WHERE TO SPLIT? BENCHMARKING SUBTASK SPLITTING IN OPEN-WORLD ROBOT DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Subtask splitting is crucial for structuring long robot demonstrations into semantically meaningful units for robot learning. However, existing benchmarks and methods largely focus on constrained settings, limiting their applicability to rapidly growing and increasingly diverse open-world robot data. We introduce RoboSplitBench, a benchmark for systematically evaluating subtask splitting in open-world robot data. It contains demonstrations from diverse open-world scenes, with proprioception, multi-view observations, and human-annotated temporal subtask transition points. To establish a strong reference solution on this benchmark, we further propose RoboSplit. It is a training-free approach that explicitly assigns complementary roles to proprioception and Vision-Language Model (VLM) reasoning. Proprioception guides temporal windowing and selective use of different views, while VLMs predict subtask transition points. We systematically evaluate different methods on RoboSplitBench across diverse scenes and transition types, revealing the challenges of open-world subtask splitting. The strongest baseline achieves an F1 score of 0.479, while RoboSplit establishes a strong reference at 0.713 and consistently outperforms representative baselines across diverse scenes. As robot datasets continue to grow rapidly, our work provides a foundation for their structuring. *We provide an anonymous link to RoboSplitBench's data, and an example data sample in supplementary materials.*

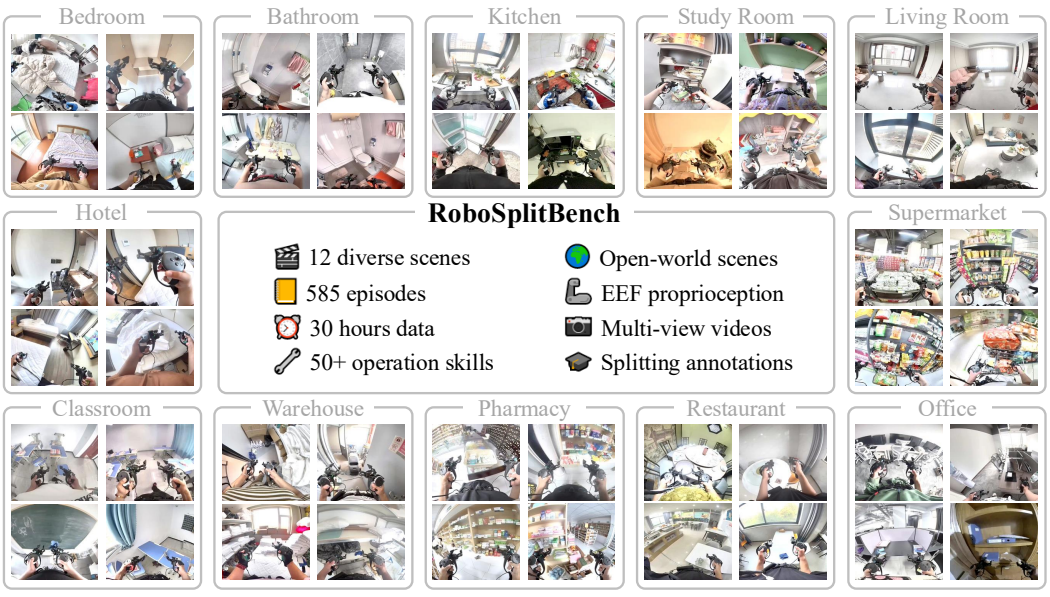

Figure 1: We introduce RoboSplitBench to evaluate the ability of models to split open-world robot trajectories into subtasks. It contains 30 hours of data across 12 open-world scenes, with EEF proprioception, multi-view observations, and human splitting annotations. Using RoboSplitBench, we evaluate representative splitting baselines and further propose our algorithm, RoboSplit.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 2: Subtask splitting of robot data. Given a long-horizon robot trajectory, the goal is to split it into subtasks by predicting the temporal transition points between adjacent subtasks.

1 Introduction

Robot data is growing at an unprecedented scale, with increasingly diverse demonstrations collected in open-world environments and emerging sources such as egocentric recordings becoming available to robot learning (O’Neill et al., 2024; Khazatsky et al., 2024; Walke et al., 2023; Jones et al., 2025; Punamiya et al., 2026; Grauman et al., 2022; Hoque et al., 2026). As shown in Figure 2, long trajectories often contain multiple stages, and splitting them into subtasks makes them available as individual units for robot learning, providing fine-grained information. At this scale, manually annotating such splits becomes increasingly impractical, creating a need for automated splitting.

This growing need for automated splitting is not yet matched by benchmarks for current open-world robot data, where environments, objects, and operation skills vary naturally without fixed subtask structure. Existing benchmarks largely originate from video understanding tasks or constrained robot settings (Zappa et al., 2025), failing to jointly capture proprioception, visual complexity, and task diversity (Luo et al., 2025; Li et al., 2023). Efficient algorithms also remains challenging. Proprioception-based methods detect physical transitions precisely but lack semantic understanding (Lu et al., 2026; Duque-Domingo et al., 2025), whereas vision-based methods capture semantics but localize transitions less reliably; naive multimodal fusion rarely achieves both, often requires additional training, and open-world generalization remains unclear (Chen et al., 2026; Jiang et al., 2025). These motivate two questions: (1) How to evaluate subtask splitting on realistic open-world robot data; (2) How to effectively exploit complementary vision and proprioception for this task.

First, we introduce RoboSplitBench, a benchmark for subtask splitting in open-world robot data. As shown in Figure 1, it is built from demonstrations collected with a handheld device across 12 open-world scenes and over 50 operation skills, totaling 30 hours of data (about 2.5 hours per scene). Each trajectory provides end-effector (EEF) proprioception and multi-view observations. Human experts annotate subtask transitions under a unified protocol based on subject-predicate-object changes and task-independent temporal criteria. These properties enable systematic evaluation across diverse open-world scenes and transition types with or without proprioceptive events.

To provide a strong reference solution, we develop RoboSplit, a training-free splitting method that explicitly assigns complementary roles to proprioception and VLM reasoning. It first partitions the trajectory into proprioception-guided windows. Then it selectively adds wrist-view observations around gripper events to focus VLMs on fine-grained interactions. VLMs then predict subtask transitions within each window, with adjacent window seams re-split to recover the missed transitions. The proprioception organizes physical evidence, while VLM reasoning makes the splitting decisions, enabling efficient subtask splitting of long-horizon trajectories in open-world robot data.

We systematically evaluate RoboSplit against proprioception-based, vision-based, and multimodal baselines on RoboSplitBench. RoboSplit achieves the highest F1 score of 0.713, a 48.9% relative improvement over the strongest baseline, and remains robust across all 12 open-world scenes. Ablations confirm the effectiveness of individual components. Notably, without wrist-view input, RoboSplit achieves an F1 of 0.692, suggesting its potential for egocentric data without wrist cameras. Efficiency analyses reveal the effects of video FPS and image resolution, suggesting that preserving sampling rate while reducing resolution is preferable under constrained inference budgets.

Here, we summarize our contributions:

- We introduce RoboSplitBench, a benchmark for systematic evaluation of subtask splitting in open-world robot data, with diverse scenes, proprioception, and multi-view observations.
- We propose RoboSplit, a training-free algorithm that assigns complementary roles to proprioception and VLM reasoning, and systematically evaluate it on RoboSplitBench.

We make our benchmark data available through an anonymous link and provide an example data sample in supplementary materials (`example_data/`, compressed to meet size limit).

2 Related Works

Subtask Splitting Benchmarks Existing video-understanding benchmarks such as 50Salads (Stein & McKenna, 2013), GTEA (Fathi et al., 2011), and Breakfast (Kuehne et al., 2014) provide dense annotations of goal-directed human activities and manipulation interactions, but are not designed for robot trajectory splitting and generally lack synchronized robot end-effector and gripper states. GEBD (Shou et al., 2021) studies taxonomy-free event boundaries, but remains lacking robot settings. Robot-specific benchmarks reduce this gap but remain limited in complexity and diversity. They often focus on narrow domains, e.g., surgical robotics (Ahmidi et al., 2017; Xu et al., 2025), with homogeneous observations and limited procedures and subtasks. Recently, RoboSegNet (Chen et al., 2026) builds a splitting benchmark from existing robot datasets, but its trajectories are short, structured, and limited to predefined stages. In contrast, RoboSplitBench targets open-world demonstrations with diverse tasks, less standardized transitions, and varied visual environments.

Proprioception-based methods. A line of work detects subtask transitions from robot kinematics, force, or gripper events (Shi et al., 2023; Krishnan et al., 2017). CHAMP (Niekum et al., 2015) models changes in motion dynamics, GMM–Kalman (Karlsson et al., 2019) uses force and torque signals, and DexSkills (Mao et al., 2024) learns skill transitions from proprioceptive and haptic inputs. These methods capture physical transitions but may misalign with semantic subtasks.

Vision-based methods. Early vision-based methods, e.g., MS-TCN (Farha & Gall, 2019), ASFormer (Yi et al., 2021), and ASRF (Ishikawa et al., 2021), detect transitions from visual features but require task-specific training, while recent methods use frozen representations or foundation models (Zhang et al., 2024; Huang et al., 2024; Wang et al., 2025; Ma et al., 2025; Sharma et al., 2026). However, purely visual methods lack physical cues and remain sensitive to visual ambiguity.

Multimodal methods. Recent methods combine vision with proprioception. NILS (Blank et al., 2024) uses object-centric visual changes with gripper heuristics, while Suzuki et al. (2026) provides images and proprioception to VLMs. These combine modalities without specifying their roles. RoboSegNet (Chen et al., 2026) encodes visual and kinematic streams for prediction, but it requires annotated training data and generalization beyond the training embodiments remains unexplored.

3 RoboSplitBench

We introduce RoboSplitBench, a benchmark for splitting open-world robot data. We formulate the task, and then describe the data collection, splitting annotation protocol, and evaluation metric.

3.1 Problem Formulation

We consider a demonstration trajectory $\tau = (\ell, \mathcal{V}, \mathcal{P})$ of duration T, where ℓ denotes the task instruction for the entire trajectory, $\mathcal{V} = (\mathcal{V}^m, \mathcal{V}^l, \mathcal{V}^r)$ denotes the timestamped RGB observations from the main, left-wrist, and right-wrist views, and $\mathcal{P}$ denotes the synchronized EEF proprioception. A splitting annotation is represented by an ordered set of temporal subtask transition points $\mathcal{B}^* = \{b_1^*, \dots, b_K^*\}$, where $0 < b_1^* < \cdots < b_K^* < T$. Letting $b_0^* = 0$ and $b_{K+1}^* = T$, these transition points split the trajectory into $K+1$ consecutive subtasks $S_k^* = [b_{k-1}^*, b_k^*)$, $k = 1, \dots, K+1$.

Given τ, a splitting algorithm f predicts a variable-length ordered set of transition points $\hat{\mathcal{B}} = f(\tau) = \{\hat{b}_1, \dots, \hat{b}_{\hat{K}}\}$. RoboSplitBench evaluates the temporal accuracy of these predicted transitions using the human annotations and matching protocol described in Section 3.4.

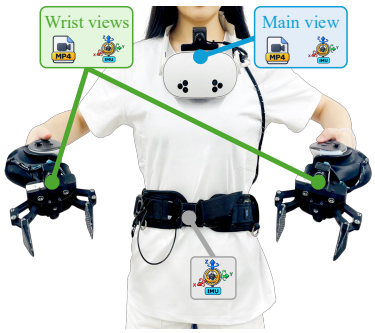

Figure 3: Data collection setup of RoboSplitBench using a universal handheld device.

3.2 Data Collection

The data in RoboSplitBench is collected using a universal handheld device, as shown in Figure 3. The device is a portable bimanual data-collection platform that records gripper states, end-effector motion, and videos while a human operator performs manipulation tasks. Each demonstration we released contains three synchronized 640×480 multi-view observations $\mathcal{V}$ at 30 fps and proprioception $\mathcal{P}$ at 30 Hz, including dual-arm end-effector poses and gripper states. The benchmark contains 585 trajectories across 12 open-world scenes, totaling about 30 hours. Detailed data statistics, including trajectory durations, operation skills, and the "proprioception events and transition points" correspondence are provided in Appendix C.

3.3 Splitting Annotation Protocol

Human experts split each trajectory into subtasks S_k^* under a unified protocol. Given extensive prior work on video captioning, RoboSplitBench focuses on subtask splitting; we describe the splitting protocol here and provide the action-description guidelines in Appendix B.

We characterize each subtask using a subject–predicate–object structure. The "*subject*" specifies the active arm, e.g., the left arm, or both arms; the "*predicate*" describes the operation, such as moving, wiping, or changing motion direction; and the "*object*" denotes the manipulated target or task-relevant region. Changes in these elements introduce candidate transitions.

However, predicate granularity is inherently ambiguous, as an interaction may be described as moving an object or decomposed into grasping, transporting, and placing. To avoid overly fine-grained splitting, we impose a minimum subtask duration of 3 seconds. Shorter units are merged with an adjacent subtask, guided by continuity in the active arm, manipulated target, and surrounding actions. For example, a grasping phase lasting less than 3 seconds may be merged with the subsequent transport of the same object. Longer units are not subject to mandatory merging under this rule. We acknowledge the inherent subjectivity of human splitting annotations, even under this constraint. To mitigate this effect, we engage two independent annotators for each demonstration, see Section 3.4.

Detailed splitting protocols and examples are provided in Appendix A. We also provide rules for failure and idle intervals in Appendix A.4 to avoid noise-induced spurious transitions.

3.4 Evaluation Metric

To reduce subjectivity in human annotation, each trajectory is independently annotated by two human experts. Using a ± 1-second tolerance, we first greedily match the transition points from the two annotations with nearest-neighbor one-to-one matching. A matched pair is treated as a "*consensus transition*", with its timestamp defined as the average of the two annotated timestamps. Transitions that remain unmatched in either annotation are treated as "*single-annotator transitions*".

We first match the predicted transition points $\hat{\mathcal{B}}$ to "*consensus transition*" within the same ± 1-second tolerance, then match the remaining predictions to "*single-annotator transitions*." Each predicted or annotated transition can be matched at most once. Precision is the proportion of predicted transitions matched either a "*consensus*" or "*single-annotator transition*", so predictions supported by either annotator are not penalized. Recall is computed only over "*consensus transitions*", measuring the proportion of transitions agreed upon by both annotators that are successfully identified. The F1 score is the harmonic mean of Precision and Recall and is used as the primary evaluation metric. **Notably,** unless otherwise specified as individual-annotator Precision, Recall, and F1 in Appendix D.1, all reported metrics collectively use annotations from both annotators following the protocol above.

The ± 1-second tolerance is the default evaluation setting, with other tolerance results reported in Appendix F. We use inter-annotator agreement for quality control, retaining only trajectories with an inter-annotator F1 of at least 0.6 at ± 1-second tolerance; details are provided in Appendix D, together with evaluations against each annotator independently.

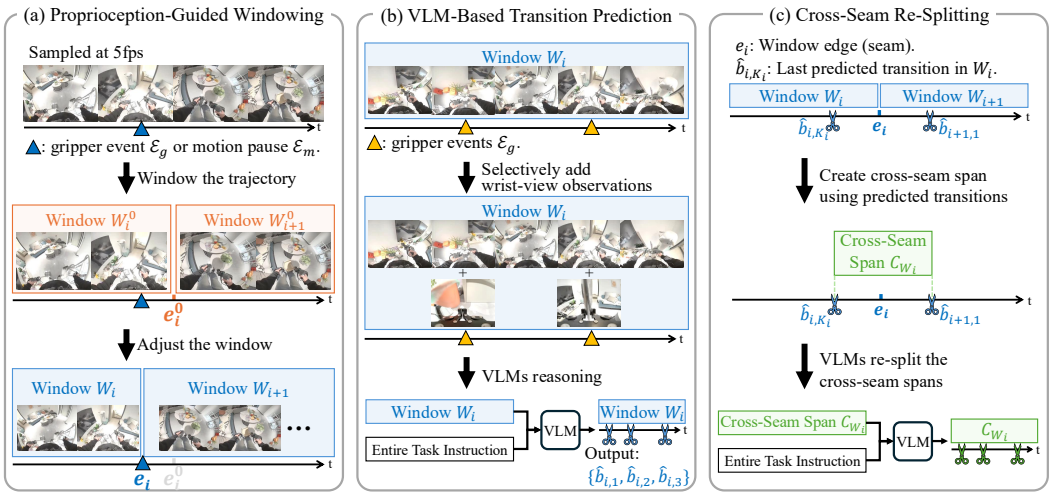

Figure 4: Overview of RoboSplit. The proprioception first guides temporal windowing. VLMs then identify subtask transition points within each window, selectively use wrist-view observations around gripper events, and re-split cross-seam spans to recover the missed transitions.

4 RoboSplit

Given a trajectory $\tau = (\ell, \mathcal{V}, \mathcal{P})$, RoboSplit predicts the subtask transition set $\hat{\mathcal{B}}$. The key idea is to assign complementary roles to proprioception and VLM reasoning. Proprioceptive changes provide temporally precise cues around physical interactions, but they do not necessarily correspond to subtask transitions. We therefore use proprioception to organize temporal windows and select task-relevant visual evidence, while leaving the final transition decisions to VLM reasoning.

As illustrated in Figure 4, RoboSplit consists of three stages. First, proprioception is used to adjust the trajectory windowing. Second, VLMs predict subtask transitions within each adjusted window, using the main-view stream together with selectively added wrist-view observations. Finally, cross-window-seam spans are re-split to recover transitions missed due to windowing.

4.1 Proprioception-Guided Windowing

As shown in Figure 4(a), we partition the trajectory into non-overlapping windows of length L. Let e_i^0 denote the i-th candidate window edge and $W_i^0 = [e_{i-1}^0, e_i^0)$ the corresponding window. Within each window, the main-view video is sampled at 5 fps. We set $L = 60$ s to balance VLM context and input length. Empirically, 30 s performs similarly, while 120 s degrades performance (Appendix K).

Fixed window edges may fall in the middle of an ongoing manipulation. We therefore extract gripper events $\mathcal{E}_g$ and motion pauses $\mathcal{E}_m$ from the proprioception to adjust the windows. Gripper events are sustained transitions in the smoothed gripper states, and motion pauses are prominent local minima of the smoothed sum of both end-effectors' translational speeds. For each e_i^0, we search within a ± 10 s neighborhood for the nearest valid event in $\mathcal{E}_g \cup \mathcal{E}_m$, allowing the window edge to align with nearby physical cues without drifting far from its nominal position. We analyze the sensitivity to this range in Appendix K. If one is found, the edge is shifted to its timestamp; otherwise, e_i^0 is retained. The resulting edge e_i defines the adjusted window $W_i = [e_{i-1}, e_i)$ for subsequent VLM inference.

4.2 VLM-Based Transition Prediction

As shown in Figure 4(b), for each adjusted window W_i, we sample the main-view observations $\mathcal{V}^m$ at 5 fps, obtaining a timestamped visual sequence. Then, for each gripper event within W_i, we find the nearest sampled timestamp and add the corresponding wrist-view observation from $\mathcal{V}^l$ or $\mathcal{V}^r$, yielding the augmented visual input. Given the trajectory instruction and visual input, the VLM predicts an ordered set of subtask transitions $\hat{\mathcal{B}}_i = \{\hat{b}_{i,1}, \ldots, \hat{b}_{i,K_i}\}$, where $\hat{b}_{i,j} \in W_i$. The complete VLM prompts are provided in the supplementary material (`prompt.txt`).

4.3 CROSS-SEAM RE-SPLITTING

Processing windows W_i and W_{i+1} independently may miss transitions near the seam e_i, where the context is separated across the two windows. As shown in Figure 4(c), for each seam e_i, we identify the last predicted transition $\hat{b}_{i,K_i}$ in W_i and the first predicted transition $\hat{b}_{i+1,1}$ in W_{i+1}, defining a cross-seam span $C_{W_i} = [\hat{b}_{i,K_i}, \hat{b}_{i+1,1}]$. VLMs predict additional transitions using the context from C_{W_i}. After processing all seams, the resulting transition set forms the final prediction $\hat{\mathcal{B}}$.

In summary, RoboSplit transforms the initial windows W_i^0 into proprioception-adjusted windows W_i using $\mathcal{E}_g \cup \mathcal{E}_m$, augments them with event-triggered wrist-view observations, predicts local transition sets $\hat{\mathcal{B}}_i$ with VLMs, and re-splits cross-seam spans C_{W_i} to recover missed transitions and obtain the final transition set $\hat{\mathcal{B}}$. We provide the pseudocode for RoboSplit in Appendix N.

5 EXPERIMENTS

We use RoboSplitBench to systematically evaluate representative approaches for subtask splitting in long-horizon open-world robot data and to examine the effectiveness of RoboSplit. Our experiments focus on four main questions: (1) How different splitting approaches perform on RoboSplitBench; (2) How their performance varies across open-world scenes and different transition types; (3) How each component and hyperparameter contributes to RoboSplit; (4) How sampling rate and image resolution affect the efficiency-performance trade-off under constrained inference budgets.

5.1 SETUPS

Evaluation metrics. We use the evaluation protocol in Section 3.4, which jointly uses two independent annotations with a ± 1-second matching tolerance. We report F1 as the primary metric, with Precision and Recall reported as complementary metrics. We further evaluate the sensitivity of all methods on RoboSplitBench to transition matching tolerance, revealing their ability to precisely localize transitions under strict tolerances and coarsely localize the correct transitions under looser ones, in Appendix F. And we also evaluate all methods against each annotator independently in Appendix D.1 to examine whether the two-annotator evaluation affects the comparative conclusions.

Implementation details. Unless otherwise specified, we use ByteDance Doubao-Seed-2.1-Pro (ByteDance Seed, 2026) as the default VLM, with thinking disabled during inference. To assess the dependence on the underlying VLM, we additionally evaluate different VLM models and thinking modes in Appendix L. We further analyze the sensitivity to two hyperparameters, the temporal window length and the search range for proprioception-guided window adjustment, in Appendix K. Note that RoboSplit was developed on separate scenes outside RoboSplitBench, with the method and default configuration fixed before evaluation on all 12 scenes of RoboSplitBench; see Appendix M.

Baselines. We first construct four controlled baselines to examine different uses of proprioception and VLMs. "*State Only*" detects transitions solely from proprioceptive events, including gripper events and motion pauses, without visual input. "*Whole-Video VLMs*" predicts subtask transitions over the entire video using only visual observations and the task instruction. "*State-Prompted VLMs*" converts structured proprioceptive events into timestamped textual cues and provides them directly to the VLM. "*State-to-VLMs Refine*" first obtains candidate transitions from "*State Only*" and then uses a VLM to merge adjacent segments according to their semantic action stages.

We also compare with representative methods, including UVD (Zhang et al., 2024), which recursively detects peaks in visual embedding distances; and Suzuki et al. (2026), which identifies transitions from similarity between captions of sampled visual-proprioceptive inputs. Detailed implementations of both controlled and open-source baselines are provided in Appendix E.

5.2 OVERALL PERFORMANCE ON ROBOSPLITBENCH

We first compare RoboSplit with representative baselines on RoboSplitBench, reporting overall and scene-wise performance across all 12 scenes. Each method is evaluated over five stochastic runs, with error bars indicating one standard deviation. Beyond comparing overall performance, this evaluation reveals how different splitting methods behave across diverse open-world scenes.

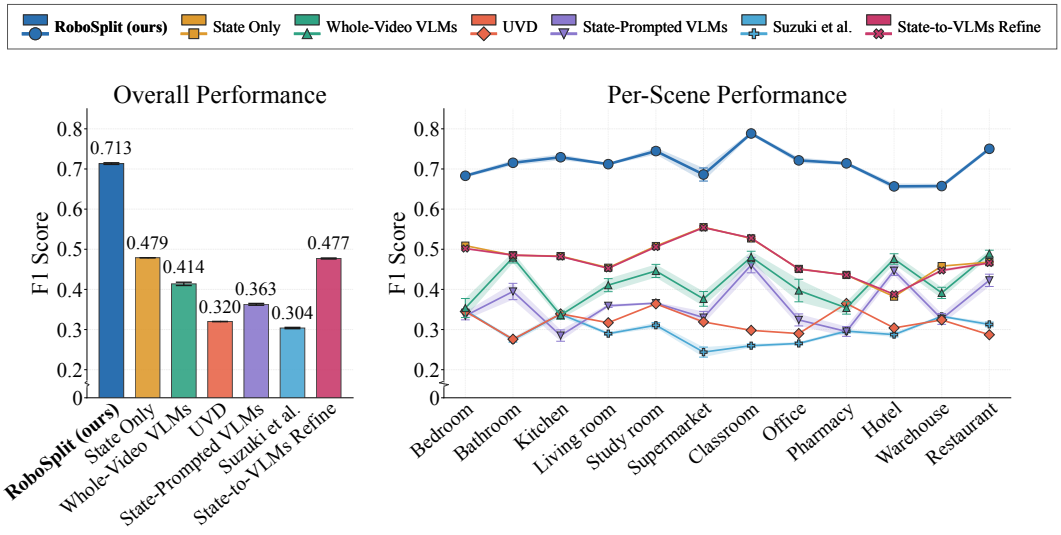

Figure 5: Overall and scene-wise performance on RoboSplitBench. RoboSplit achieves the highest F1 score overall and consistently outperforms baseline methods across all 12 open-world scenes.

Figure 5 shows clear performance differences among existing splitting approaches on RoboSplitBench. **Overall performance.** RoboSplit outperforms all competing methods, achieving an overall F1 score of 0.713 compared with 0.479 for the strongest baseline. **Cross-scene robustness.** RoboSplit consistently outperforms the competing methods across all 12 unseen scenes. **Run-to-run stability.** Across five independent runs, RoboSplit also exhibits lower run-to-run variance than the "*Whole-Video VLMs*" baseline, indicating more stable performance across repeated VLM inference.

Table 1: Overall Precision and Recall performance across all 12 scenes in RoboSplitBench.

	RoboSplit (ours)	State Only	Whole-Video VLMs	UVD [39]	State-Prompted VLMs	Suzuki et al. [33]	State-to-VLMs Refine
Precision ↑	0.763	0.477	0.551	0.192	0.556	0.225	0.490
Recall ↑	0.669	0.480	0.331	0.953	0.269	0.469	0.465

Precision–Recall trade-off. Table 1 further reveals distinct failure modes of different splitting paradigms on RoboSplitBench. Different methods exhibit distinct Precision–Recall trade-offs, with some favoring Recall and others Precision. In contrast, RoboSplit achieves a better balance between the Precision and Recall. Scene-wise Precision and Recall results are reported in Appendix G.

Proprioception-aware evaluation. Beyond aggregate and scene-wise scores, RoboSplitBench exposes a key mismatch between physical events and semantic transitions. Many valid transitions lack nearby proprioceptive events, while many physical events do not correspond to a split. This enables capability-level diagnosis of whether a method can recover semantic transitions without salient physical cues while suppressing false splits from non-semantic events; see Appendices I and J.

5.3 Ablation Studies

We ablate each key component of RoboSplit individually while keeping all others fixed. Specifically, "*w/o Proprio.-guided Windowing*" replaces proprioception-guided window edge adjustment with uniformly fixed 60-second windows; "*w/o Selective Wrist View*" removes all wrist-view observations and uses only the main view observations; and "*w/o Cross-seam Re-splitting*" only output predictions from individual windows without the cross-seam re-splitting step to recover the missed transitions. All variants are evaluated using the same protocol as the full method.

As shown in Table 2, all components provide complementary gains. The fixed-window VLM variant already achieves an F1 of 0.614, suggesting that temporal windowing itself contributes to performance. Without cross-seam re-splitting, F1 reaches 0.661, demonstrating the benefit of our com-

Table 2: Ablation of the key components in RoboSplit on RoboSplitBench.

Proprio.-guided Windowing	Selective Wrist View	Cross-seam Re-splitting	Precision ↑	Recall ↑	**F1 ↑**
×	×	×	0.684	0.557	**0.614**
×	✓	✓	0.760	0.574	**0.654**
✓	×	✓	0.760	0.636	**0.692**
✓	✓	×	0.771	0.579	**0.661**
✓	✓	✓	0.763	0.669	**0.713**

plementary multimodal design. The full method reaches 0.713; removing proprioception-guided windowing, selective wrist views, or cross-seam re-splitting reduces F1 to 0.654, 0.692, and 0.661, respectively, with the larger drops from windowing and re-splitting mainly due to lower Recall.

5.4 View Configurations

We further investigate how wrist-view observations should be provided to VLMs. We compare four strategies while keeping other components unchanged. "*Main View Only*" removes the wrist views entirely. "*All Wrist Views*" provides both wrist views throughout the whole trajectory. "*Both Wrists at Gripper Events*" provides both wrist views whenever a gripper event occurs. In contrast, RoboSplit selectively provides only the wrist view corresponding to the gripper where the event occurs. This comparison evaluates whether proprioceptive events can effectively guide selective wrist-view use while reducing redundant visual observations.

Table 3: Comparison of different wrist-view selection configurations on RoboSplitBench.

Wrist-view Configuration	Input Tokens ↓	Precision ↑	Recall ↑	**F1 ↑**
Main View Only	124M	0.760	0.636	**0.692**
All Wrist Views	262M	0.784	0.670	**0.723**
Both Wrists at Gripper Events	167M	0.761	0.668	**0.712**
RoboSplit	153M	0.763	0.669	**0.713**

As shown in Table 3, RoboSplit performs similarly to "*Both Wrists at Gripper Events*" while using fewer input tokens, indicating that providing only the corresponding wrist view is often sufficient. Compared with "*All Wrist Views*", which achieves the highest F1 score, RoboSplit maintains comparable performance with 41.6% fewer input tokens. These results show that proprioception-guided wrist-view selection provides a favorable trade-off between splitting performance and input cost.

Prospects for egocentric data. In RoboSplit, wrist-view observations provide additional complementary close-up evidence for resolving fine-grained interactions, while we also examine whether the method remains effective when dedicated wrist-view observations are unavailable. As shown in Table 3, the main-view-only setting retains relatively strong performance, achieving an F1 score of 0.692 despite removing the dedicated wrist-view observations entirely. This property is especially relevant to emerging large-scale egocentric datasets, which typically provide only a primary first-person view and lack dedicated wrist cameras. Together with recent approaches for recovering proprioception from egocentric recordings (Lin et al., 2026; Niu et al., 2026; Hoque et al., 2026), these results suggest that RoboSplit could potentially be extended to egocentric embodied data.

5.5 Efficiency Analysis

We further analyze the efficiency–performance trade-off of RoboSplit with respect to two key visual input configurations, the sampling rate and image resolution. These two factors control the temporal density and spatial detail of visual observations, respectively, and directly affect both splitting performance and inference cost. In addition to F1 score, we measure the input-token consumption and inference time, and normalize both costs relative to our default settings, i.e., 5 fps and 640×480 resolution. Under this default configuration, the full benchmark requires approximately 153M input tokens (using Doubao-Seed-2.1-Pro) and 6,200 seconds of total inference time.

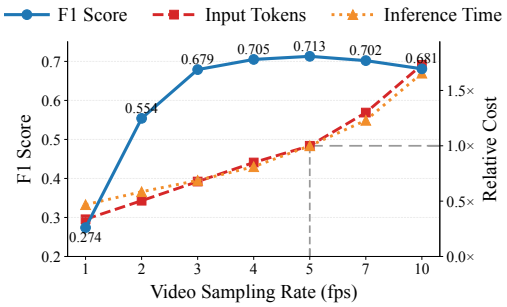

Figure 6: Sampling rate efficiency trade-off.

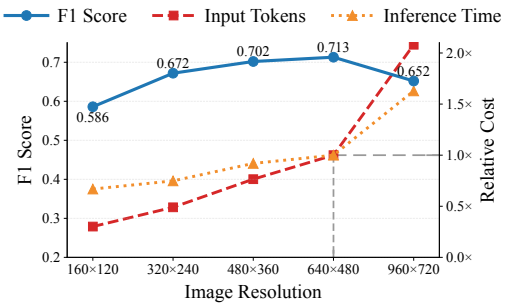

Figure 7: Image resolution efficiency trade-off.

Sampling Rate. As shown in Figure 6, increasing the sampling rate substantially improves performance at low rates, with the highest F1 score at 5 fps. Further increasing it to 7 or 10 fps slightly degrades performance, while both input tokens and inference time continue to increase. This may be because excessively dense visual sampling introduces substantial redundant information, making it harder for the VLM to focus on the most relevant subtask transitions. We adopt 5 fps as a favorable operating point, with the best F1 score at moderate inference cost.

Image Resolution. As shown in Figure 7, increasing image resolution improves F1 from 0.586 to 0.713 at 640×480, with diminishing gains at higher resolutions. Increasing the resolution further to 960×720 (using original camera streams rather than upsampling) reduces F1 to 0.652, while substantially increasing both input tokens and inference time. This suggests that additional spatial detail beyond 640×480 offers limited value for subtask splitting relative to its computational cost. We adopt 640×480 as the default resolution, with the highest F1 score at moderate inference cost.

Comparing the two efficiency analyses reveals that, under similar input-token budgets, reducing image resolution preserves substantially better splitting performance than reducing the sampling rate. This suggests that temporal coverage is more important than fine spatial detail under constrained input budgets in our tested configurations. Therefore, under constrained inference budgets, we recommend reducing image resolution before lowering the sampling rate.

Optional calibration under low sampling rates. For further efficiency at lower sampling rates, sparse visual observations can make precise subtask transition localization more challenging. We therefore optionally calibrate VLM-predicted transitions using nearby proprioceptive events. This calibration is most effective at low sampling rates, with diminishing gains as sampling becomes denser. We apply it only in the FPS analysis, since proprioceptive cues provide high-frequency temporal information but cannot recover spatial detail lost at lower image resolutions. Detailed calibration procedures and experiment results are provided in Appendix H.

6 Conclusion

We introduce RoboSplitBench, a benchmark for subtask splitting in open-world robot data. It provides real-world demonstrations with proprioception, multi-view observations, and human-annotated subtask transition points, enabling systematic evaluation across diverse open-world scenes and transition types. Building on this benchmark, we propose RoboSplit, a training-free splitting method that assigns complementary roles to proprioception and VLM reasoning. Proprioception organizes temporal windows and task-relevant visual evidence, while VLMs determine where to split. We systematically evaluate different methods on RoboSplitBench, and RoboSplit consistently outperforms representative baselines and remains robust across diverse scenes. Our analyses further reveal the effects of visual input efficiency, suggesting that preserving temporal sampling can be considered before preserving image resolution under constrained inference budgets. We hope RoboSplitBench and RoboSplit provide a foundation for systematically studying and advancing subtask splitting as robot datasets continue to grow in scale and diversity.

Future Work. The current version of RoboSplitBench is primarily designed for systematic evaluation. In future work, we plan to expand it with more trajectories and real-world scenes, supporting training methods and more comprehensive evaluation of generalization.

AI USE STATEMENT

Large Language Model tools were used to assist with language editing and revision of the manuscript. All AI-assisted contents were critically reviewed by the authors. The experimental results were obtained from author-conducted experiments, and the final methodological choices, analyses, interpretations, and claims were determined and verified by the authors. The authors take full responsibility for the content of this work.

ETHICS STATEMENT

RoboSplitBench consists of real-world robot demonstrations collected in open-world environments, where people may occasionally appear in the recorded videos. To reduce privacy risks, visible human faces in all released videos are anonymized using mosaic blurring. No demographic attributes or other personal information are used in the annotations or evaluation, and the benchmark is not designed for identifying, tracking, or profiling individuals.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we release RoboSplitBench through an anonymous link. We also provide the complete prompts used for VLM inference in the supplementary materials, and detail implementation settings and the pseudocode of RoboSplit in our paper and Appendix.

REFERENCES

Narges Ahmidi, Lingling Tao, Shahin Sefati, Yixin Gao, Colin Lea, Benjamin Bejar Haro, Luca Zappella, Sanjeev Khudanpur, René Vidal, and Gregory D Hager. A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering*, 64(9):2025–2041, 2017.

Nils Blank, Moritz Reuss, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Wenzel, Oier Mees, and Rudolf Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. *arXiv preprint arXiv:2410.17772*, 2024.

ByteDance Seed. Seed2.1 model card: Agentic intelligence for productivity. Technical report, ByteDance, 2026. URL https://lf3-static.bytednsdoc.com/obj/eden-cn/lapzild-tss/ljhwZthlaukjlkulzlp/seed2.1/Seed2_1_Model_Card.pdf.

Kaiyuan Chen, Shuangyu Xie, Andrew Goldberg, and Ken Goldberg. Learning multi-task robot trajectory segmentation from visual and kinematic streams. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 1452–1461, June 2026.

Jaime Duque-Domingo, Riccardo Caccavale, Alberto Finzi, Eduardo Zalama, and Jaime Gómez-García-Bermejo. One-shot learning for rapid generation of structured robotic manipulation tasks from 3d video demonstrations. *Journal of Intelligent Manufacturing*, pp. 1–24, 2025.

Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3575–3584, 2019.

Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *2011 international conference on computer vision*, pp. 407–414. IEEE, 2011.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18995–19012, 2022.

Ryan Hoque, Peide Huang, David Yoon, Jian Zhang, et al. Egodex: Learning dexterous manipulation from large-scale egocentric video. In *International Conference on Learning Representations*, volume 2026, pp. 4218–4237, 2026.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2321–2330. IEEE, 2021.

Jiawei Jiang, Kei Ota, Devesh K Jha, and Asako Kanezaki. Modality selection and skill segmentation via cross-modality attention. In *2025 19th International Conference on Machine Vision and Applications (MVA)*, pp. 1–6. IEEE, 2025.

Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5961–5968, 2025.

Martin Karlsson, Anders Robertsson, and Rolf Johansson. Segmentation of robot movements using position and contact forces. *arXiv preprint arXiv:1909.08289*, 2019.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel, and Ken Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. *The International journal of robotics research*, 36(13-14):1595–1618, 2017.

Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 780–787, 2014.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023.

Xingyao Lin, Guojin Zhong, Tianyi Lu, Ziyi Ye, Yichen Zhu, Zuxuan Wu, and Yu-Gang Jiang. Activemimic: Egocentric video pretraining with active perception. *arXiv preprint arXiv:2606.06194*, 2026.

Yupu Lu, Hang Xu, Yizhou Chen, and Jia Pan. Keypose exploration: Efficient automatic trajectory labelling and cross-embodiment policy transfer. *arXiv preprint arXiv:2606.29028*, 2026.

Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 44(4):592–606, 2025.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025.

Xiaofeng Mao, Gabriele Giudici, Claudio Coppola, Kaspar Althoefer, Ildar Farkhatdinov, Zhibin Li, and Lorenzo Jamone. Dexskills: Skill segmentation using haptic data for learning autonomous long-horizon robotic manipulation tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5104–5111. IEEE, 2024.

Scott Niekum, Sarah Osentoski, Christopher G Atkeson, and Andrew G Barto. Online bayesian changepoint detection for articulated motion models. In *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 1468–1475. IEEE, 2015.

Yichen Niu, Haoran Lv, Xinrui Zhang, Xueyao Wan, Shiyu Gao, Ying Ai, Hui Xu, Yongqi Hu, Hengyi Zhang, Yang Xie, et al. Egoaero: Learning dexterous manipulation from a single egocentric video without object assets. *arXiv preprint arXiv:2606.08057*, 2026.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. Technical report, OpenAI, 2026. URL https://openai.com/index/gpt-5-6/.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Ryan Punamiya, Simar Kareer, Zeyi Liu, Josh Citron, Ri-Zhao Qiu, Xiongyi Cai, Alexey Gavryushin, Jiaqi Chen, Davide Liconti, Lawrence Y Zhu, et al. Egoverse: An egocentric human dataset for robot learning from around the world. *arXiv preprint arXiv:2604.07607*, 2026.

Dharmendra Sharma, Archit Sharma, John Rebeiro, Vaibhav Kesharwani, Peeyush Thakur, Narendra Kumar Dhar, and Laxmidhar Behera. Robosubtasknet: Temporal sub-task segmentation for human-to-robot skill transfer in real-world environments. *arXiv preprint arXiv:2602.10015*, 2026.

Lucy Xiaoyang Shi, Archit Sharma, Tony Z Zhao, and Chelsea Finn. Waypoint-based imitation learning for robotic manipulation. *arXiv preprint arXiv:2307.14326*, 2023.

Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8055–8064. IEEE, 2021.

Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 729–738, 2013.

Kanata Suzuki, Shota Shimizu, and Tetsuya Ogata. Proprioception enhances vision language model in generating captions and subtask segmentations for robot task. In *2026 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1–7. IEEE, 2026.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Beichen Wang, Juexiao Zhang, Shuwen Dong, Irving Fang, and Chen Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 17215–17222. IEEE, 2025.

Haozheng Xu, Alistair Weld, Chi Xu, Alfie Roddan, João Cartucho, Mert Asim Karaoglu, Alexander Ladikos, Yangke Li, Yiping Li, Daiyun Shen, et al. Surgripe challenge: Benchmark of surgical robot instrument pose estimation. *Medical Image Analysis*, 105:103674, 2025.

Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. *arXiv preprint arXiv:2110.08568*, 2021.

Isacco Zappa, Alessandro Tomasoni, Andrea Maria Zanchettin, and Paolo Rocco. Robust segmentation and classification of robot skills from demonstrations with detection of unknown skills. *Procedia Computer Science*, 253:1535–1544, 2025.

Zichen Zhang, Yunshuang Li, Osbert Bastani, Abhishek Gupta, Dinesh Jayaraman, Yecheng Jason Ma, and Luca Weihs. Universal visual decomposer: Long-horizon manipulation made easy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6973–6980. IEEE, 2024.

APPENDIX

A DETAILED SPLITTING ANNOTATION PROTOCOL

In this section, we present detailed splitting annotation guidelines used to construct the ground-truth subtask in RoboSplitBench. The examples below illustrate representative situations that naturally occurred in the collected open-world data.

A.1 SUBTASK TRANSITION RULES

Rule 1: Subject–Predicate–Object Change

Each subtask is interpreted using a subject–predicate–object structure.

- The **subject** specifies the active arm configuration, such as the left arm, the right arm, or both arms.
- The **predicate** specifies the manipulation action or motion pattern, such as moving, wiping, rotating, pushing, pulling, or changing the direction of motion.
- The **object** specifies the manipulated object or task-relevant region.

Changes in these elements introduce candidate transitions. Candidate transitions are retained subject to the minimum-duration constraint in Rule 2.

Example: Change of Subject

The following occurring transitions indicate changes in the active embodiment configuration:

- the left arm is replaced by the right arm;
- a single-arm operation changes to a bimanual operation;
- a bimanual operation changes to a single-arm operation.

Example: Change of Predicate

Representative predicate changes observed in the collected data include:

- picking or placing changes to wiping;
- pushing changes to pulling;
- wiping a table vertically changes to wiping it horizontally.

A predicate change may arise not only from a different action category, but also from a persistent change in the motion pattern or trajectory direction.

Example: Change of Object or Target Region

Representative object-level transitions include:

- manipulating a towel changes to manipulating a cup;
- wiping a door changes to wiping a wall;
- wiping the interior of a cabinet changes to wiping its frame.

A task-relevant region may therefore function as the object of a subtask, even when the physical object itself remains unchanged.

A.2 Temporal Granularity Rule

Rule 2: Minimum Segment Duration

A subtask is retained as an independent segment only when it lasts for at least 3 seconds. This task-agnostic temporal criterion is used to resolve the inherent ambiguity of predicate granularity. For example, the same interaction may be described coarsely as moving an object or decomposed into picking it up, transferring it, and placing it. The 3-second criterion prevents brief transitional changes from producing unstable or excessively fine-grained annotations.

Example: Brief Transitional Motion

When a candidate subtask lasts for less than 3 seconds, it is not retained as an independent segment. Instead, it is merged with the preceding or following segment according to semantic continuity. The neighboring segment is selected based on which merge produces the more coherent description of the complete manipulation process.
During a manipulation process, an operator may briefly adjust the arm direction before continuing the same task. Although the motion predicate changes momentarily, the adjustment is merged with the surrounding segment when it lasts for less than 3 seconds.

A.3 Semantic Completeness

Rule 3: Semantic Continuity for Short-Unit Merging

When a candidate transition produces a subtask shorter than the 3-second minimum duration in Rule 2, the short unit is merged with an adjacent subtask according to semantic continuity. We consider continuity in the active arm, manipulated target, and surrounding actions when determining the merging direction.
This rule applies only to subtasks shorter than 3 seconds. Subtasks that already satisfy the minimum duration are not mandatorily merged.

Example: Pick-and-Place with Unequal Stage Durations

Consider a pick-and-place process consisting of:

$$\text{grasp object (1 s)} \rightarrow \text{transport object (2 s)} \rightarrow \text{place object (1 s)}.$$

All three stages are shorter than 3 seconds and are therefore merged according to semantic continuity. Since they form a continuous manipulation of the same object, they are combined into a single 4-second subtask.

A.4 Failures And Idle Intervals

We further define explicit rules for unsuccessful and idle motions. A failed attempt immediately followed by a successful repetition of the same action is grouped with the successful execution and described using the intended successful action. If an abandoned failure lasts less than 3 seconds, it is merged with an adjacent valid segment; if it lasts at least 3 seconds, it is isolated and marked as "*unannotatable*". Similarly, idle or semantically unclear intervals at least 3 seconds are annotated as separate unannotatable segments. These rules prevent transient execution noise from producing excessive transitions while preserving substantial failures and inactive periods in the ground-truth segmentation. The following are some examples:

Rule 4: Failed Attempt Followed by Success

The brief failure is treated as execution variation within the same semantic goal rather than as an independent subtask. When a failed attempt is immediately followed by a successful repetition of the same action, the failed and successful attempts are grouped into one segment. The segment is described according to the successfully completed action.

Example: Object Dropped and Grasped Again

The right arm attempts to pick up a pen, drops it because of an unstable grasp, and immediately grasps it again successfully.
The entire process is annotated as one segment and described as "pick up the pen with the right arm." The unsuccessful first attempt does not produce an additional subtask transition.

Rule 5: Abandoned Failures Lasting at Least 3 Seconds

If an unsuccessful and subsequently abandoned operation lasts for at least 3 seconds, it is retained as an independent segment and assigned the label "*unannotatable*."
Although the interval has sufficient temporal extent to remain independent, it does not correspond to a successfully completed and semantically reliable manipulation goal.

Example: Extended Unsuccessful Grasp

The right arm repeatedly attempts to pick up a pen, fails to secure it, and eventually abandons the operation after more than 3 seconds.
This interval is isolated as a separate segment and labeled "*unannotatable*." The subsequent meaningful action begins a new segment.

Rule 6: Idle Content

Idle intervals, unexplained movements, and other semantically ambiguous content lasting for at least 3 seconds are retained as separate segments and labeled "*unannotatable*." Shorter ambiguous intervals are merged with an adjacent segment when possible.

Example: Extended Idle Interval

The device remains stationary for more than 3 seconds between two manipulation steps without exhibiting a clear task intention.
The stationary interval is annotated separately as "*unannotatable*," rather than being assigned to either neighboring subtask.

B Action Description Annotation Protocol

For each valid subtask, annotators follow a structured but open-ended description template: "*[left gripper / right gripper / both grippers] + [action] + [manipulated object]*." When two objects are involved, the template is extended to specify the operated object, the action, and the target object. The template constrains only the essential semantic roles, while annotators describe the observed objects, motion directions, rotation angles, spatial relations, and object indices as needed, rather than selecting from a predefined action vocabulary.

C Detailed Dataset Information

Data collection and diversity. RoboSplitBench is constructed from handheld demonstrations collected in open-world environments. The dataset covers 12 diverse scenes and 50+ operation skills, amounting to about 30 hours of demonstrations (about 2.5 hours for each scene). A detailed list of representative operation skills observed in the dataset, together with their empirical frequencies, is provided in supplementary materials (`operation_skills.xlsx`). Compared with datasets collected in highly controlled laboratory settings, our data exhibit substantial variations in scene layout, object appearance, background clutter, camera motion, and task execution. The recorded trajectories contain both relatively simple interactions and long-horizon tasks composed of multiple semantically distinct stages, making them well suited for evaluating automatic subtask splitting.

To further quantify operation diversity, we extract canonical action–object pairs from episode-level task descriptions. Across 585 trajectories, we obtain 485 distinct pairs. Among them, 85.2% occur in only one scene, while the mean and median pairwise Jaccard similarities between scenes are only 0.034 and 0.021, respectively. Moreover, the ten most frequent pairs account for only 21.5% of all occurrences, indicating a pronounced long-tail distribution. These statistics show that the 12 scenes differ not only visually, but also in their action–object compositions, rather than repeatedly instantiating a small set of skills in different backgrounds.

Trajectory duration distribution. The trajectories in RoboSplitBench are generally long and exhibit substantial variation in duration. The average trajectory duration is $186.9\,\mathrm{s}$, while the median duration is $195.0\,\mathrm{s}$. The first and third quartiles are $134.7\,\mathrm{s}$ and $249.3\,\mathrm{s}$, respectively, meaning that the middle 50% of trajectories span approximately 2.25–4.15 minutes. These statistics indicate that the benchmark is dominated by multi-minute trajectories rather than short, isolated manipulation clips. Such long-horizon demonstrations typically contain multiple semantic stages, making accurate transition localization and semantic decomposition particularly important.

Relationship between proprioception and annotated transitions. We further examine the relationship between human-annotated subtask transitions and proprioceptive events, including gripper events and motion pauses. Across RoboSplitBench, the two independent annotations contain 13,316 and 12,530 transition points, respectively, while the proprioceptive signals yield 35,637 events. Using a ± 1-second one-to-one matching tolerance, **67.95%** (9,048) and **67.42%** (8,448) of the transitions in the two annotations can be matched to a proprioceptive event, respectively. In contrast, only **25.39%** (9,048) and **23.71%** (8,448) of the proprioceptive events can be matched to an annotated transition. These statistics show that RoboSplitBench cannot be reduced to detecting gripper events or motion pauses: a substantial fraction of valid transitions require cues beyond salient proprioceptive changes, while most physical events should not trigger a split. The benchmark therefore tests not only temporal event localization, but also the ability to distinguish task-relevant subtask transitions from physical changes occurring within the same subtask.

D Benchmark Construction by Inter-Annotator Agreement

Data in RoboSplitBench is selected from a substantially larger pool of demonstration trajectories. Each candidate trajectory is independently annotated by two human experts following the same protocol described in Section 3.3. To quantify annotation consistency, we match the two sets of internal subtask transition points within a temporal tolerance of ± 1 second using one-to-one matching, and compute Precision, Recall, and F1 between the two annotations.

Only trajectories with an inter-annotator F1 score of at least **0.6** are included in RoboSplitBench. We use this moderate agreement threshold for annotation quality control, excluding trajectories whose transition structure is highly inconsistent between annotators while retaining the natural ambiguity of subtask splitting. The resulting benchmark contains 585 trajectories across 12 open-world scenes.

Under the default ± 1-second matching tolerance, the two annotations contain **13,316** and **12,530** unique transition timestamps, respectively. Their matching yields **10,166** consensus transitions and **5,514** single-annotator transitions, including **3,150** annotator-1-only and **2,364** annotator-2-only transitions. Accordingly, **76.3%** of annotator-1 transitions and **81.1%** of annotator-2 transitions are supported by the other annotator. Among the matched transition pairs, the absolute temporal difference is **0.298 s** on average and **0.233 s** at the median.

D.1 Evaluation Against Individual Annotators

Our primary evaluation metric explicitly accounts for annotation ambiguity by treating transitions supported by either annotator as valid for Precision, while computing Recall over transitions agreed upon by both annotators. To examine whether this reference construction affects the comparative conclusions, we additionally evaluate all methods using each annotator independently as the sole reference. Under this setting, we use the conventional transition Precision, Recall, and F1.

As shown in Table 4, the main comparative conclusions remain stable when either annotator is independently treated as the reference. RoboSplit achieves the highest F1 under both annotations, with

Table 4: Evaluation against each annotator independently using conventional transition Precision, Recall, and F1.

Metric	RoboSplit	State Only	Whole-Video VLMs	UVD	State-Prompted VLMs	Suzuki et al.	State-to-VLMs Refine
Annotator 1 Precision ↑	0.690	0.420	0.495	0.164	0.503	0.194	0.431
Annotator 1 Recall ↑	0.555	0.446	0.279	0.952	0.227	0.465	0.431
Annotator 1 F1 ↑	0.615	0.433	0.357	0.280	0.312	0.273	0.431
Annotator 2 Precision ↑	0.696	0.399	0.511	0.154	0.510	0.180	0.411
Annotator 2 Recall ↑	0.594	0.451	0.306	0.947	0.244	0.459	0.436
Annotator 2 F1 ↑	0.641	0.423	0.383	0.265	0.330	0.259	0.423

scores of 0.615 and 0.641, respectively. The relative behavior of the baselines is also largely preserved: State Only and State-to-VLMs Refine remain the strongest baselines, while UVD maintains very high Recall but substantially lower Precision.

The absolute scores are lower than those obtained with our primary benchmark metric, which is expected from the different treatment of annotation disagreement. When a single annotator is used as the sole reference, a predicted transition supported only by the other annotator is counted as a false positive, even though it may represent a plausible subtask transition. Moreover, annotator-specific transitions are included in the Recall denominator, whereas our primary metric computes Recall only over transitions agreed upon by both annotators. The individual-annotator evaluation is therefore stricter in absolute score, but the consistent relative performance across both references indicates that the main conclusions are not driven by our consensus-aware reference construction.

E Details of Controlled Baselines

State Only. This baseline performs splitting using only proprioception. Gripper events serve as primary cues and are aligned to nearby motion pauses; motion pauses are used in long intervals without gripper events. The results are temporally filtered to avoid overly dense predictions.

Whole-Video VLMs. This baseline performs subtask splitting using VLM reasoning alone, without access to any proprioception. The entire video, sampled at the same rate as RoboSplit, together with the high-level task instruction, is directly provided to the VLM in a single pass. The VLM predicts subtask transitions solely from the visual observations and semantic context, without proprioception-guided windowing, prompting, or transition calibration.

State-Prompted VLMs. This baseline injects proprioception into VLM reasoning as timestamped textual hints. Structured events, including gripper events, motion pauses, and motion-pattern transitions, are extracted from proprioception and converted into timestamped descriptions. These hints are provided with the video and task instruction to support subtask splitting.

State-to-VLMs Refine. This baseline starts from the transitions produced by *State Only* and uses VLMs for post-hoc refinement. For each transition, up to 15 seconds from each adjacent segment are separately described by the VLM, and a subsequent textual judgment decides whether the two sides represent distinct semantic stages. The transition is retained if they differ and merged otherwise.

Open-source baselines. Since RoboSplit was developed outside RoboSplitBench without benchmark-specific tuning, we follow the same principle for competing methods. For UVD (Zhang et al., 2024) and Suzuki et al. (2026), we directly use their reported hyperparameters. We directly reproduce UVD following its official implementation, using the frozen VIP visual encoder on all original 30-FPS frames, with $min_{\text{interval}} = 18$, RBF smoothing with $\gamma = 0.08$; For Suzuki et al. (2026), we use the same VLM as RoboSplit, align each sampled image with the corresponding Cartesian arm and gripper states, encode the resulting captions with `stsb-xlm-r-multilingual`, and use the reported similarity threshold of 0.65 for segmentation.

We exclude RoboSegNet (Chen et al., 2026), which requires annotated training data, while its generalization beyond training embodiments remains unexplored. We also exclude NILS (Blank et al., 2024), whose object-centric keystate detection requires heuristics and embodiment-specific settings, leaving no unique reported configuration for a tuning-free evaluation on our embodiment.

Table 5: F1 scores under different transition matching tolerances on RoboSplitBench. In the standard setting, reference sets are reconstructed at each tolerance; Fixed Reference keeps the reference sets constructed at the default ± 1 s tolerance.

Matching Tolerance	**RoboSplit (ours)**	State Only	Whole-Video VLMs	UVD	State-Prompted VLMs	Suzuki et al.	State-to-VLMs Refine
± 0.5 s	**0.516**	0.328	0.251	0.258	0.207	0.216	0.327
± 1 s (RoboSplitBench)	**0.713**	0.479	0.414	0.320	0.363	0.304	0.477
± 2 s	**0.795**	0.665	0.559	0.322	0.512	0.395	0.662
± 0.5 s (fixed reference)	**0.491**	0.312	0.249	0.238	0.202	0.201	0.311
± 2 s (fixed reference)	**0.801**	0.669	0.567	0.337	0.526	0.407	0.666

F Evaluation under Different Matching Tolerances

Table 5 evaluates the performance under different matching tolerances. In the standard setting, the consensus and single-annotator reference sets are reconstructed at each tolerance, while the fixed-reference setting keeps the references constructed at the default ± 1 s tolerance and varies only prediction matching. The consistent trends across both settings show that varying the tolerance provides a complementary view of behavior: stricter tolerances emphasize precise temporal localization, whereas looser tolerances reveal whether methods coarsely identify the correct transitions.

G Scene-Wise Precision and Recall

To complement the overall Precision and Recall results reported in Table 1, we provide scene-wise results in Table 6, across all 12 scenes in RoboSplitBench.

Table 6: Scene-wise transition Precision and Recall across all 12 scenes in RoboSplitBench.

Scene	Metric	RoboSplit (ours)	State Only	Whole-Video VLMs	UVD	State-Prompted VLMs	Suzuki et al.	State-to-VLMs Refine
Bedroom	Precision	0.766	0.499	0.569	0.211	0.572	0.242	0.510
	Recall	0.616	0.519	0.257	0.945	0.236	0.617	0.494
Bathroom	Precision	0.740	0.489	0.593	0.162	0.566	0.204	0.504
	Recall	0.692	0.481	0.403	0.941	0.303	0.419	0.468
Kitchen	Precision	0.758	0.506	0.465	0.207	0.487	0.252	0.522
	Recall	0.703	0.462	0.263	0.932	0.202	0.516	0.448
Living Room	Precision	0.745	0.451	0.553	0.190	0.544	0.203	0.462
	Recall	0.681	0.457	0.327	0.951	0.268	0.505	0.442
Study Room	Precision	0.809	0.494	0.574	0.224	0.580	0.264	0.506
	Recall	0.690	0.523	0.365	0.969	0.267	0.379	0.505
Supermarket	Precision	0.724	0.532	0.465	0.191	0.495	0.260	0.544
	Recall	0.652	0.579	0.316	0.970	0.247	0.229	0.565
Classroom	Precision	0.810	0.492	0.579	0.176	0.624	0.191	0.504
	Recall	0.768	0.567	0.411	0.967	0.365	0.403	0.554
Office	Precision	0.754	0.449	0.496	0.171	0.495	0.190	0.459
	Recall	0.691	0.452	0.332	0.944	0.241	0.442	0.443
Pharmacy	Precision	0.771	0.414	0.500	0.225	0.507	0.263	0.420
	Recall	0.665	0.460	0.274	0.968	0.208	0.338	0.454
Hotel	Precision	0.723	0.390	0.656	0.182	0.597	0.226	0.416
	Recall	0.602	0.374	0.374	0.925	0.356	0.394	0.363
Warehouse	Precision	0.750	0.505	0.552	0.195	0.539	0.218	0.522
	Recall	0.585	0.419	0.304	0.967	0.234	0.702	0.392
Restaurant	Precision	0.787	0.511	0.621	0.169	0.607	0.211	0.532
	Recall	0.717	0.431	0.404	0.946	0.324	0.603	0.415

H PROPRIOCEPTION-GUIDED TRANSITION CALIBRATION

As shown in Section 5.5, reducing visual input cost can substantially decrease input tokens and inference time, but also affect splitting performance. We consider two common efficiency-oriented settings: reducing the video sampling rate and reducing the image resolution. These two forms of input compression affect the model differently. Lower image resolution mainly removes spatial visual details, whereas lower sampling rates directly weaken the temporal evidence available around action transitions. Since accurate transition localization depends strongly on observing such transitions, performance can degrade noticeably when observations become temporally sparse.

To mitigate this issue, we introduce an optional proprioception-guided transition calibration step for low-FPS inference. Proprioceptive signals are recorded at a higher temporal frequency than the sampled video and provide direct physical cues around interaction changes. We use these signals to refine the temporal locations of predicted transitions.

Specifically, for each predicted subtask transition, we search within a local temporal neighborhood of 2 seconds for nearby proprioceptive events, including gripper events and motion pauses, and calibrate the transition toward the closest valid event. Each proprioceptive event can be assigned to at most one predicted transition, preventing multiple transitions from collapsing onto the same physical transition. If no valid event is found, the original transition is retained. We further apply the calibration conservatively: transitions associated with continuous in-hand manipulation are not forced toward release events, and adjustments that would violate the minimum 3-second segment duration are rejected. In this way, the semantic structure is still determined by VLM reasoning, while proprioceptive signals provide only fine-grained temporal correction.

Table 7: Effect of proprioception-guided transition calibration under different video sampling rates and image resolutions.

Factor	Setting	Calibration	Precision ↑	Recall ↑	**F1 ↑**
Video Sampling Rate	1 fps	Without Calibration	0.324	0.238	**0.274**
		With Calibration	0.451	0.349	**0.394**
	2 fps	Without Calibration	0.609	0.509	**0.554**
		With Calibration	0.614	0.511	**0.558**
	3 fps	Without Calibration	0.733	0.633	**0.679**
		With Calibration	0.730	0.630	**0.677**
Image Resolution	160×120	Without Calibration	0.628	0.549	**0.586**
		With Calibration	0.621	0.542	**0.579**
	320×240	Without Calibration	0.713	0.636	**0.672**
		With Calibration	0.702	0.626	**0.662**
	480×360	Without Calibration	0.742	0.667	**0.702**
		With Calibration	0.731	0.655	**0.691**

As shown in Table 7, proprioception-guided calibration is most effective when the video sampling rate is low and visual observations are temporally sparse. At 1 fps, calibration improves F1 from 0.274 to 0.394, together with clear gains in both precision and recall. As the sampling rate increases, however, the benefit becomes limited. At 2 fps, F1 improves only slightly from 0.554 to 0.558, while at 3 fps the difference is negligible. This suggests that when sufficiently dense visual observations are already available, the VLM can localize subtask transitions relatively accurately, leaving less room for proprioceptive refinement.

In contrast, proprioception-guided calibration is not suitable for compensating for reduced image resolution. Lower resolution primarily removes spatial appearance information, whereas proprioceptive cues provide complementary temporal evidence rather than recovering missing visual details. Accordingly, calibration does not improve performance across the tested image resolutions and can introduce small degradations. We therefore use proprioception-guided calibration only as an optional mechanism for low-FPS inference, rather than as a general correction strategy for all forms of visual input reduction.

I Performance on Annotated Transitions With and Without Proprioceptive Events

As shown in Appendix C, annotated subtask transitions and proprioceptive events (i.e., gripper events or motion pauses) are not in one-to-one correspondence: about one-third of annotated transitions have no nearby proprioceptive event, while most proprioceptive events do not correspond to a transition. We therefore separately report recall for annotated transitions with and without a nearby proprioceptive event within ± 1 second. We do not report Precision for these groups because the grouping is defined by properties of ground-truth transitions: false-positive predictions have no matched annotated transition and therefore cannot be unambiguously assigned to either group.

Table 8: Recall on annotated transitions grouped by the presence of nearby proprioceptive events. *UVD achieves high recall in both groups but has substantially lower precision.

	RoboSplit	State Only	Whole-Video VLMs	UVD*	State-Prompted VLMs	Suzuki et al.	State-to-VLMs Refine
Recall: With Proprioceptive Events	0.709	0.768	0.347	0.958	0.302	0.458	0.745
Recall: Without Proprioceptive Events	0.602	0.000	0.280	0.945	0.214	0.487	0.000

As shown in Table 8, proprioception-based methods achieve high Recall when nearby physical events are present, but State Only and State-to-VLMs Refine fail to identify transitions without such events because their candidates originate from proprioception. Vision-based methods, in contrast, retain non-zero Recall in the absence of such cues. RoboSplit maintains relatively high Recall in both groups, whereas UVD achieves high Recall at substantially lower Precision. These results highlight two complementary capabilities exposed by RoboSplitBench: exploiting physical events as temporal cues and recognizing valid subtask transitions without salient proprioceptive cues.

J False Predicted Transitions on Different Unannotated Points

To complement the transition-level Recall analysis, we further examine false predicted transitions at unannotated points. We sample trajectories at 5 fps and exclude timestamps and predicted transition points within ± 1 second of any annotation from either annotator, as well as the first and last second of each trajectory. Among the remaining points, 45.5% have a nearby proprioceptive event (a gripper event or motion pause within ± 1 second), while 54.5% do not.

For each group, we report the fraction of unannotated points within ± 1 second of a predicted transition, termed the false-transition match rate; lower is better. Since one prediction may match multiple neighboring 5-fps points, this is not a conventional pointwise false-positive rate, but remains directly comparable across methods under the same sampling grid.

Table 9: False-transition match rates at unannotated points, grouped by the presence of nearby proprioceptive events. Lower is better.

	RoboSplit	State Only	Whole-Video VLMs	UVD	State-Prompted VLMs	Suzuki et al.	State-to-VLMs Refine
With Proprioceptive Events	0.133	0.447	0.114	0.936	0.112	0.466	0.415
Without Proprioceptive Events	0.065	0.000	0.084	0.921	0.054	0.436	0.000

As shown in Table 9, State Only and State-to-VLMs Refine produce no false predicted transitions without nearby proprioceptive events by construction, but their match rates rise to 44.70% and 41.45% when such events are present, showing that physical events alone are not reliable indicators of subtask transitions. In contrast, RoboSplit maintains much lower rates with and without nearby events (13.26% and 6.50%). Together with Table 8, these results show that RoboSplitBench evaluates both the ability to detect valid transitions under different physical cues and to suppress

false transitions induced by non-semantic physical events. These analyses show that RoboSplit-Bench supports capability-level diagnosis beyond aggregate scores, revealing whether a method can appropriately exploit physical cues while remaining robust to non-semantic events.

K Hyperparameter Analysis

We evaluate the sensitivity of RoboSplit to two hyperparameters: the temporal window length and the search range for proprioception-guided window adjustment.

Table 10: Effect of temporal window length on splitting performance.

Window Length	Precision ↑	Recall ↑	**F1 ↑**
30 s	0.706	0.711	**0.708**
60 s (ours)	0.763	0.669	**0.713**
120 s	0.774	0.587	**0.668**

Table 11: Effect of proprioception-guided window adjustment range.

Search Range	Precision ↑	Recall ↑	**F1 ↑**
± 5 s	0.768	0.676	**0.719**
± 10 s (ours)	0.763	0.669	**0.713**
± 15 s	0.761	0.671	**0.713**

As shown in Tables 10 and 11, RoboSplit remains relatively stable across a reasonable range of hyperparameter settings. For the temporal window length, 30 s and 60 s achieve comparable F1 scores, while an excessively long 120-second window leads to a noticeable drop in recall, likely because longer inputs make fine-grained semantic transitions harder to identify. For proprioception-guided window adjustment, the performance varies only marginally across search ranges from ± 5 s to ± 15 s, indicating low sensitivity to this parameter.

L Evaluation with Different VLMs

We further evaluate the sensitivity of RoboSplit to the underlying VLM backbone and inference mode. We first compare different Doubao-Seed-2.1 variants under our standard experimental setting, including both the Turbo and Pro models, as well as the standard and thinking-enabled inference modes of Doubao-Seed-2.1-Pro.

Table 12: Performance of RoboSplit with different Doubao-Seed VLM backbones and thinking modes on RoboSplitBench.

VLM	Thinking Mode	Precision ↑	Recall ↑	**F1 ↑**
Doubao-Seed-2.1-Turbo	Disabled	0.706	0.640	**0.671**
Doubao-Seed-2.1-Pro	Disabled	0.763	0.669	**0.713**
Doubao-Seed-2.1-Pro	Enabled	0.760	0.676	**0.715**

As reported in Table 12, Doubao-Seed-2.1-Pro consistently outperforms the Turbo variant, while enabling thinking brings only a marginal improvement in F1. We therefore use Doubao-Seed-2.1-Pro with thinking disabled as our default backbone, as it provides a favorable trade-off between splitting performance and inference cost.

Evaluation with GPT-5.6-Sol (OpenAI, 2026). To further examine whether RoboSplit benefits from a stronger VLM, we additionally conduct a controlled experiment with GPT-5.6-Sol through the official Azure OpenAI service. Due to the limit on the number of input images per request, we cannot directly apply our standard video sampling configuration. We therefore use shorter 20-second temporal windows with a sampling rate of 1 fps and disable thinking. For a controlled comparison, we evaluate Doubao-Seed-2.1-Pro under exactly the same configuration.

As shown in Table 13, GPT-5.6-Sol substantially improves subtask splitting performance over Doubao-Seed-2.1-Pro under the same constrained input configuration, increasing F1 from 0.308 to 0.592. This result suggests that RoboSplit can benefit considerably from stronger VLM backbones. Nevertheless, the constrained setting differs from our standard configuration and is intro-

Table 13: Comparison between GPT-5.6-Sol and Doubao-Seed-2.1-Pro under the same constrained setting (20-s windows, 1 fps, and thinking disabled).

VLM	Precision ↑	Recall ↑	**F1 ↑**
Doubao-Seed-2.1-Pro	0.326	0.292	**0.308**
GPT-5.6-Sol	0.561	0.627	**0.592**

duced specifically to accommodate the input-image limit of the GPT-5.6-Sol endpoint; therefore, these results should not be directly compared with those in Table 12.

M Method Development and Evaluation Protocol

RoboSplit was developed using a small set of **separate bedroom environments** outside and different from RoboSplitBench. After fixing the method and default configuration, we applied it without modification to all 12 scenes in RoboSplitBench. Thus, all benchmark scenes were unseen during method development. The scene-wise evaluation therefore reflects the behavior of the fixed method across environments beyond those used during development.

N Algorithm Details of RoboSplit

We provide pseudocode for the three stages of RoboSplit: proprioception-guided windowing (Algorithm 1), VLM-based transition prediction (Algorithm 2), and cross-seam re-splitting (Algorithm 3).

Algorithm 1 Proprioception-Guided Windowing

Require: Trajectory duration T; proprioception $\mathcal{P}$; window length L; search range δ
Ensure: Adjusted windows $\{W_i\}_{i=1}^{N}$

Extract gripper events $\mathcal{E}_g$ and motion pauses $\mathcal{E}_m$ from $\mathcal{P}$. Construct edges $\{e_i^0\}_{i=0}^{N}$ at interval L
$e_0 \leftarrow 0, \quad e_N \leftarrow T$
for $i = 1, \ldots, N-1$ **do**
    $\mathcal{E}_i \leftarrow \{e \in \mathcal{E}_g \cup \mathcal{E}_m \mid |e - e_i^0| \le \delta\}$
    if $\mathcal{E}_i \neq \emptyset$ **then**
        $e_i \leftarrow \arg\min_{e \in \mathcal{E}_i} |e - e_i^0|$
    else
        $e_i \leftarrow e_i^0$
    end if
end for
$W_i \leftarrow [e_{i-1}, e_i)$ for $i = 1, \ldots, N$
return $\{W_i\}_{i=1}^{N}$

Algorithm 2 VLM-Based Transition Prediction

Require: Task instruction ℓ; visual observations $\mathcal{V}$; sampling rate f; adjusted windows $\{W_i\}_{i=1}^{N}$; gripper events $\mathcal{E}_g$
Ensure: Local transition sets $\{\hat{\mathcal{B}}_i\}_{i=1}^{N}$

for $i = 1, \ldots, N$ **do**
    $\mathcal{X}_i \leftarrow$ sample main-view observations $\mathcal{V}^m$ in W_i at f fps
    for all $g \in \mathcal{E}_g \cap W_i$ **do**
        Find the sampled timestamp t_g nearest to g
        Add the wrist-view observation corresponding to g at t_g to $\mathcal{X}_i$
    end for
    $\hat{\mathcal{B}}_i \leftarrow$ VLM-Split$(\ell, \mathcal{X}_i)$
end for
return $\{\hat{\mathcal{B}}_i\}_{i=1}^{N}$

Algorithm 3 Cross-Seam Re-Splitting

Require: Task instruction ℓ; visual observations $\mathcal{V}$; adjusted windows $\{W_i\}_{i=1}^{N}$; local transition sets $\{\hat{\mathcal{B}}_i\}_{i=1}^{N}$; gripper events $\mathcal{E}_g$; sampling rate f

Ensure: Final transition set $\hat{\mathcal{B}}$

$\hat{\mathcal{B}} \leftarrow \bigcup_{i=1}^{N} \hat{\mathcal{B}}_i$
for $i = 1, \ldots, N-1$ **do**
  if $\hat{\mathcal{B}}_i \neq \emptyset$ **then**
    $\hat{b}_{i,K_i} \leftarrow \max \hat{\mathcal{B}}_i$
  else
    $\hat{b}_{i,K_i} \leftarrow e_{i-1}$
  end if
  if $\hat{\mathcal{B}}_{i+1} \neq \emptyset$ **then**
    $\hat{b}_{i+1,1} \leftarrow \min \hat{\mathcal{B}}_{i+1}$
  else
    $\hat{b}_{i+1,1} \leftarrow e_{i+1}$
  end if
  $C_{W_i} \leftarrow [\hat{b}_{i,K_i}, \hat{b}_{i+1,1}]$
  $\mathcal{X}_{C_i} \leftarrow$ construct visual input over C_{W_i}
  $\hat{\mathcal{B}}_i^{\text{seam}} \leftarrow \text{VLM-SPLIT}(\ell, \mathcal{X}_{C_i})$
  $\hat{\mathcal{B}} \leftarrow \hat{\mathcal{B}} \cup \hat{\mathcal{B}}_i^{\text{seam}}$
end for
return $\text{sort}(\hat{\mathcal{B}})$