controlled analysis across increasing horizon length, tasks are divided into three difficulty levels according to action-sequence length: Level 1 (1–4 actions), Level 2 (5–8 actions), and Level 3 (9–18 actions). This stratification enables evaluation under progressively longer horizons and denser spatial interactions.

C. Evaluation Metrics

LongVILBench uses three metrics to evaluate long-horizon program generation: *Exact Match Accuracy (EMA)*, *Final State Accuracy (FSA)*, and *Step-wise Matching Score (SMS)*.

Exact Match Accuracy (EMA): EMA measures whether the predicted program exactly matches the reference program:

$$\text{EMA} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\hat{\mathbf{\Pi}}_i = \mathbf{\Pi}_i], \tag{10}$$

where N is the number of examples, $\hat{\mathbf{\Pi}}_i$ is the predicted program, and $\mathbf{\Pi}_i$ is the reference program.

Final State Accuracy (FSA): FSA measures whether the predicted and reference programs produce the same final environment state:

$$\text{FSA} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\hat{\mathcal{S}}_i = \mathcal{S}_i], \tag{11}$$

where $\hat{\mathcal{S}}_i$ and $\mathcal{S}_i$ denote the final states produced by the predicted and reference programs, respectively. In practice, we execute both programs using a lightweight state-transition script to derive their final states, avoiding irrelevant variability from full simulation and enabling reliable comparison.

Step-wise Matching Score (SMS): SMS measures the correct program prefix before the first deviation:

$$\text{SMS} = \frac{1}{N}\sum_{i=1}^{N}\frac{l_i}{T_i}, \tag{12}$$

where T_i is the number of steps in the reference program $\mathbf{\Pi}_i$, and l_i is the matching prefix length between $\hat{\mathbf{\Pi}}_i$ and $\mathbf{\Pi}_i$. We use prefix-based evaluation because long-horizon manipulation is strongly sequential: an early error can invalidate many later steps even when they are locally plausible.

D. Dataset Collection

Dataset construction follows three stages: (1) task-plan generation, (2) simulation-based validation, and (3) real-world human demonstration recording. For each task category and difficulty level, we use prompt templates to guide GPT-4o in generating diverse task plans, which are manually reviewed for semantic and logical correctness and tested for physical feasibility in PyBullet [43] with a UR5e robot. Each validated plan is then enacted by a human demonstrator in a real-world tabletop setup and recorded twice under two visual conditions: once in a controlled setup with fixed lighting, camera position, and a clean background, and once with viewpoint and lighting variations to simulate more realistic visual conditions. In total, LongVILBench contains 150 tabletop manipulation tasks, 300 RGB demonstration videos, and 2,332 annotated actions (7.8 actions per video on average). Each difficulty level contains 50 tasks and 100 videos.

V. Experiments

A. Experimental Setup

We evaluate LongVIL on two benchmarks: LongVILBench and the SeeDo benchmark [18]. LongVILBench provides a controlled benchmark for long-horizon demonstration-to-code imitation with three difficulty levels, while the SeeDo benchmark enables evaluation on an existing benchmark with different task categories and data distribution. We report EMA, FSA, and SMS for each LongVILBench level and in total, together with overall results on the SeeDo benchmark.

B. Baselines and Backbones

We compare LongVIL with four representative baselines. SeeDo and GPTforRobots [22] are direct video-to-code approaches that generate robot programs from demonstration videos. Self-Refine [31] is a generic self-correction baseline, while KitchenVLA [24] is a representative closed-loop refinement method based on robot execution feedback. We instantiate LongVIL with four vision-language backbones: GPT-4o [44], Qwen-VL-Max [45] (QwenMax), Qwen3-VL-8B [46] (Qwen3), and InternVL-3.5-8B [47] (InternVL). **Base** denotes direct generation without verification, and **Verify** denotes the verification-enhanced variant.

C. Implementation Details

To control for backbone effects, all compared baseline methods are instantiated with GPT-4o. We use the official code for direct video-to-code baselines, including SeeDo and GPTforRobots. For refinement-style baselines, including Self-Refine and KitchenVLA, we compare their correction mechanisms under the same initial plan in a unified symbolic setting. Table V summarizes the key implementation settings.

D. Quantitative Results

Table III reports the main results on LongVILBench and the SeeDo benchmark. On LongVILBench, performance decreases substantially from Level 1 to Level 3 for all methods, confirming the long-horizon difficulty of the benchmark. For example, Verify-GPT4o drops from 0.920 EMA on Level 1 to 0.250 EMA on Level 3. Despite this difficulty increase, the Verify variants generally improve over their corresponding Base variants across all four backbones, indicating that the gains mainly come from the verification mechanism rather than a specific backbone. Overall, Verify-GPT4o achieves the best LongVILBench Total results, with 0.537 EMA, 0.547 FSA, and 0.701 SMS, outperforming the strongest non-LongVIL baseline, Self-Refine, which obtains 0.433, 0.473, and 0.653. On SeeDo, Verify-GPT4o also achieves the best overall performance with 0.562 EMA, 0.580 FSA, and 0.731 SMS, showing that LongVIL is effective beyond LongVILBench.

E. Qualitative Results

Fig. 3 presents a representative example comparing Base and Verify. As shown in the figure, LongVIL first verifies the

TABLE III
MAIN RESULTS ON LONGVILBENCH ACROSS THREE DIFFICULTY LEVELS AND IN TOTAL, TOGETHER WITH OVERALL GENERALIZATION RESULTS ON THE SEEDO BENCHMARK. BEST RESULTS ARE HIGHLIGHTED IN BOLD

Method	LongVILBench Level 1			LongVILBench Level 2			LongVILBench Level 3			LongVILBench Total			SeeDo Benchmark		
	EMA	FSA	SMS	EMA	FSA	SMS	EMA	FSA	SMS	EMA	FSA	SMS	EMA	FSA	SMS
SeeDo [18]	0.350	0.350	0.542	0.060	0.060	0.220	0.000	0.000	0.113	0.137	0.137	0.292	0.363	0.363	0.665
GPTforRobots [22]	0.340	0.340	0.805	0.140	0.140	0.480	0.150	0.150	0.432	0.210	0.210	0.572	0.342	0.342	0.508
Self-Refine [31]	0.800	0.920	0.940	0.320	0.320	0.576	0.180	0.180	0.444	0.433	0.473	0.653	0.412	0.418	0.587
KitchenVLA [24]	0.840	0.920	0.940	0.360	0.360	0.567	0.090	0.090	0.248	0.430	0.457	0.585	0.366	0.366	0.501
Base-Qwen3	0.640	0.640	0.740	0.190	0.190	0.409	0.110	0.110	0.337	0.313	0.313	0.495	0.372	0.400	0.583
Verify-Qwen3	0.700	0.700	0.820	0.220	0.220	0.463	0.160	0.180	0.412	0.360	0.367	0.565	0.474	0.485	0.655
Base-InternVL	0.520	0.520	0.708	0.030	0.030	0.249	0.000	0.000	0.098	0.183	0.183	0.352	0.310	0.312	0.542
Verify-InternVL	0.630	0.680	0.789	0.210	0.210	0.384	0.040	0.040	0.204	0.293	0.310	0.459	0.411	0.434	0.597
Base-QwenMax	0.680	0.680	0.819	0.280	0.280	0.566	0.120	0.120	0.465	0.360	0.360	0.617	0.407	0.407	0.597
Verify-QwenMax	0.730	0.730	0.866	0.300	0.300	0.538	0.170	0.170	0.490	0.400	0.400	0.631	0.506	0.506	0.663
Base-GPT4o	0.760	0.760	0.870	0.340	0.340	0.564	0.190	0.190	0.422	0.430	0.430	0.618	0.466	0.466	0.663
Verify-GPT4o	**0.920**	**0.940**	**0.960**	**0.440**	**0.440**	**0.648**	**0.250**	**0.260**	**0.495**	**0.537**	**0.547**	**0.701**	**0.562**	**0.580**	**0.731**

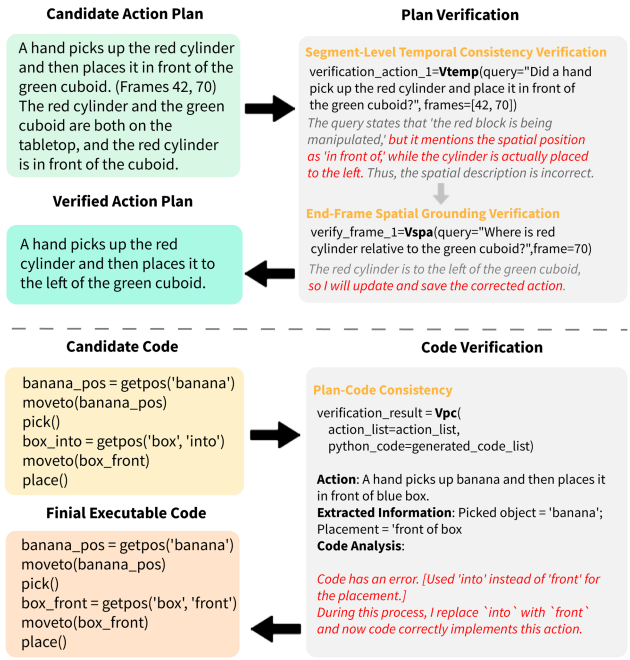

Fig. 3. Representative qualitative comparison between Base and Verify.

generated plan against the aligned video segment by checking segment-level temporal consistency and end-frame spatial grounding, and then verifies whether the synthesized code remains consistent with the corrected plan. In this example, the Base pipeline produces unsupported plan and code predictions, whereas Verify revises them before execution and yields a program that is more faithful to the demonstration. This example provides an intuitive illustration of how video-grounded verification improves long-horizon visual imitation.

F. Ablation Study

We conduct ablation studies on keyframe completion (KC), plan verification (PV), and code verification (CV), with results reported in Table IV. Along the incremental path from A to B, C, and E, performance improves from 0.430/0.430/0.618 to 0.537/0.547/0.701 in EMA/FSA/SMS, showing that KC, PV, and CV provide complementary gains. KC improves candidate-plan construction by providing denser temporal evidence, PV corrects temporally unsupported or spatially mis-grounded actions, and CV further reduces plan–code mismatches. Setting D evaluates verification without keyframe completion and still outperforms Base, indicating that both verification stages remain beneficial even without KC. Its lower performance than the full model E shows that keyframe completion and verification are complementary, and that combining all three components yields the best performance.

TABLE IV
ABLATION RESULTS ON LONGVILBENCH. KC, PV, AND CV DENOTE KEYFRAME COMPLETION, PLAN VERIFICATION, AND CODE VERIFICATION

Setting	Configuration	EMA	FSA	SMS
A	Base	0.430	0.430	0.618
B	+ KC	0.463	0.463	0.657
C	+ KC + PV	0.500	0.500	0.679
D	+ PV + CV, w/o KC	0.473	0.473	0.662
E	+ KC + PV + CV, Full	**0.537**	**0.547**	**0.701**

G. Real-World Experiments

We further evaluate whether the verified programs generated by LongVIL can be executed on a physical robot. Experiments are conducted on a UR5e arm equipped with a Robotiq 2F-85 gripper and a fixed overhead RGB camera, using the same predefined motion-primitives interface as in LongVIL. We select 5 tasks from each difficulty level of LongVILBench, repeat each task 5 times under the same initial setup, and count a trial as successful only if the robot completes the demonstrated plan. Table VI summarizes the results.

LongVIL achieves an overall real-world success rate of 86.7%. The success rate decreases from 96.0% on Level 1 to 76.0% on Level 3, consistent with the increasing task horizon and difficulty. Fig. 4 shows representative executions on object organization and stacking tasks. Most failures are caused by low-level execution limitations, such as grasp misalignment or imperfect placement, rather than incorrect high-level symbolic plans.

TABLE V
KEY IMPLEMENTATION SETTINGS OF LONGVIL

Item	Setting
Keyframe detection settings	MediaPipe HandLandmarker; max hands = 2, detection / presence / tracking thresholds = 0.2 / 0.2 / 0.2
Smoothing parameters	Gaussian filter with $\sigma = 5$
Threshold Δ	$\Delta = 50$ for clean videos, $\Delta = 100$ for complex videos
Correction rules	Each iteration performs plan verification followed by code verification
Maximum correction iterations	1
VLM decoding settings	API default settings, temperature = 1.0 and top-$p = 1.0$

TABLE VI
REAL-WORLD DEPLOYMENT RESULTS ON A UR5E ROBOT. TASK-WISE SR IS REPORTED AS MEAN ± STANDARD DEVIATION OVER TASKS

Level	Num. Tasks	Successful Trials / Total	Task-wise SR (%)
Level 1	5	24 / 25	96.0 ± 8.0
Level 2	5	22 / 25	88.0 ± 9.8
Level 3	5	19 / 25	76.0 ± 8.0
Total	15	65 / 75	86.7 ± 11.9

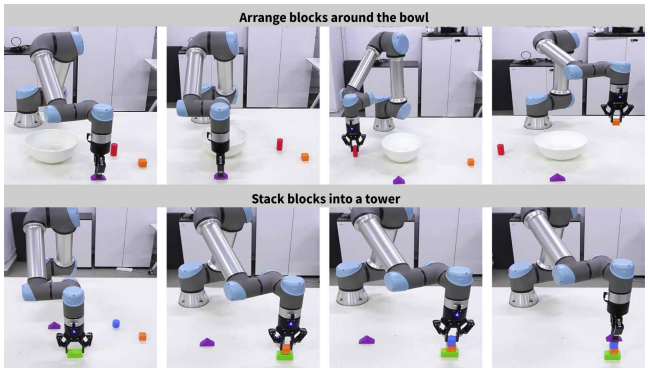

Fig. 4. Representative real-world executions of LongVIL on a UR5e robot for object organization and stacking tasks.

TABLE VII
PERFORMANCE AND COST OF REPRESENTATIVE METHODS ON LONGVILBENCH. TOKENS ARE REPORTED IN MILLIONS AND RUNTIME IN THOUSANDS OF SECONDS

Method	EMA	FSA	SMS	Tokens (M)	Time (k s)
SeeDo [18]	0.137	0.137	0.292	16.8	85.4
GPTforRobots [22]	0.210	0.210	0.572	17.2	17.5
Self-Refine [31]	0.433	0.473	0.653	31.8	28.5
KitchenVLA [24]	0.430	0.457	0.585	19.2	75.4
Base-GPT4o	0.430	0.430	0.618	**15.6**	**14.1**
Verify-GPT4o	**0.537**	**0.547**	**0.701**	32.3	33.7

H. Cost Analysis

Table VII summarizes the performance–cost trade-off on LongVILBench. Compared with Base-GPT4o, Verify-GPT4o uses more tokens and runtime (32.3 M vs. 15.6 M tokens and 33.7 k vs. 14.1 k seconds), but improves EMA/FSA/SMS from 0.430/0.430/0.618 to 0.537/0.547/0.701. Compared with refinement baselines, Verify-GPT4o uses roughly as many tokens as Self-Refine while achieving higher accuracy, and requires much less runtime than KitchenVLA while also obtaining better performance. Direct video-to-code baselines use fewer tokens in general, but their lower cost does not lead to competitive long-horizon imitation performance.

TABLE VIII
ERROR RATES OF THE THREE TARGETED FAILURE TYPES ON LONGVILBENCH FOR **BASE-GPT4O** AND **VERIFY-GPT4O**. EACH FAILED CASE IS ASSIGNED TO THE FIRST ERROR CAUSING DEVIATION FROM THE DEMONSTRATION. LOWER IS BETTER

Error Type	Base-GPT4o (%)	Verify-GPT4o (%)
Temporal inconsistency	19.0	**15.7**
Spatial mis-grounding	28.3	**22.3**
Plan–code mismatch	9.7	**8.3**

I. Error Analysis

Table VIII reports the three targeted failure types in LongVIL—temporal inconsistency, spatial mis-grounding, and plan–code mismatch—for Base-GPT4o and Verify-GPT4o over the full set of 300 demonstration instances in LongVILBench. Verify-GPT4o reduces all three targeted error types compared with Base-GPT4o, indicating that the gains are aligned with the intended verification mechanisms. Spatial mis-grounding remains the dominant failure source, although its error rate decreases from 28.3% to 22.3%, suggesting that fine-grained spatial reasoning from visual evidence remains challenging for current VLMs even with explicit verification.

Some errors remain because the VLM-based verifiers are not perfect oracles. Although verification improves performance overall, an incorrect verifier judgment may leave an existing error uncorrected or occasionally degrade an initially correct prediction. The remaining temporal errors are often related to imperfect action–segment correspondence. Since this correspondence is generated during candidate-plan construction rather than manually annotated, the assigned keyframes may be incomplete, temporally shifted, or mixed with neighboring actions. Spatial grounding remains the largest error source because fine-grained relations such as left/right, front/behind, into, and on top of can be visually subtle, occluded, or view-dependent. Plan–code mismatches are less frequent, but they can still occur when an incorrect object argument or spatial parameter is syntactically valid and difficult for the verifier to distinguish from a faithful implementation.

VI. DISCUSSIONS AND CONCLUSIONS

We presented LongVIL, a framework for long-horizon visual imitation learning from human demonstration videos. By introducing video-grounded verification into both plan and code generation, LongVIL reduces temporal inconsistency, spatial mis-grounding, and plan–code mismatch in long-horizon demonstration-to-code imitation. We also introduced LongVILBench, a benchmark for systematic evaluation of long-horizon visual imitation with explicit difficulty levels and diverse visual conditions. Experiments on LongVILBench and the SeeDo benchmark, together with real-world robot deployment results, show that LongVIL consistently improves long-horizon visual

imitation and produces executable programs that are more faithful to the demonstrated behavior.

These results suggest that video-grounded verification is a practical way to improve long-horizon demonstration-to-code imitation without additional training. At the same time, performance still depends on the quality of candidate-plan construction, video–segment alignment, and predefined execution primitives. Future work will explore stronger perception, more accurate video grounding, more adaptive execution primitives, and tighter integration between verification and execution.

REFERENCES

[1] C. Finn et al., "One-shot visual imitation learning via meta-learning," in *Proc. Conf. Robot Learn.*, 2017, pp. 357–368.

[2] S. Dasari and A. Gupta, "Transformers for one-shot visual imitation," in *Proc. Conf. Robot Learn.*, 2021, pp. 2071–2084.

[3] S. Li et al., "Robust visual imitation learning with inverse dynamics representations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 12, 2024, pp. 13609–13618.

[4] P. Sharma, D. Pathak, and A. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 2597–2607.

[5] G. Chen et al., "FMimic: Foundation models are fine-grained action learners from human videos," *Int. J. Robot. Res.*, 2025, Art. no. 02783649251377335, doi: 10.1177/02783649251377335.

[6] A. Jonnavittula, S. Parekh, and D. P. Losey, "View: Visual imitation learning with waypoints," *Auton. Robots*, vol. 49, no. 1, pp. 1–26, 2025.

[7] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4126–4133, Apr. 2022.

[8] A. Bonardi, S. James, and A. J. Davison, "Learning one-shot imitation from humans without humans," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3533–3539, Apr. 2020.

[9] C. Zhu et al., "Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets," in *Proc. Robot. Sci. Syst.*, 2025.

[10] Y. Mu et al., "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 25081–25094, 2023.

[11] D. Patel, H. Eghbalzadeh, N. Kamra, M. L. Iuzzolino, U. Jain, and R. Desai, "Pretrained language models as visual planners for human assistance," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 15302–15314.

[12] Y. Wang, G. Gonzalez-Pumariega, Y. Sharma, and S. Choudhury, "Demo2Code: From summarizing demonstrations to synthesizing code via extended chain-of-thought," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023.

[13] A. Gupta et al., "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. Conf. Robot Learn.*, 2019, pp. 1025–1037.

[14] R. Yang et al., "EGOVLA: Learning vision-language-action models from egocentric human videos," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2026.

[15] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 844–851, Apr. 2020.

[16] M. Grunde-McLaughlin, R. Krishna, and M. Agrawala, "AGQA: A benchmark for compositional spatio-temporal reasoning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11287–11297.

[17] S. Jain, S. Sachdeva, and R. Paul, "Enabling long(er) horizon imitation for manipulation tasks by modeling subgoal transitions," in *Proc. Conf. Robot Learn.*, 2025, pp. 2219–2238.

[18] B. Wang, J. Zhang, S. Dong, I. Fang, and C. Feng, "VLM see, robot do: Human demo video to robot action plan via vision language model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2025, pp. 17215–17222.

[19] J. Liang et al., "Code as policies: Language model programs for embodied control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9493–9500.

[20] D. Driess et al., "PaLM-e: An embodied multimodal language model," in *Proc. Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 8469–8488.

[21] S. Xie, H. Wang, Z. Xiao, R. Wang, and X. Chen, "Robotic programmer: Video instructed policy code generation for robotic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2025, pp. 14923–14930.

[22] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi, "GPT-4V(ision) for robotics: Multimodal task planning from human demonstration," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 10567–10574, Nov. 2024.

[23] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24824–24837 .

[24] K. Lu, C. Ma, C. Hori, and D. Romeres, "KitchenVLA: Iterative vision-language corrections for robotic execution of human tasks," in *Proc. Int. Conf. Robot. Automat., Workshop Safe Vis.-Lang. Models Robot*, 2025.

[25] H. Huang et al., "RoboGround: Robotic manipulation with grounded vision-language priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 22540–22550.

[26] S. Kareer et al., "Emergence of human to robot transfer in vision-language-action models," in *Proc. Robot. Sci. Syst.*, 2026.

[27] H. Luo et al., "Being-H0: Vision-language-action pretraining from large-scale human videos," in *Proc. Int. Conf. Mach. Learn.*, 2026.

[28] J. Sedlar et al., "Imitrob: Imitation learning dataset for training and evaluating 6D object pose estimators," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2788–2795, May 2023.

[29] B. Han et al., "FetchBench: A simulation benchmark for robot fetching," in *Proc. Conf. Robot Learn.*, 2025, vol. 270, pp. 3053–3071.

[30] N. Shinn et al., "Reflexion: Language agents with verbal reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 8634–8652.

[31] A. Madaan et al., "Self-refine: Iterative refinement with self-feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 46534–46594 .

[32] S. Yuan et al., "Agent-R: Training language model agents to reflect via iterative self-training," 2025, *arXiv:2501.11425*.

[33] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan, "ReplanVLM: Replanning robotic tasks with visual language models," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 10201–10208, Nov. 2024.

[34] S. Pchelintsev et al., "LERa: Replanning with visual feedback in instruction following," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2025, pp. 19218–19225.

[35] Y. Yang, N. P. Bhatt, W. Ward, Z. Hu, J. Biswas, and U. Topcu, "Joint verification and refinement of language models for safety-constrained planning," 2024, *arXiv:2410.14865*.

[36] Z. Hu, J. J. Li, A. Guha, and J. Biswas, "Robo-instruct: Simulator-augmented instruction alignment for finetuning code LLMs," in *Proc. 2nd Conf. Lang. Model.*, 2025.

[37] H. Ahn and D. Lee, "Refining action segmentation with hierarchical video representations," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16302–16310.

[38] T. J. Schoonbeek, T. Houben, H. Onvlee, P. H. N. de With, and F. Van der Sommen, "IndustReal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2024, pp. 4365–4374.

[39] K. Aouaidjia, W. Zhang, A. Li, and C. Zhang, "Improving action segmentation via explicit similarity measurement," *Eng. Appl. Artif. Intell.*, vol. 164, Art. no. 113251, 2026.

[40] M. Choi, H. Goel, M. Omama, Y. Yang, S. Shah, and S. Chinchali, "Towards neuro-symbolic video understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 220–236.

[41] F. Zhang et al., "MediaPipe hands: On-device real-time hand tracking," in *Proc. CVPR Workshop Comput. Vis. Augmented Virtual Reality*, 2020.

[42] C. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," 2019, *arXiv:1906.08172*.

[43] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016. [Online]. Available: https://pybullet.org

[44] OpenAI, "GPT-4o system card," OpenAI Tech. Rep., 2024. [Online]. Available: https://openai.com/index/gpt-4o-system-card/

[45] Alibaba Cloud, "Alibaba cloud model studio: Qwen-VL-Max series," Official Documentation, 2026. [Online]. Available: https://www.alibabacloud.com/help/en/model-studio/

[46] S. Bai et al., "Qwen3-VL technical report," 2025, *arXiv:2511.21631*.

[47] W. Wang et al., "InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency," 2025, *arXiv:2508.18265*.