

FLORA: An Open Model Suite for Scaling Robotics Data Engines

Sicong Leng*, Yunze Man*, Zhiqi Li*, Jing Wang, Yuhe Zhang, Ryan Punamiya, Siddharth Patki, Kaiyuan Zheng, Zi-ang Cao, Zhiding Yu, Sifei Liu, Danfei Xu, Ben Levin, Jan Kautz, Yuke Zhu†, Johan Bjorck*†, Linxi “Jim” Fan†

Blog Post

research.nvidia.com/labs/gear/flora/

Open Model Suite

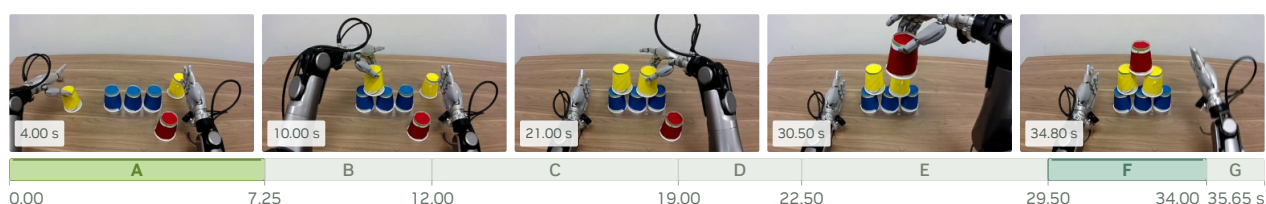
huggingface.co/collections/nvidia/flora

Abstract

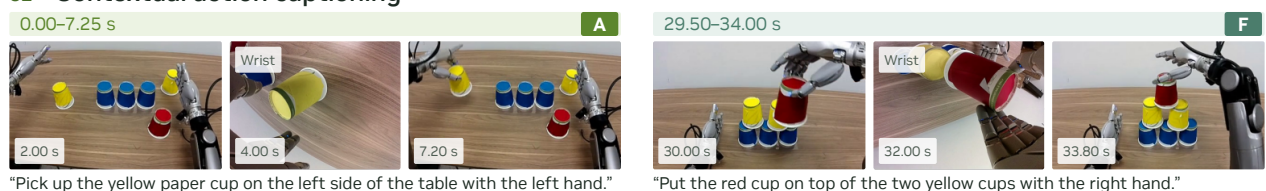
The life-cycle of robotics foundation models crucially depends on data labels. Outcome reward labels are needed for monitoring real-world deployments. Process reward labels are needed during RL post-training. Timestamped captions are needed for filtering and balancing pretraining data. We present Flora, an industrial-strength data engine used for training GR00T models, that simultaneously covers these capabilities. Flora achieves SOTA scores on the captioning benchmark WGO-Bench and reward benchmarks RoboReward and Robometer. On internal benchmarks it is competitive with GPT-6 Astra while beating other frontier models. The model is substantially cheaper than API-based alternatives, roughly $\sim 125\times$ cheaper than GPT-6 Astra. Flora is battle tested at scale, labeling 200,000 hours of data per day across 512 GB300s. By open-sourcing our data engine, we enable researchers to scale custom robotics datasets and explore emerging applications such as physical auto-research and human-to-robot transfer.

01 Temporal segmentation

Build a cup pyramid

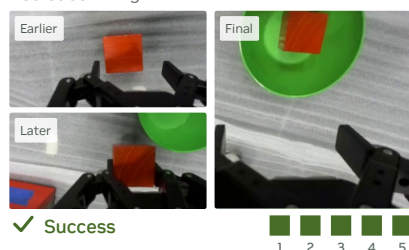


02 Contextual action captioning



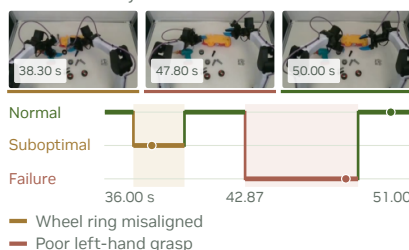
03 Outcome rewards

Red cube $\rightarrow$ green bowl



04 Process rewards

Assemble a toy car



05 Rubric rewards

Scoop beans to the fill line

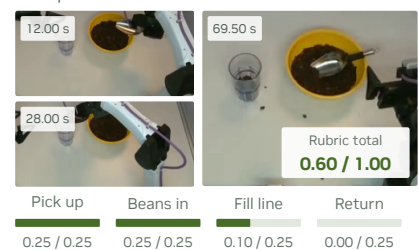


Figure 1: **Capabilities.** Flora produces timestamped captions by first segmenting videos and then captioning individual segments. Flora can also produce outcome and process rewards, and rewards based on rubrics specified at test-time.

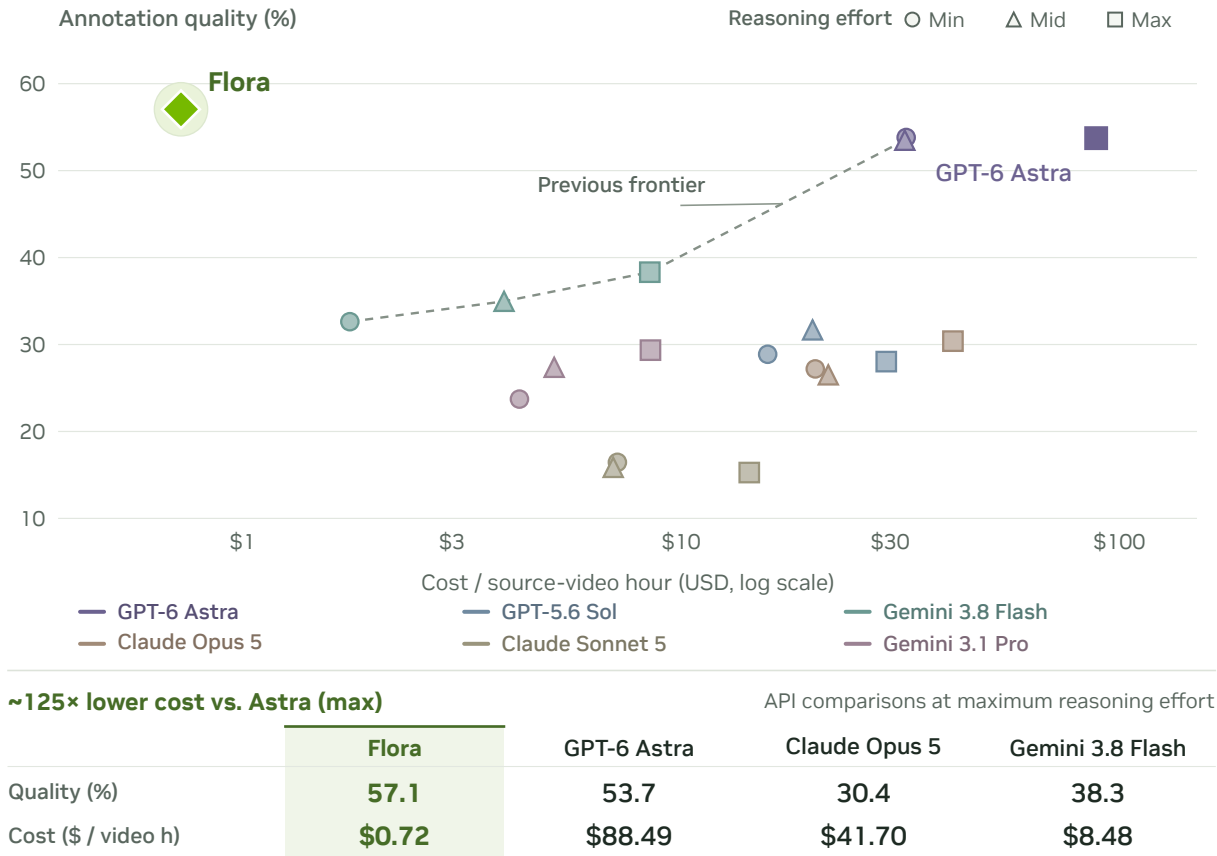


Figure 3: **Annotation quality versus cost.** Flora achieves competitive quality at substantially lower cost compared to GPT-6 Astra. Evaluation and cost assumptions appear in Appendix D.

1. Overview

Robotics foundation models have evolved from single-embodiment policies [1] to generalist models trained on large, heterogeneous corpora [2–5]. Across these large datasets, high-quality labels are imperative for performing models. *Pretraining* benefits from timestamped action instructions that enable language conditioning [6] and let practitioners rebalance the data mixture [7, 8]. *RL post-training* [9, 10] requires *process rewards* that measure progress within an episode, analogous to step-level supervision in language models [11, 12]. *Deployment* needs *outcome rewards* to monitor fleets, detect failures [13], and evaluate policies at scale [14].

Most existing systems cover only one annotation type. General-purpose frontier models can provide several types, but are expensive at the scale of modern robot datasets. We present Flora, an industrial-quality data engine used to train GROOT models. It consists of three models—a captioning model, a segmentation model, and a reward model—that together produce timestamped captions, process rewards, and outcome rewards. User-defined rubrics additionally support flexible reward modeling at test time. The engine is battle tested at scale, labeling up to 200,000 hours of data per day across 512 GB300 GPUs. By open-sourcing Flora, we enable researchers to scale custom robot datasets and explore emerging applications such as physical auto-research [15] and human-to-robot transfer [16].

2. Capabilities and Results

Table 1: **Benchmark scores.** Flora is competitive with GPT-6 Astra while beating other baselines. Values are scaled by 100. Full comparisons appear in Appendix A.

BENCHMARK / METRIC			PROPRIETARY				OPEN-WEIGHT	
		Flora	GPT-6 Astra	GPT-5.6 Sol	Gemini 3.8 Flash	Claude Opus 5	Kimi K3	Qwen3.5 397B-A17B
TIMESTAMPED CAPTIONS	WGO-Bench							
	Segmentation F1@0.75	↑ 50.3	46.0	33.7	26.4	25.5	26.1	19.5
	Gold-window accuracy	↑ 64.7	77.4	56.6	70.9	49.3	50.6	46.0
	Joint semantic F1@0.75	↑ 34.5	39.0	24.6	21.7	15.4	16.7	11.2
	DenseCaptionBench Human100 subset							
	Segmentation F1@0.5	↑ 54.6	44.2	32.6	33.5	22.7	21.1	19.6
	Caption point quality	↑ 66.5	60.3	36.4	46.7	31.0	30.3	29.1
	Joint point F1@0.5	↑ 37.4	29.7	14.6	17.7	8.2	7.9	7.8
	DenseCaptionBench Robot200 subset							
	Segmentation F1@0.5	↑ 66.9	71.9	45.8	45.8	45.9	41.2	37.0
	Caption point quality	↑ 73.7	80.2	69.6	66.0	49.7	48.0	49.5
	Joint point F1@0.5	↑ 51.0	57.9	33.6	32.2	23.6	22.2	19.8
REWARD ANNOTATIONS	RoboRewardBench / RobometerBench							
	RoboReward MAE	↓ 40.0	55.4	78.0	83.3	88.2	68.2	87.1
	Robometer OOD Kendall	↑ 72.0	82.3	64.2	63.3	63.1	63.5	48.8
	Robometer ID VOC	↑ 98.9	98.9	97.1	96.9	91.8	97.2	91.5
	Robometer OOD VOC	↑ 99.6	99.7	99.3	98.6	98.6	99.7	97.1
	DenseRewardBench							
	Macro F1	↑ 46.2	40.7	36.3	38.8	33.9	31.2	34.7
	Balanced accuracy	↑ 54.2	44.1	37.8	40.2	35.1	32.3	39.2
	Bad-action F1	↑ 47.7	44.8	26.4	31.2	22.9	17.6	31.3
	Failure recall	↑ 53.3	0.0	1.0	11.7	0.3	0.3	3.3
	RubricBench							
	Pooled L1 error	↓ 6.5	3.6	8.4	32.6	17.1	15.6	8.9

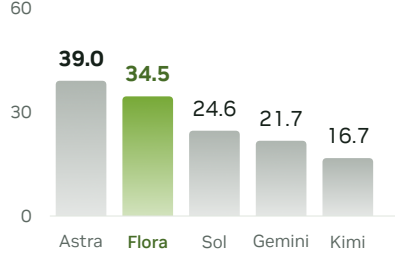
Flora Best in row (ties)

Arrows indicate better

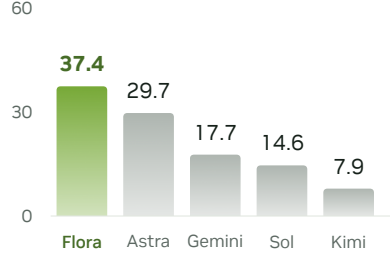
TIMESTAMPED CAPTIONS

WGO action annotation

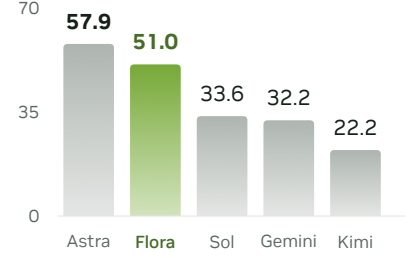
Joint semantic F1@0.75 ↑


Human action annotation

Joint point F1@0.5 ↑


Robot action annotation

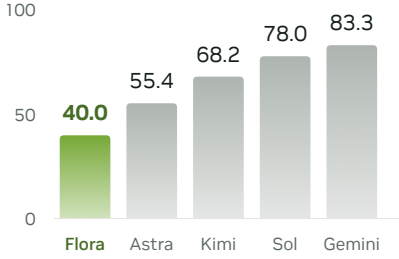
Joint point F1@0.5 ↑



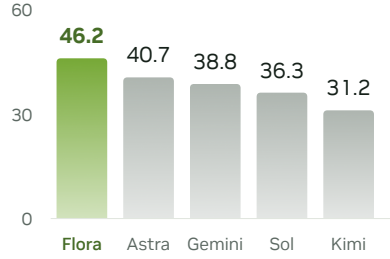
REWARD ANNOTATIONS

Outcome accuracy

RoboReward MAE ↓


Dense reward labeling

DenseRewardBench macro F1 ↑


Rubric grading

RubricBench pooled L1 ↓

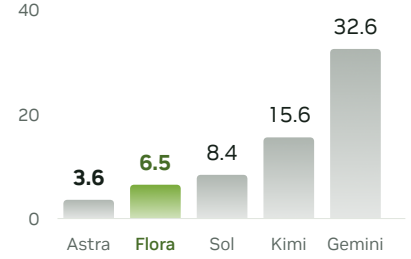


Figure 4: **One data engine, complementary capabilities.** Timestamped caption and reward labeling results across six tasks. All values are scaled by 100. Full comparisons appear in Appendix A.

2.1. Timestamped Action Captions

We evaluate *temporal segmentation*, *captioning* on gold reference intervals, and *joint annotation* on predicted intervals. Flora uses its default pipeline (Section 4.1) with the validation-selected segmentation and captioning RL checkpoints.

Common pipeline. We use a shared two-stage pipeline for off-the-shelf models. One call segments timestamped episode contact sheets, followed by one captioning call per fixed interval with up to two seconds of context on either side. Both stages use the same model and fixed stage-specific prompts. Caption-only evaluation substitutes reference intervals. Reference captions are never supplied. These comparisons evaluate complete systems with different visual inputs and inference compute (Appendix B).

WGO-Bench. Table 2 reports model comparisons on WGO-Bench [17], comprising 25 human egocentric episodes (HomER), 50 robot episodes with external views (DROID), and 25 with head-camera views (Galaxea). All systems, including Flora, receive the episode instruction in both stages. Comparisons use one-to-one temporal matching at $\text{tIoU} \geq 0.75$ and the public text-only caption rubric. Joint matches require both temporal and caption agreement. Flora attains 50.3% mean segmentation F1, 34.5% mean joint F1, and 64.7% gold-window caption accuracy. It leads segmentation and ranks second in joint F1 behind Astra’s 39.0%. Caption accuracy remains below Astra and Gemini 3.8 Flash, so the gains primarily reflect temporal localization.

Pipeline comparison. Figure 5 compares Flora with the common pipeline, Refiner [18], LeRobot [19], and Pegasus [20]. Compared with Refiner, Flora improves mean segmentation F1 by 16.9 points and joint F1 by 7.3 points, with similar gold-window caption accuracy (64.7% vs. 64.3%). Additional backbone comparisons and API substitution diagnostics appear in Appendix B.6.

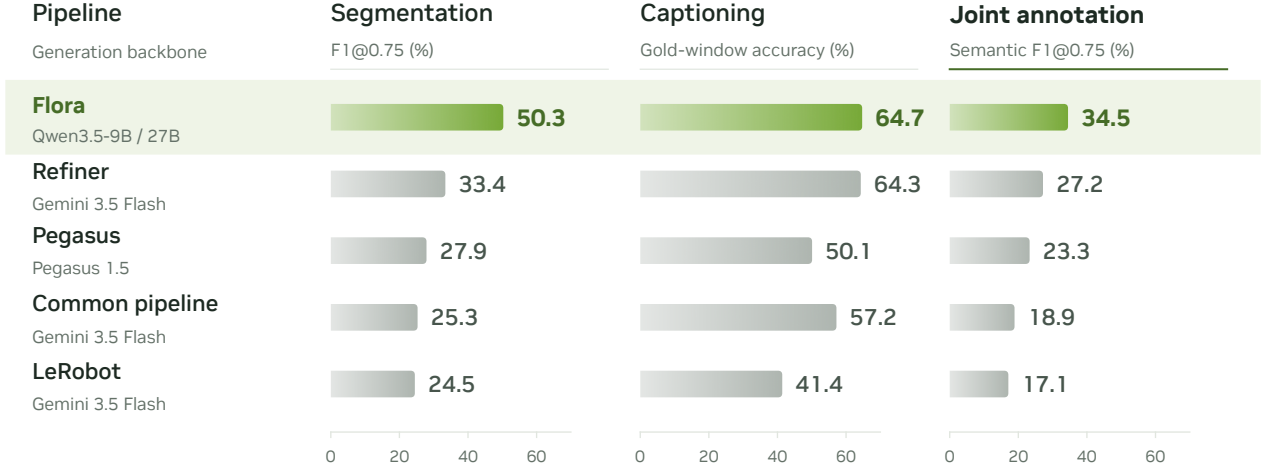


Figure 5: **Complete pipelines on WGO-Bench.** Equal-subset mean scores (%), with generation backbones shown. Full results and protocols appear in Appendix B.6.

DenseCaptionBench. WGO-Bench provides a public test of subtask annotation, but its 100 single-view episodes offer limited task and scenario coverage and do not test multi-view recordings or dexterous-hand manipulation. Its binary caption matches also do not quantify partial coverage of action details. We introduce DenseCaptionBench to assess these requirements on 300 complete episodes, with 100 each for humans, parallel-gripper robots, and dexterous-hand robots. Robot episodes retain all camera views. Selection prioritizes task, action, and manipulator-role coverage without using model performance scores. The human track spans 56 environment labels and 95 task labels. Segmentation reports F1 at $\text{tIoU} \geq 0.5$. Oracle captioning gives partial credit for intended action, object identity, manipulator coordination, and spatial relations while penalizing demonstrated factual errors. Appendix B details construction and scoring.

Table 3 reports Human100 and Robot200 results. Flora obtains 54.6% segmentation F1, 66.5% oracle caption quality, and 37.4% joint F1 on Human100, leading all three metrics. Robot Avg scores are 66.9%, 73.7%, and 51.0%, respectively. Human joint F1 exceeds Astra’s 29.7% by 7.7 points. On Robot200, Flora ranks second to Astra across the three metrics. Its 51.0% joint F1 compares with 57.9% for Astra and 33.6% for the next-best baseline.

2.2. Reward Labeling

We evaluate Flora-Rewarder on outcome, dense, and rubric-derived rewards across two public and two internal benchmarks. General-purpose baselines receive the same task instructions and answer formats as Flora. Frame budgets and decoding settings have model-specific settings. Results for specialized reward models are taken from their publications and retain their original protocols. Further details are given in Appendix C.

Outcome accuracy. RoboRewardBench [21] scores 2,831 episodes on a human-labeled 1–5 outcome scale, and we report the mean absolute error (MAE) of the decoded score. As Table 4 shows, Flora-Rewarder achieves an MAE of 40.0, compared with 55.4 for GPT-6 Astra and 66.5 for the specialized RoboReward model, with all errors multiplied by 100. This is a 27.8% reduction relative to Astra. The advantage comes from partially completed tasks.

Trajectory ranking and progress. RobometerBench [22] tests whether rewards rank 567 out-of-distribution (OOD) trajectories by quality, measured by Kendall’s τ_a , and whether predicted progress follows elapsed time on successful in-distribution (ID) and OOD trajectories, measured by value-order correlation (VOC). Flora-Rewarder ranks trajectories by the expected outcome reward and derives progress by accumulating its dense reward. In OOD ranking, it trails only GPT-6 Astra at 72.0 versus 82.3. It ties Astra at 98.9 ID VOC and reaches 99.6 OOD VOC, exceeding the reported specialized reward baselines on their evaluated metrics. VOC

measures ordering on successful episodes and does not establish failure-detection accuracy.

Dense reward. Public reward benchmarks score whole episodes or track task progress. They do not test whether a model can find the moments when a robot acts poorly or fails. DenseRewardBench contains 40 held-out bimanual episodes with human action-quality labels, scored on 4,861 time steps. Flora-Rewarder achieves the best value on all four metrics in Table 5, with a macro F1 of 46.2. Its largest advantage is failure detection: it recalls 53.3% of failure steps, whereas no baseline exceeds 11.7%. GPT-6 Astra comes closest in bad-action F1 but detects no failures. Higher failure recall has a cost: Flora-Rewarder also labels about 11% of normal steps as failures.

Rubric-derived reward. RubricBench is an in-house evaluation benchmark which contains 40 teleoperated robot episodes from 10 tasks, each graded by humans against a human-created rubric. The metric is the L1 distance between the ground-truth scores and the model predicted scores. Flora-Rewarder reaches a pooled L1 of 6.5, which is more accurate than nine of the ten general-purpose models (Table 5). Astra achieves 3.6. These are system-level comparisons, Appendix C specifies model-specific camera settings, missing-response accounting, and further details

2.3. Annotation Efficiency

Figure 3 compares Flora with six proprietary models at three reasoning settings on a fixed 50-episode subset of DenseCaptionBench, comprising 20 human, 15 parallel-gripper, and 15 dexterous-hand episodes. Quality equally weights segmentation, reference-window captioning, and joint annotation across the three embodiments. Flora achieves 57.1% quality at an estimated \$0.72 per hour of source video under an eight-GB300 deployment at \$10/GPU-hour. GPT-6 Astra at maximum reasoning effort achieves 53.7% quality at a known-usage API cost of \$88.49 per hour. The estimated cost is approximately $125\times$ lower, with mean quality 3.3 percentage points above Astra, computed from unrounded means. Cost accounting, reference versions, confidence intervals, and quota-based throughput estimates appear in Appendix D.

3. Qualitative Examples

Figure 6 compares Flora to GPT-6 Astra on temporal coverage, intended actions, and reward decisions. Additional qualitative evidence appears in Appendix E.

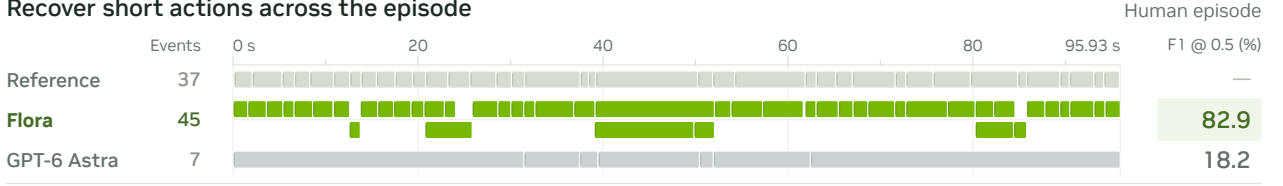
Temporal coverage and local operations. In the selected episode, Flora matches 34 of 37 reference intervals, compared with four for Astra, yielding F1@0.5 scores of 82.9 and 18.2. The caption examples show the importance of preserving the intended operation: Flora identifies wire insertion and a right-to-left handover, where Astra describes a flower piece and pouch folding, respectively. The first caption also distinguishes the hand holding the connector from the hand inserting the wire.

Failure severity and partial outcomes. Both models describe the falling component, but only Flora-Rewarder assigns the failure label. In the bottle task, Flora-Rewarder distinguishes partial progress from complete failure: toppling the bottle without knocking it off the table receives the reference score of 3/5, whereas Astra assigns 1/5.

4. Methods

We now go into technical details of how Flora is created. Flora aligns intended-action descriptions with execution quality. Captioning localizes each action and describes it using visual context. Reward labeling evaluates individual steps, episode completion, or a user-defined rubric.

Recover short actions across the episode



Wire insertion

Left: connector · Right: wire insertion



93.50–95.23 s

Flora

“hold connector with left hand, hold wires and insert wire into connector with right hand”

GPT-6 Astra

“Hold the small white flower piece steady and work the tip of the yellow handheld tool against its base, turning the piece slightly between your fingers.”

Hand-to-hand transfer

Right to left hand · Tea leaves



102.43–104.53 s

Flora

“hold tea leaves with right hand, pass tea leaves to left hand”

GPT-6 Astra

“Pinch and fold the opening of the small white pouch closed with both hands above the tray of loose leaves.”

Detect an assembly failure

Execution quality during toy-airplane assembly



Red component falls off

Flora Failure GPT-6 Astra Suboptimal

Credit partial progress

Goal: knock the bottle off the table



Bottle topples but remains on the table

Flora 3/5 GPT-6 Astra 1/5 Reference 3/5

Figure 6: **From intended actions to execution quality.** Selected examples show temporal coverage (top), human action captions (middle), and failure recognition and partial progress on DenseRewardBench and RoboRewardBench (bottom).

4.1. Timestamped Captions

In robotics we often need dense, timestamped captions. Given a human or robot video X of duration T , we output tuples (s, e, c) . The string c describes the intended action between time s and time e . An action may span several movements and need not succeed. One model provides temporal segments (s_i, e_i) while another provides the captions c_i . Flora thus produces

$$\mathcal{A}(X) = \{(s_i, e_i, c_i)\}_{i=1}^N, \quad I_i = [s_i, e_i], \quad 0 \leq s_i < e_i \leq T. \quad (1)$$

The pipeline applies a segmentation model to overlapping windows, stitches its predictions into episode-level intervals, and applies a captioning model to each interval with local context (Figure 7).

Temporal segmentation and stitching. We divide an episode into overlapping windows, each with a central output region and surrounding context. The segmentation model predicts start and end times relative to the window, with binary flags indicating continuation past its left and right edges. A deterministic filter retains only fragments intersecting the central region. An action may appear partially in several windows. We convert local timestamps to episode time and join compatible fragments using temporal overlap and predicted continuation flags, while retaining separate intervals for concurrent actions. Boundary refinement reuses the segmenter on shifted windows. Appendix F details the stitching and refinement rules.

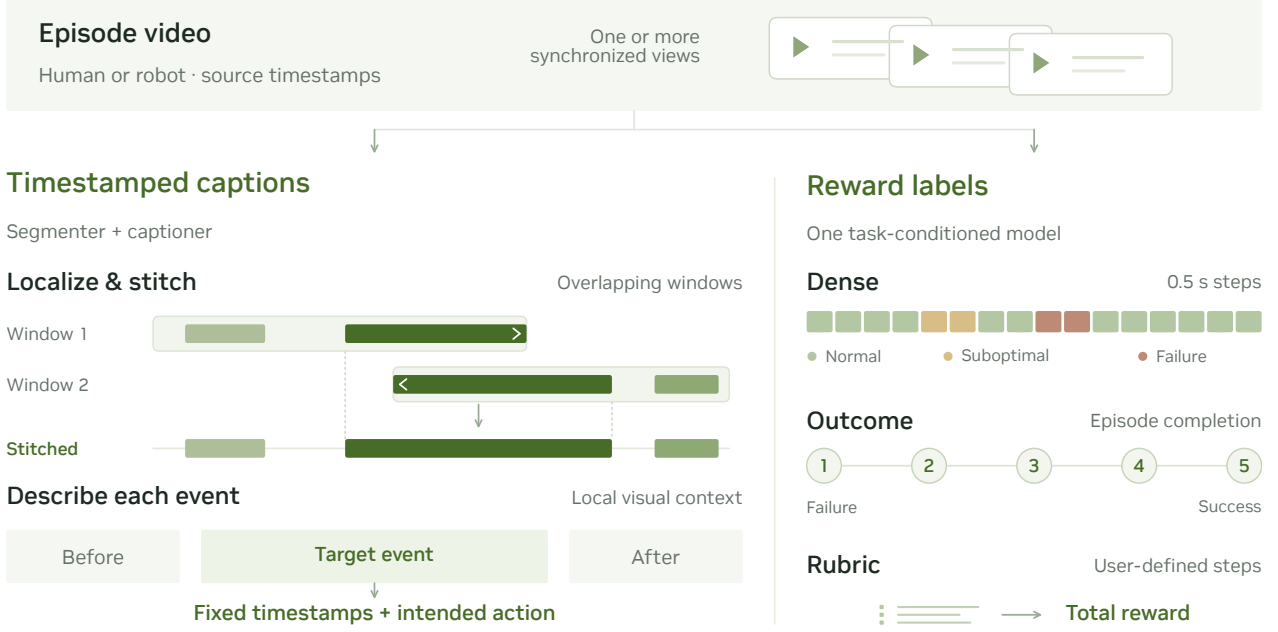


Figure 7: **From video to reusable supervision.** Overlapping windows are stitched into event intervals for contextual captioning. One reward model provides dense, outcome, or rubric labels. Timelines are schematic.

Contextual captioning. For each reconstructed interval I_i , the pipeline prepares target frames X_i and context frames X_i^- and X_i^+ from the nearest non-overlapping events before and after it. The captioning model g_ϕ independently generates an instruction c_i :

$$c_i = g_\phi(X_i^-, X_i, X_i^+, q_{\text{cap}}). \quad (2)$$

The pipeline explicitly labels the before, target, and after blocks. Neighboring frames help resolve object identity and local intent, while the caption remains restricted to I_i . Unavailable context is omitted. For multi-camera recordings, the pipeline tiles synchronized views into one image at each sampled time. The prompt identifies camera roles and requests one description of each action across views. Single-camera recordings use the same temporal blocks. The prompt requests a self-contained imperative describing the intended action, objects, manipulator roles, and relevant spatial relations. Failed attempts retain their intended goal without implying success. Finally, the pipeline pairs each generated caption c_i with its fixed interval I_i to form $\mathcal{A}(X)$.

4.2. Reward Labels

Robot foundation models need rewards at several granularities. *Outcome* rewards judge whether an episode accomplished its task, for deployment monitoring, data filtering, and policy evaluation. *Dense* rewards localize where an episode goes wrong, for per-step credit assignment in RL post-training and fine-grained data curation. For flexible reward modeling at test time, we also support *rubric-derived* rewards. Flora-Rewarder is a single model r_ψ that provides all three, selected by its prompt. Outcome and dense prompts tile synchronized camera views into one mosaic per sampled time and condition on the global task instruction. Rubric grading receives separate synchronized camera videos.

Outcome reward. Reinforcement learning for robot policies often learns only from whether each episode succeeded. Simulators report this outcome automatically [9], but on real robots a person labels each episode [10]. An outcome reward model can provide these labels instead. The outcome reward scores task completion from complete failure ($s = 1$) to complete success ($s = 5$), optionally after a short rationale. For a continuous reward, we read the distribution over the five score tokens and rescale its expectation to $[0, 1]$ (Eq. (19)).

Dense execution quality. Feedback at every step helps where a single outcome is too coarse. Each 0.5-second step is labeled failure (1), suboptimal (2), or normal (3). A failure causes a major task failure, an unsafe or destructive event, or a loss of task viability. A suboptimal action is hesitant, inefficient, imprecise, or locally incorrect, but recoverable. Labels are retrospective: later frames may reveal a mistake, but a later recovery does not improve an earlier step. The model also returns a rationale for each contiguous non-normal span. Windows span 16 seconds and advance by 10 seconds. We drop the last four seconds of each non-final window so that retained labels have later visual evidence, then stitch labels on the episode clock. These retrospective annotations are intended for offline labeling.

Rubric-derived reward. A scoring rubric lets users flexibly define rewards at test time. A rubric lists task steps, each with a set of possible scores whose maxima sum to one. Given the episode, instruction, and rubric, the model reasons in thinking mode and then awards one score per step. The total credit is

$$R_\rho(X) = \sum_{j=1}^n \hat{y}_j, \quad \hat{y}_j \in V_j, \quad \sum_{j=1}^n \max V_j = 1, \quad (3)$$

where V_j contains the allowed grades for step j . Table 11 gives an example. Appendix G specifies the inputs, output formats, and score extraction.

4.3. Training

Training infrastructure. To build the Flora data engine, we fine-tune open-source VLMs from the Qwen3.5 family [23]. SFT uses NeMo AutoModel [24], and RL uses a custom implementation in PyTorch. Optimization and annotation-RL details appear in Appendices F and G.

Data strategy. We train on a combination of open-source datasets, licensed proprietary datasets, and in-house datasets that we collect and label ourselves. Human annotators provide dense, timestamped captions and action-quality labels, guided by detailed labeling guidelines. Reward supervision also includes episode outcomes and rubric grades.

Supervised fine-tuning. We initialize the segmentation model f_θ from Qwen3.5-9B, and the captioning model g_ϕ and reward model r_ψ from Qwen3.5-72B. All three models minimize output-token cross-entropy. The segmenter uses LoRA with a frozen visual encoder. The captioner updates the language model and visual merger. Segmentation targets encode window-local intervals and continuation flags. Instruction targets use annotated event boundaries and neighboring context.

For the reward model, we fine-tune the language model, visual merger, and visual encoder, using a lower learning rate for the visual encoder. Rows from all three reward families are shuffled together, without reweighting or oversampling, and trained jointly for one epoch. Rubric examples are answered in thinking mode, while outcome and dense examples are answered directly. The released reward model uses SFT only.

Reinforcement learning for segmentation. We further train the SFT segmenter with GRPO [25]. A verifiable reward is $r_{\text{seg}} = 0.45F_{0.5} + 0.15F_{0.75} + 0.25P_{0.5} + 0.15B$, where F_τ and P_τ are temporal F1 and precision at tIoU threshold τ , and B measures endpoint and continuation-flag accuracy. We update only LoRA parameters, with group-mean advantages and KL regularization to the fixed SFT policy. Scoring is restricted to each emission core.

Reinforcement learning for captioning. We further optimize the SFT captioner with GRPO using a reference-grounded point-quality reward. Source-video evidence and reference requirements define $r_{\text{cap}} = 2M/(R + M + F)$, where R is supported reference mass, M is covered mass, and F penalizes demonstrated factual errors. The reward gives partial credit for supported content. We use group-mean advantages and KL regularization to the fixed SFT policy. Validation selects the two annotation checkpoints independently before benchmark evaluation.

5. Conclusion

We present Flora, an industrial-quality data engine for timestamped captions, process rewards, and outcome rewards for robotics datasets. Flora has been used to train GR00T models. The model is competitive with GPT-6 Astra on captioning benchmarks while improving on reported specialized reward models on RoboReward and Robometer. Its estimated labeling cost offers a practical route to annotating large pretraining corpora. By open-sourcing Flora, we hope to enable the community to scale custom robot datasets and to explore novel research directions such as physical auto-research and human-to-robot transfer.

Acknowledgements. We are grateful for discussions and help from Cristaldo Campos, Omar Laymoun, Anita Agasaveeran, Marina Davila, Jenna Diamond, and the broader NVIDIA community.

References

- [1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 2
- [2] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023. 2
- [3] Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- [4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- [5] Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. 2
- [6] Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, et al. Robotic skill acquisition via instruction augmentation with vision-language models. *arXiv preprint arXiv:2211.11736*, 2022. 2, 37
- [7] Joey Hejna, Chethan Anand Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. ReMix: Optimizing data mixtures for large scale imitation learning. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pages 145–164, 2025. URL <https://proceedings.mlr.press/v270/hejna25a.html>. CoRL 2024, proceedings published in 2025. 2
- [8] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, et al. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2
- [9] Haozhan Li et al. SimpleVLA-RL: Scaling VLA training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025. 2, 8
- [10] Physical Intelligence. $\pi_{0.6}^*$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025. 2, 8
- [11] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, et al. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. 2
- [12] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, et al. Let’s verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. 2
- [13] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, et al. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024. 2
- [14] Pranav Atreya, Karl Pertsch, et al. RoboArena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025. 2
- [15] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. 2
- [16] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, et al. EgoMimic: Scaling imitation learning via egocentric video. *arXiv preprint arXiv:2410.24221*, 2024. 2
- [17] Macrodata Labs. Segmenting robot video into actionable subtasks. <https://macrodata.co/blog/annotating-robot-video-subtasks>, 2026. June 25, 2026; accessed September 15, 2026. 4, 18
- [18] Macrodata Labs. Refiner. <https://github.com/macrodata-labs/refiner>, 2026. Open-source data processing framework. Accessed October 14, 2026. 4, 18, 19, 37

- [19] Steven Palma, Pepijn Kooijmans, Caroline Pascal, Khalil Meftah, Maxime Ellerbach, Martino Russi, Nikodem Bartnik, Nicolas Rabault, and Thomas Wolf. LeRobot v0.6.0: Imagine, evaluate, improve. <https://huggingface.co/blog/lerobot-release-v060>, 2026. 4, 18, 19, 37
- [20] Twelve Labs. Pegasus 1.5. Model documentation. <https://docs.twelvelabs.io/v1.3/docs/concepts/models/pegasus/pegasus-1-5>, 2026. Accessed October 14, 2026. 4, 18, 19
- [21] Tony Lee et al. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026. 5, 15, 37
- [22] Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026. 5, 15, 37
- [23] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL <https://qwen.ai/blog?id=qwen3.5>. 9, 15, 29
- [24] NVIDIA. NeMo AutoModel: DTensor-native SPMD library for scalable and efficient training. <https://github.com/NVIDIA-NeMo/Autodel>, 2025–2026. GitHub repository. 9
- [25] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL <https://arxiv.org/abs/2402.03300>. 9, 30
- [26] OpenAI. GPT-6 Astra system card. <https://deploymentsafety.openai.com/gpt-6-astra>, September 2026. OpenAI Deployment Safety Hub. 15
- [27] OpenAI. GPT-5.6 system card. <https://deploymentsafety.openai.com/gpt-5-6>, July 2026. Covers GPT-5.6 Sol, Terra, and Luna. OpenAI Deployment Safety Hub. 15
- [28] Anthropic. System card: Claude Opus 5. <https://anthropic.com/claude-opus-5-system-card>, July 2026. 15
- [29] Anthropic. System card: Claude Sonnet 5. <https://anthropic.com/claude-sonnet-5-system-card>, June 2026. 15
- [30] Google DeepMind. Gemini 3.1 Pro model card. <https://deepmind.google/models/model-cards/gemini-3-1-pro/>, February 2026. 15
- [31] Google DeepMind. Gemini 3.8 Flash model card. <https://deepmind.google/models/model-cards/gemini-3-8-flash/>, September 2026. 15, 18
- [32] Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL <https://arxiv.org/abs/2607.24653>. 15
- [33] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-Dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025. 15, 37
- [34] Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Jiahao Tang, Jianfei Yang, Donglin Wang, Peixi Peng, Mingxiu Chen, Deli Zhao, and Xin Li. RynnValue: Scaling robotic value foundation models with temporal distance. *arXiv preprint arXiv:2608.09853*, 2026. 15
- [35] Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. LLM-as-a-Verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026. 15
- [36] Google DeepMind. Gemini 3.5 Flash: Model evaluation – approach, methodology and results. https://storage.googleapis.com/deepmind-media/gemini/gemini_3-5_flash_model_evaluation.pdf, 2026. 19
- [37] Ben Levin. Expanding our data engine for physical AI. <https://scale.com/blog/physical-ai>, 2025. 37
- [38] Macrodata Labs. WGO-Bench: What’s going on benchmark. <https://huggingface.co/datasets/macrodata/WGO-Bench>, 2026. 37
- [39] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in PyTorch. <https://github.com/huggingface/lerobot>, 2024. 37

- [40] Raehyuk Jung, Hyojun Go, Jaehyuk Yi, Jiho Jang, et al. Pegasus-v1 technical report. *arXiv preprint arXiv:2404.14687*, 2024. URL <https://arxiv.org/abs/2404.14687>. 37
- [41] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024. 37
- [42] Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. 37
- [43] Clement Farabet and Olivier Lacombe. Gemma 4: Byte for byte, the most capable open models. <https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/>, 2026. 37
- [44] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL <https://arxiv.org/abs/2511.21631>. 37
- [45] Weiyun Wang et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. 37
- [46] GLM-V Team. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025. 37
- [47] Kimi Team. Kimi-VL technical report. *arXiv preprint arXiv:2504.07491*, 2025. 37
- [48] Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. 37
- [49] Long Xing, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jianze Liang, Qidong Huang, Jiaqi Wang, Feng Wu, and Dahua Lin. CapRL: Stimulating dense image caption capabilities via reinforcement learning. *arXiv preprint arXiv:2509.22647*, 2025. 37
- [50] Penghui Yang, Long Xing, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Yibin Wang, Yujie Zhou, Jiazi Bu, Jianze Liang, Qidong Huang, Jiaqi Wang, Feng Wu, and Dahua Lin. CapRL++: Unified reinforcement learning with verifiable rewards for dense image and video captioning. *arXiv preprint arXiv:2606.09393*, 2026. 37
- [51] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023. 37
- [52] Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning (ICML)*, 2023. URL <https://proceedings.mlr.press/v202/ma23b.html>. 37
- [53] Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024. 37
- [54] Sumedh A. Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Biyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 37
- [55] Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, et al. Vision language models are in-context value learners. *arXiv preprint arXiv:2411.04549*, 2024. 37

A. Complete Benchmark Results

Figure 4 and Table 1 summarize selected baselines. The tables below retain every evaluated model and metric breakdown. All values are scaled by 100. Green identifies Flora, and shaded cells mark the best displayed result, including ties. Evaluation protocols appear in Appendices B and C.

A.1. Timestamped captions: WGO-Bench

Table 2: **WGO-Bench results.** Subset scores and equal-weight averages (% , $\uparrow$). Off-the-shelf models use the **common pipeline**, and Flora uses its default pipeline (Section 4.1).

Model / system	Segmentation F1@0.75				Captioning Gold-window accuracy				Joint annotation Semantic F1@0.75			
	HomER	DROID	Galaxea	Avg	HomER	DROID	Galaxea	Avg	HomER	DROID	Galaxea	Avg
Proprietary models												
GPT-6 Astra	25.2	65.5	47.2	46.0	64.0	82.0	86.2	77.4	19.3	54.5	43.1	39.0
GPT-5.6 Sol	23.5	44.0	33.5	33.7	48.7	62.7	58.5	56.6	16.9	33.6	23.4	24.6
GPT-5.6 Luna	20.6	32.3	37.8	30.2	46.8	55.3	57.7	53.3	13.3	22.3	24.0	19.9
Gemini 3.8 Flash	23.9	30.8	24.5	26.4	63.0	70.0	79.7	70.9	20.6	23.7	20.9	21.7
Gemini 3.1 Pro	12.0	27.0	38.1	25.7	57.7	58.7	56.9	57.8	7.5	14.9	29.4	17.3
Claude Opus 5	10.6	38.8	27.2	25.5	45.1	50.0	52.8	49.3	7.4	20.1	18.7	15.4
Claude Sonnet 5	3.2	29.5	29.6	20.8	36.8	31.3	37.4	35.2	2.6	10.9	19.4	11.0
Open-weight models												
Qwen3.5-27B	12.7	28.8	25.7	22.4	42.3	39.3	49.6	43.7	5.4	15.8	20.0	13.7
Qwen3.5-397B-A17B	10.5	28.7	19.4	19.5	47.0	44.7	46.3	46.0	8.2	14.0	11.5	11.2
Kimi K3	18.9	28.2	31.1	26.1	44.0	57.3	50.4	50.6	12.7	17.9	19.6	16.7
Flora (ours)	23.9	56.1	70.8	50.3	64.0	55.3	74.8	64.7	17.9	32.5	53.1	34.5

A.2. Timestamped captions: DenseCaptionBench

Table 3: **DenseCaptionBench results.** Scores on 100 human, 100 parallel-gripper (Grip.), and 100 dexterous-hand (Dex.) episodes under the shared point-quality protocol (% , $\uparrow$). Off-the-shelf models use the **common pipeline**, and Flora uses its production pipeline (Section 4.1).

Model / system	Segmentation F1@0.5				Oracle captioning Point quality				Joint annotation Point F1@0.5			
	Human	Robot		Avg	Human	Robot		Avg	Human	Robot		Avg
		Grip.	Dex.			Grip.	Dex.			Grip.	Dex.	
Proprietary models												
GPT-6 Astra	44.2	70.7	73.1	71.9	60.3	79.6	80.8	80.2	29.7	57.7	58.1	57.9
GPT-5.6 Sol	32.6	32.4	59.2	45.8	36.4	68.8	70.3	69.6	14.6	22.8	44.3	33.6
GPT-5.6 Luna	21.4	23.8	51.9	37.9	28.5	54.2	52.1	53.2	8.1	13.7	31.5	22.6
Gemini 3.8 Flash	33.5	33.7	57.9	45.8	46.7	64.4	67.5	66.0	17.7	22.6	41.7	32.2
Gemini 3.1 Pro	20.7	29.0	43.3	36.2	39.1	44.7	48.0	46.4	9.3	11.5	21.1	16.3
Claude Opus 5	22.7	42.5	49.2	45.9	31.0	50.0	49.3	49.7	8.2	23.2	24.0	23.6
Claude Sonnet 5	13.8	33.6	42.1	37.9	23.0	31.9	40.4	36.2	4.4	13.1	17.8	15.5
Open-weight models												
Qwen3.5-27B	13.3	21.5	54.8	38.2	29.3	44.2	47.4	45.8	5.3	11.0	27.7	19.4
Qwen3.5-397B-A17B	19.6	26.3	47.6	37.0	29.1	46.2	52.7	49.5	7.8	13.9	25.7	19.8
Kimi K3	21.1	28.7	53.6	41.2	30.3	45.6	50.3	48.0	7.9	15.0	29.4	22.2
Flora (ours)	54.6	71.0	62.7	66.9	66.5	74.3	73.1	73.7	37.4	52.6	49.4	51.0

A.3. Public reward benchmarks

Table 4: **Public reward benchmarks.** Flora-Rewarder has the lowest reported MAE and exceeds the specialized reward baselines on their reported metrics. Published specialized-model scores retain their original protocols. All metrics are multiplied by 100.

Model	RoboRewardBench	RobometerBench		
	MAE ↓	OOD Kendall τ_a ↑	ID VOC ↑	OOD VOC ↑
Proprietary models				
GPT-6 Astra	55.4	82.3	98.9	99.7
GPT-5.6 Sol	78.0	64.2	97.1	99.3
GPT-5.6 Luna	77.9	59.3	95.7	98.7
Gemini 3.8 Flash	83.3	63.3	96.9	98.6
Gemini 3.1 Pro	99.6	57.2	85.3	96.8
Claude Opus 5	88.2	63.1	91.8	98.6
Claude Sonnet 5	96.6	48.3	86.1	89.7
Open-weight general models				
Qwen3.5-27B	84.8	53.4	91.4	94.1
Qwen3.5-397B-A17B	87.1	48.8	91.5	97.1
Kimi K3	68.2	63.5	97.2	99.7
Specialized reward models				
RoboDopamine	—	45.0	79.0	80.0
RynnValue	—	67.5	—	—
LLM-as-a-Verifier	72.0	—	—	—
RoboReward	66.5	47.0	82.0	88.0
Robometer	75.0	65.5	92.0	94.0
Ours				
Flora-Rewarder	40.0	72.0	98.9	99.6

A.4. Dense and rubric rewards

Table 5: **Internal reward benchmarks.** Flora-Rewarder achieves the best value on all four DenseRewardBench metrics and grades rubrics more accurately than nine of the ten general-purpose models. All metrics are multiplied by 100

Model	DenseRewardBench				RubricBench
	Macro F1 ↑	Balanced acc. ↑	Bad-action F1 ↑	Failure recall ↑	Pooled L1 ↓
Proprietary models					
GPT-6 Astra	40.7	44.1	44.8	0.0	3.6
GPT-5.6 Sol	36.3	37.8	26.4	1.0	8.4
GPT-5.6 Luna	34.7	34.9	17.6	2.3	22.6
Gemini 3.8 Flash	38.8	40.2	31.2	11.7	32.6
Gemini 3.1 Pro	32.8	33.2	16.0	3.0	15.1
Claude Opus 5	33.9	35.1	22.9	0.3	17.1
Claude Sonnet 5	31.5	32.4	13.3	0.3	22.4
Open-weight general models					
Qwen3.5-27B	30.8	32.4	7.7	5.7	15.4
Qwen3.5-397B-A17B	34.7	39.2	31.3	3.3	8.9
Kimi K3	31.2	32.3	17.6	0.3	15.6
Ours					
Flora-Rewarder	46.2	54.2	47.7	53.3	6.5

Baselines. We consider both VLM models and full labeling pipelines as baselines. For models we consider publicly available proprietary models: GPT-6 Astra [26], GPT-5.6 Sol and Luna [27], Claude Opus 5 and Sonnet 5 [28, 29], Gemini 3.1 Pro [30] and Gemini 3.8 Flash [31]. For open-weights models we consider Kimi K3 [32] and Qwen3.5-27B and Qwen3.5-397B-A17B [23]. We also compare with a few specialized reward models: RoboDopamine [33], RynnValue [34], LLM-as-a-Verifier [35], RoboReward [21], and Robometer [22]. Pipeline backbones and implementations appear in Appendix B.6.

B. Timestamped Captioning: Evaluation Details

We evaluate localization, caption quality on reference intervals, and joint annotation on complete episodes. The following sections specify benchmark construction, visual inputs, scoring, and supplementary pipeline comparisons. All results use the model configurations reported in the main text.

B.1. DenseCaptionBench construction

Table 6 summarizes the fixed 300-episode DenseCaptionBench. Every episode supports all three evaluation tasks. The parallel-gripper track is labeled YAM in Table 3, and Dex denotes the dexterous-hand track.

Table 6: **DenseCaptionBench composition.** Reference intervals may include idle gaps and concurrent intended actions.

Embodiment	Visual evidence	Episodes	References
Human	Egocentric camera	100	3,220
Parallel grippers	Main and two wrist cameras	100	1,266
Dexterous hands	Main and two wrist cameras	100	391
Total		300	4,877

Coverage-based selection. We select episodes using available task, action, environment, and manipulator-coordination metadata. For a feature group g , let $\mathcal{T}_g$ be its labels, N_t the number of eligible pool episodes with label t , and $n_t(S)$ the number in a selected set S . The diminishing-return coverage objective is

$$\Phi(S) = \sum_{g \in \mathcal{G}} \frac{1}{|\mathcal{T}_g|} \sum_{t \in \mathcal{T}_g} \sqrt{\frac{n_t(S)}{N_t}}. \quad (4)$$

Greedy selection adds the admissible episode with the largest marginal gain, using fixed-seed tie-breaking and duplicate filtering. Counts measure episode presence, not repeated event frequency. Robot selection also limits repeated tasks and collection sessions and maintains source-family quotas. Missing metadata contributes no coverage gain. Initial selection uses existing annotations and metadata without model outputs or scores, and imposes no single-/multiple-event, duration, or event-density quota.

The human track spans 56 environment labels and 95 task labels. The two robot tracks span 99 and 75 source-qualified task groups, respectively. These are metadata coverage measures, not independently verified counts of distinct skills. Unavailable or excluded sources are replaced using the same metadata-based criteria before comparison on the final roster. The human cohort is therefore conditioned on source and service availability.

Reference construction. Intended actions originate from human annotations and are refined through source review and adjudication. Reference specifications record action requirements, object identities, coordination, phase dependencies, and observation limits before candidate-caption access in the scoring stage. Robot references include 1,364 units with established intended actions and 293 observed-operation proxies. These types remain separately identified. Sampled review uses temporal coverage and targeted boundary inspection, with uncertainty retained when the evidence is insufficient. It does not establish continuous visual coverage or exact transition times. Reference eligibility is shared across compared models.

B.2. Inputs and inference settings

Common pipeline. Off-the-shelf models use the two-stage interface in Section 2.1. One call segments timestamped episode contact sheets. Each interval is then captioned independently from labeled before, target, and after blocks, using the same model in both stages. Oracle captioning substitutes reference intervals for predicted ones. Generation never receives reference captions, neighboring labels, or segmentation-stage draft captions. Captioning cannot alter interval boundaries.

For WGO and human episodes, segmentation samples at 2 Hz, capped at 1,024 frames. Caption targets sample at 4 Hz, capped at 256 frames. Context extends at most 2 seconds per side at 2 Hz, with at most five frames per side. Capped sampling preserves the full interval span. Frames use timestamped 320×240 letterboxed tiles in 4×4 contact sheets, encoded at JPEG quality 90.

Robot episodes retain their three camera streams. Segmentation samples uniform temporal bin centers at nominal 2 Hz, capped at 256 mosaics. Caption targets use duration-adaptive sampling at nominal 8 Hz. Nominal budgets are 8–40 target frames and 4–12 frames per available context block, up to 64 frames total. Source availability can reduce these counts. Context remains limited to 2 seconds per side. Main and wrist views form 768×648 mosaics for the parallel-gripper track and 640×720 mosaics for the dexterous-hand track. A fixed image-scale ladder fits lossless PNG inputs within the request budget without removing selected timestamps. Temporal packing may differ across request containers. Source-limited inputs retain explicit padding or unavailable-input status.

Native Flora pipeline. The segmenter samples 64 frames at 4 Hz in 16-second windows, with a 12-second core and 2-second guards. Long-horizon robot sources use 32-second windows, a 24-second core, and 4-second guards. Local output coordinates use 20 Hz ticks with continuation flags. The production parser, adaptive window processing, and stitching are fixed before evaluation. Inference uses BF16 computation while retaining Gated DeltaNet parameters and temporal LoRA tensors in FP32. Greedy decoding allows 1,024 temporal tokens and 512 caption tokens. The captioner uses the duration-adaptive 8 Hz sampling rule above through its trained video processor. WGO uses adjacent-event context, while DenseCaptionBench evaluation clips real context to 2 seconds per side. Generation on DenseCaptionBench receives neither task labels nor reference text. All WGO systems receive the dataset episode instruction in both stages. These are complete-system comparisons. Native processors, prompts, window schedules, and inference compute differ from the common pipeline.

Table 7: **Baseline decoding configurations.** Output limits are in thousands of tokens ($k = 1,024$). Default denotes an omitted request field, not a verified reasoning level.

Model	Reasoning and sampling	Output limit	
		DenseCaptionBench	WGO
GPT-6 Astra	xhigh, default sampling	32k	32k
GPT-5.6 Sol / Luna	xhigh, default sampling	32k	32k
Gemini 3.8 Flash	Default reasoning, temperature 0	32k	32k
Gemini 3.1 Pro	Default reasoning, temperature 0	16k	32k
Claude Opus 5 / Sonnet 5	Provider defaults	32k	32k
Kimi K3	max, default sampling	32k	32k
Qwen3.5-27B	Greedy on DenseCaptionBench, sampling on WGO	16k	32k
Qwen3.5-397B-A17B	Temperature 0 on DenseCaptionBench, sampling on WGO	16k	32k

Qwen WGO runs enable thinking with temperature 1, top- $p = 0.95$, top- $k = 20$, and presence penalty 1.5. On DenseCaptionBench, the 27B runs disable thinking and use greedy decoding. The 27B baseline uses official BF16 weights. For hosted models, weight revisions, numerical precision, and effective backend settings are not independently exposed. Requested settings are fixed within each benchmark and are not compute matched.

B.3. DenseCaptionBench scoring and aggregation

Temporal segmentation. Let $\hat{\mathcal{I}}$ and $\mathcal{I}^*$ denote predicted and reference intervals. An edge is eligible at $\text{tIoU} \geq 0.5$. Caption-blind maximum-cardinality one-to-one matching gives TP and

$$F_{\text{seg}} = \frac{2\text{TP}}{|\hat{\mathcal{I}}| + |\mathcal{I}^*|}. \quad (5)$$

We also retain $\text{tIoU} 0.75$ as a sensitivity setting. Boundary arithmetic uses integer microseconds. There is no reference-edge snapping. Only the final outer endpoint is intersected with the available media clock.

Caption quality. Each reference contains requirements in four groups: action, identity, coordination, and detail, weighted 0.50, 0.25, 0.20, and 0.05. Each supported requirement a has a fixed mass m_a and a binary coverage indicator z_a . Coverage requires a supporting caption span and satisfaction of the reference’s logical dependencies. A phase-specific requirement additionally requires its associated action-phase predicates, so a state from another phase cannot receive credit. With supported requirements $\mathcal{A}$ and distinct demonstrated factual-error families $\mathcal{E}$, we compute

$$R = \sum_{a \in \mathcal{A}} m_a, \quad M = \sum_{a \in \mathcal{A}} m_a z_a, \quad F = 1.25 \sum_{h \in \mathcal{E}} w_{g(h)}, \quad (6)$$

where $w_{g(h)}$ is the weight of error family h ’s semantic group. Repeated descriptions of the same erroneous fact incur one penalty. The resulting quality is

$$q = \frac{2M}{R + M + F}. \quad (7)$$

Equivalent paraphrases receive credit. Unverifiable assertions are distinct from demonstrated errors, and confidence does not multiply the score. Source-uncertain requirements are excluded from supported mass. A pair with no supported mass is unavailable. Failed attempts may still match the intended action without claiming successful execution.

Oracle and joint tasks. Oracle captioning averages q over all reference intervals in an episode. Joint annotation first constructs every temporally eligible prediction–reference edge, then finds the one-to-one assignment $\mathcal{M}$ maximizing total quality:

$$F_{\text{joint}} = \frac{2 \max_{\mathcal{M}} \sum_{(i,j) \in \mathcal{M}} q_{ij}}{|\hat{\mathcal{I}}| + |\mathcal{I}^*|}. \quad (8)$$

Empty captions retain their predicted intervals in the denominator. Joint F1 is not the product of separately averaged segmentation and caption scores. Each embodiment equally averages its fixed 100 episodes. Robot Avg equally weights the two robot tracks, with no combined human–robot score. Displayed subset scores use one decimal with half-up rounding. Robot Avg is computed from the displayed subset percentages using the same rule.

Judging. A fixed GPT-6 Astra judge evaluates candidate captions against reference requirements and sampled source evidence, with model identities withheld. It uses high reasoning with a 12,000-token response budget for human episodes and 20,000 for robot episodes. One truncated human judgment receives a single 24,000-token recovery attempt with unchanged inputs. The first valid judgment is retained. Because Astra is also an evaluated system, generator–judge self-preference is a limitation. The scores depend on the reference requirements and sampled visual evidence. Camera timestamp agreement does not independently verify hardware synchronization.

B.4. Output validity and incomplete measurements

Deterministic normalization may remove complete JSON fences, unwrap a single unambiguous container, or discard unrelated metadata. It does not rewrite captions or alter predicted intervals, apart from the stated endpoint rule. Native outputs use the trained schema and production parser. Cache reuse requires matching sources, intervals, prompts, model settings, and inputs. Technical recovery preserves the request. Returned semantic answers and refusals are not resampled to improve results.

Fixed-denominator reporting. Failed generation and unavailable judgments receive no operational credit. All episode, reference, and prediction denominators remain fixed, and resolved contributions within partially measured episodes are retained. Unavailable judgment is a measurement status, not a finding that the caption is factually wrong. We do not filter to the intersection of model-specific successful cases. The reported Flora WGO evaluation has complete oracle and predicted-caption coverage. In the robot tracks of DenseCaptionBench, five sources lack new caption generation and 98 judge jobs remain unavailable across oracle and joint tasks. Previously measured temporal scores for the unchanged segmenter are retained. These gaps affect the operational scores and preclude interpreting ranking marks as a fully observed ordering.

Technical recovery. WGO baseline requests generally allow up to three transport attempts with a 1,200-second timeout. Kimi and GPT-5.6 Luna use first-attempt generation. Robot baseline generation allows at most two attempts with a 300-second timeout. Archived baseline judgments allow up to six cumulative technical attempts, with at most 900 seconds for the final attempt. New Flora RL judgments allow at most two identical attempts. Recovery is restricted to technical failures with no retained valid verdict. Ambiguous transmitted requests and refusals are not reissued under the final RL protocol. These historical recovery budgets differ across runs and do not constitute matched service-compute budgets.

B.5. WGO-Bench protocol

WGO-Bench [17] contains 25 HomER, 50 DROID, and 25 Galaxea episodes with 470, 150, and 123 reference segments, respectively. The benchmark provides single-view videos and episode instructions. Following its public evaluation protocol, scoring copies snap the first predicted start and last predicted end to the corresponding reference outer edges. Raw predictions remain unchanged. Segmentation uses maximum-cardinality one-to-one matching at $\text{tIoU} \geq 0.75$. Gold-window caption accuracy uses all reference targets. Joint matching additionally requires caption acceptance under the public text-only rubric, judged by Gemini 3.5 Flash at temperature zero with an 8,192-token budget. Counts are pooled within each subset, and reported averages equally weight the three subset scores. The displayed average follows the rounded-subset convention used in the main tables. Where intervals are reported, we use 2,000 paired episode-bootstrap draws stratified by subset.

All systems receive the episode instruction in both generation stages. Reference caption text is never supplied. Flora uses the single-view inputs and native clocks with its fixed production workflow, without additional benchmark-specific event merging or refinement. Thus the WGO results test transfer to the benchmark’s annotation policy without tuning boundaries to its reference labels.

B.6. Supplementary pipeline comparisons

Protocol. The common pipeline, Refiner [18], and LeRobot annotate [19] share Gemini 3.5 Flash in Figure 5, following Refiner’s published reference setting [17]. Table 9 repeats the comparison with Gemini 3.8 Flash [31]. These are our evaluations on the fixed WGO population. Both backbones use temperature zero and a 32,768-token output limit. Refiner and LeRobot retain their native prompts and visual plans at revisions e402a667 and eb7a54c6. Pegasus 1.5 [20] and trained Flora retain their own backbones.

For oracle captioning, Refiner uses its fixed-window labeling prompt without a seed caption, and LeRobot uses its visual-description component on each reference interval. Pegasus receives an indexed list of reference intervals with full-episode video context. Every system receives the episode instruction, but no reference caption

Table 8: **WGO-Bench pipeline comparison.** Subset scores and equal-weight averages (% , $\uparrow$). Subsequent off-the-shelf model comparisons use the **common pipeline** as a simple reference protocol that standardizes input construction and prompts while limiting API-call and visual-token overhead.

Pipeline / backbone	Segmentation F1@0.75				Captioning Gold-window accuracy				Joint annotation Semantic F1@0.75			
	HomER	DROID	Galaxea	Avg	HomER	DROID	Galaxea	Avg	HomER	DROID	Galaxea	Avg
Shared backbone: Gemini 3.5 Flash [36]												
Common pipeline	13.2	28.6	34.0	25.3	55.7	56.7	59.3	57.2	10.7	19.7	26.4	18.9
Refiner [2026]	22.0	31.4	46.9	33.4	57.7	65.3	69.9	64.3	15.8	26.5	39.4	27.2
LeRobot annotate [2026]	14.2	28.3	31.0	24.5	47.9	34.7	41.5	41.4	8.0	20.3	22.9	17.1
Model-specific backbones												
Pegasus 1.5 [2026]	10.7	44.8	28.1	27.9	48.1	52.7	49.6	50.1	8.8	34.0	27.1	23.3
Flora (ours)	23.9	56.1	70.8	50.3	64.0	55.3	74.8	64.7	17.9	32.5	53.1	34.5

text. All three tasks use the same WGO scorer. Native visual representations, prompt sequences, and compute remain different.

Backbone sensitivity. Refiner’s mean joint F1 increases from 27.2% with Gemini 3.5 Flash to 36.4% with Gemini 3.8 Flash, compared with 34.5% for trained Flora. Flora retains the highest mean segmentation score, while Refiner leads caption accuracy and joint F1 in the Gemini 3.8 comparison. Pipeline rankings therefore depend on the generation backbone.

API substitution. As a diagnostic, we also replace both Flora models with the shared API backbone while retaining production windows, prompts, continuation stitching, and caption context. Selected frames are sent as timestamped PNG images rather than native video tokens. A bare segment array may be wrapped in the required object without changing its entries. With Gemini 3.5 Flash, this variant reaches 19.4% segmentation F1, 52.1% oracle accuracy, and 11.1% joint F1. Gemini 3.8 Flash improves these to 21.2%, 72.1%, and 16.4%. Neither substitution matches the trained system, so these experiments do not establish an advantage of the workflow alone.

The Gemini 3.8 reruns preserve one source exclusion following a prior refusal, including its 20 reference targets in the denominators. Previously completed common-pipeline and Pegasus outputs are retained. Reruns use a 300-second timeout and at most three explicit service-error attempts. Native format correction and adaptive window processing remain part of their respective pipelines. Unavailable captions receive no semantic match credit, while valid segmentation outputs remain eligible for temporal scoring.

Table 9: **Pipeline comparison with Gemini 3.8 Flash.** Scores are percentages ($\uparrow$). Avg equally weights the three WGO subsets. Green numbers identify Flora variants. Shaded cells mark the best averages, including ties. The Pegasus and trained Flora results match Figure 5.

Pipeline	Backbone	Subset	Segmentation F1@0.75	Captioning Accuracy	Joint F1@0.75
Common pipeline	Gemini 3.8 Flash	HomER	23.9	63.0	20.6
		DROID	30.8	70.0	23.7
		Galaxea	24.5	79.7	20.9
		Avg	26.4	70.9	21.7
Refiner	Gemini 3.8 Flash	HomER	26.3	60.0	20.7
		DROID	48.2	72.7	41.0
		Galaxea	51.5	88.6	47.6
		Avg	42.0	73.8	36.4
LeRobot annotate	Gemini 3.8 Flash	HomER	18.7	58.7	14.1
		DROID	35.2	49.3	28.7
		Galaxea	48.9	52.0	40.9
		Avg	34.3	53.3	27.9
Flora (API)	Gemini 3.8 Flash	HomER	12.3	56.6	9.9
		DROID	18.1	72.7	13.2
		Galaxea	33.1	87.0	26.1
		Avg	21.2	72.1	16.4
Pegasus 1.5	Pegasus 1.5	HomER	10.7	48.1	8.8
		DROID	44.8	52.7	34.0
		Galaxea	28.1	49.6	27.1
		Avg	27.9	50.1	23.3
Flora	Qwen3.5 9B / 27B	HomER	23.9	64.0	17.9
		DROID	56.1	55.3	32.5
		Galaxea	70.8	74.8	53.1
		Avg	50.3	64.7	34.5

C. Reward Evaluation Protocol

All reward results in Section 2.2 use the same Flora-Rewarder model.

Baselines.

General-purpose baselines receive the task instruction and the frames given to Flora-Rewarder, with the same prompts and answer formats. API models request medium reasoning where supported, and reasoning compute is not equated across models. Results for specialized reward models are taken from their publications and keep their original protocols. RynnValue ranks 976 trajectories rather than 567, and the Robometer row takes the best value reported across its variants.

RoboRewardBench.

Flora-Rewarder answers in score-only mode from $\min(64, \max(2, \lfloor 10T \rfloor))$ uniformly sampled frames, with greedy decoding and thinking disabled. MAE is computed on the 1–5 scale and multiplied by 100. Baselines allow 8,192 output tokens, and only truncated queries are retried with 16,384. Qwen3.5-397B-A17B receives 32 frames and the other baselines up to 64. Qwen3.5-27B answers with thinking disabled. Kimi K3’s MAE covers 2,815 of the 2,831 episodes.

RobometerBench.

Policy ranking uses 567 trajectories from six OOD datasets. Kendall’s τ_a is averaged over tasks within each dataset and then equally over datasets. Flora-Rewarder ranks trajectories by $\hat{y}$ (Eq. (19)), and general-purpose models by their integer scores. VOC is the Pearson correlation between predicted progress and elapsed time on 255 ID and 105 OOD successful trajectories. For Flora-Rewarder, let $q_t = \sum_k w_k p_\psi(a_t = k)$ be the expected quality of step t under the dense reward, with weights w_k of 0, 0.5, and 1 for failure, suboptimal, and normal. Progress at step t is then $\hat{y} \sum_{\tau \leq t} q_\tau / \sum_\tau q_\tau$. General-purpose models predict integer progress scores. Progress that grows linearly with time scores 100 on this metric.

DenseRewardBench.

The 40 episodes are excluded from dense training (Appendix G.2). Labels are compared at 1 Hz on 4,861 time steps (3,942 normal, 619 suboptimal, and 300 failure), and all metrics pool these steps. All models label the same 508 windows of 64 frames at 4 FPS, and predictions are stitched by Eq. (22), dropping the last 4 s of every non-final window. Flora-Rewarder decodes greedily under its answer format with thinking disabled. Baselines receive each window as 64 still images (32 for Qwen3.5-397B-A17B), with the same prompt and a note on frame grouping. They answer in free text with 8,192 output tokens, and only truncated malformed answers are retried with 16,384 and then 32,768 tokens.

RubricBench.

The 40 teleoperated episodes come from 10 tasks and share no episode or task name with the rubric training pool. Each camera is a separate video, sampled at 1 FPS with at most 83 frames for baselines and at 2 FPS with at most 166 frames for Flora-Rewarder. Flora-Rewarder answers in thinking mode with temperature 1.0 and up to 20,480 response tokens. It draws nine samples per episode and answers with their medoid, the sample with the smallest summed L1 distance to the others. We report the expected error of this answer over random draws of nine from 32 samples per episode. Baselines use one sample with the same decoding settings, with two exceptions. GPT-6 Astra and Qwen3.5-27B were evaluated with greedy decoding and a 16,384-token budget. The GPT-5.6 models and Claude Sonnet 5 do not apply a temperature setting. GPT-6 Astra’s parse failure is an unserved request. On the 39 answered episodes its L1 is 3.7. Kimi K3’s parse failure exhausted the response budget.

D. Annotation Cost and Throughput

D.1. Evaluation cohort and accounting

Figures 3 and 8 use 50 fixed DenseCaptionBench episodes: 20 human, 15 parallel-gripper, and 15 dexterous-hand episodes, totaling 4,483.54 seconds of source video. Quality equally averages segmentation F1@0.5, oracle caption point quality, and joint point F1@0.5 within each embodiment, then equally averages the three embodiments. The subset differs from the complete DenseCaptionBench evaluation in Table 3.

Quality uncertainty. Failed or unavailable contributions receive no operational credit under Appendix B.4. Flora uses the selected RL pair and its current reference snapshot. API results retain the cost study’s original reference snapshot. The resulting comparison is not fully reference-version matched. Figure 8 shows 95% intervals from 10,000 paired episode-bootstrap draws within embodiment. Figure 3 shows mean estimates only. The intervals quantify episode-sampling uncertainty, excluding generation, judge, reference, and service uncertainty. Unresolved native contributions separately bound mean quality at 57.1–59.6%. The displayed points and retained bootstrap intervals use operational scores with unavailable contributions assigned no credit.

Reasoning settings. Minimum, intermediate, and maximum denote requested provider settings. GPT-6 Astra, GPT-5.6 Sol, Claude Opus 5, and Claude Sonnet 5 use low, medium, and max. Gemini 3.8 Flash and Gemini 3.1 Pro use low, medium, and high. These settings are not compute matched across providers. The left panel of Figure 3 compares Flora with maximum-effort Astra, Claude Opus 5, and Gemini 3.8 Flash. Its dashed line traces the empirical API-only Pareto frontier. Connecting lines are visual guides, not interpolated evaluations. Pareto comparisons concern displayed mean estimates under the stated assumptions, not statistical dominance.

API cost. We price returned usage at public standard rates from October 4, 2026 and divide by source-video hours. The accounting includes segmentation and predicted-interval caption generation, excluding training, reference construction, oracle captioning, and judging. Input, cache-read, cache-write, and output usage are counted once, with reasoning included in output tokens. All physical attempts with returned usage are included. Of 7,062 attempts, 6,956 report usage and 106 remain unpriced. Affected costs are therefore known-usage subtotals, not complete invoices.

D.2. Native deployment model

The Flora cost estimate uses the same model sizes as the evaluated system, a 9B segmenter and a 27B captioner. Its reference workload of 660 windows and 890 captions comes from the earlier deployment study. It is a workload assumption, not a measured invocation count or serving rate for the RL pair. An eight-GB300 deployment model combines peak compute, bandwidth, and a production-rate envelope, assuming data parallelism and batch size 16. The conditional rate is $S = 1.23495$ episodes/s, with engineering scenarios of 0.70804–1.74862 episodes/s. The production GPU-hour denominator has not been verified.

For N episodes, g GPUs, total source duration H hours, and an assumed GPU-hour price p , the cost per source-video hour is

$$C_{\text{Flora}} = \frac{pgN}{3600HS}. \quad (9)$$

With $N = 50$, $H = 1.24543$, $g = 8$, and $p = 10$ USD/GPU-hour, this gives \$0.7224 per source-video hour, with scenarios of \$0.5102–1.2600. Against maximum-effort Astra (\$88.4922), the cost ratio is 122.49, summarized as approximately $125\times$. These ranges represent engineering scenarios, not confidence intervals. Figure 3 shows the reference cost only. GPU pricing is assumed. Storage, billing granularity, and deployment overheads are excluded.

D.3. Conditional throughput comparison

For API setting m , let A_m be physical segmentation and predicted-caption requests and $T_{m,k}$ the accounted token demand of quota type k over the cohort. A request-per-minute limit R_m and token-per-minute limits

$L_{m,k}$ give the idealized quota ceiling

$$Q_m = H \min \left(\frac{R_m}{A_m}, \min_{k \in \mathcal{K}_m} \frac{L_{m,k}}{T_{m,k}} \right) \quad \text{source-video hours/min.} \quad (10)$$

Table 10 lists the selected quota scenarios. They are fixed comparison assumptions from the October 2026 analysis, not verified account entitlements or universal provider limits.

Table 10: **Quota scenarios for throughput estimates.** RPM denotes requests per minute. M denotes million tokens per minute.

Models	RPM	Input/admission	Output
GPT-6 Astra / GPT-5.6 Sol	15,000	40M	Included in admission proxy
Claude Opus 5 / Sonnet 5	10,000	10M	2M, including reasoning
Gemini 3.8 Flash	20,000	20M	No separate allowance modeled
Gemini 3.1 Pro	2,000	8M	No separate allowance modeled

The OpenAI scenario uses $\max(32,768, \text{prompt tokens})$ per attempt as an admission proxy. Actual multimodal quota debits are unverified. Claude input demand includes uncached input and cache writes, excluding cache reads. Gemini demand includes cached input in its input-token quota. The selected scenarios follow provider rate-limit documentation and the comparison profiles available at analysis time.¹ Missing usage makes some token demands incomplete. Attained service speed, spending limits, daily/monthly quotas, and dedicated provisioning are not modeled. The estimates assume sufficient concurrency and stable service.

For r independent eight-GPU replicas, native throughput is modeled as

$$Q_{8r} = r \frac{60HS}{N}. \quad (11)$$

One replica gives 1.8457 source-video hours/min. We display 12 replicas (96 GPUs), yielding 22.15 hours/min at an assumed \$16.00/min. This configuration fits within the lowest quota-implied API spending rate in the comparison (\$16.08/min). API points retain their individual quota profiles and spending rates, so the comparison is not pointwise budget matched. Linear replica scaling is assumed, with no measured 96-GPU serving result or model for storage, decoding, networking, and interference.

¹OpenAI rate limits, Claude rate limits, and Gemini rate limits.

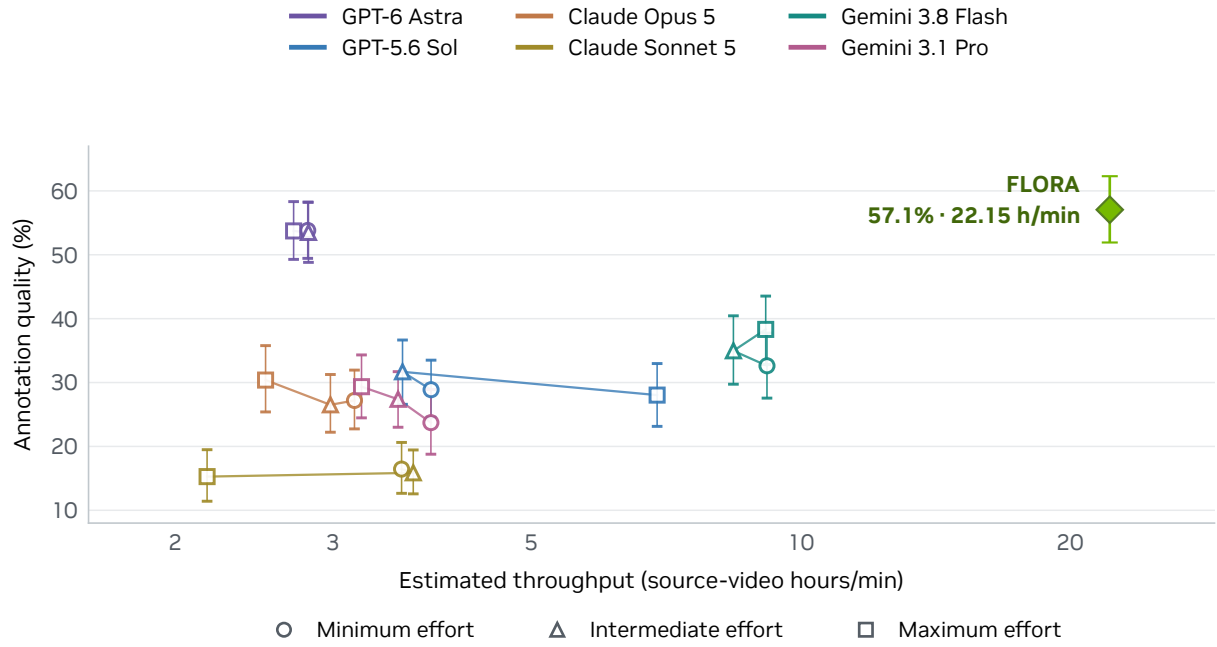
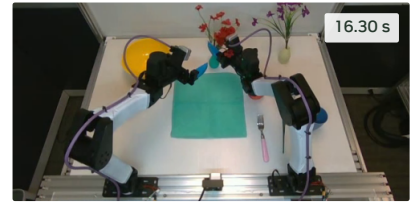
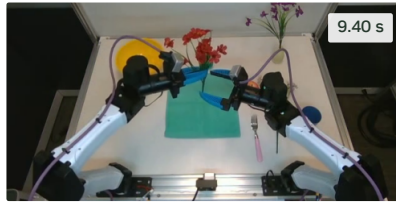
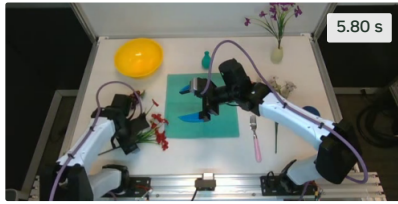


Figure 8: **Annotation quality versus conditional throughput.** The 50-episode quality scores match Figure 3. API rates follow the quota scenarios in Table 10. The Flora rate assumes 12 independent eight-GPU replicas. Vertical bars are episode-bootstrap 95% intervals. Throughput is modeled, not measured, and the points are not budget matched.

E. Additional Qualitative Evidence

Actions unfold over time

Flower arrangement



Pick up

2.40–7.70 s

“pick up the red flowers from the bottom left of the white surface **with the left hand.**”

Hand over

7.70–11.40 s

“hand over the red flowers **from the left hand to the right hand.**”

Insert

11.40–18.00 s

“insert the red flowers into the blue vase **with the right hand.**”

Quality changes within an attempt

Toy-car assembly



Normal



Failure



Normal

40.00 s

42.87 s

48.40 s

51.00 s

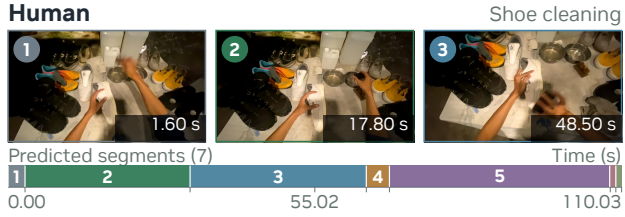
Failure rationale

“the left hand has a poor grasp on the toy car”

Figure 9: **Actions and rewards over time.** Three consecutive actions (top) and a localized grasp failure (bottom), from two separate episodes.

Episode frames and temporal segments

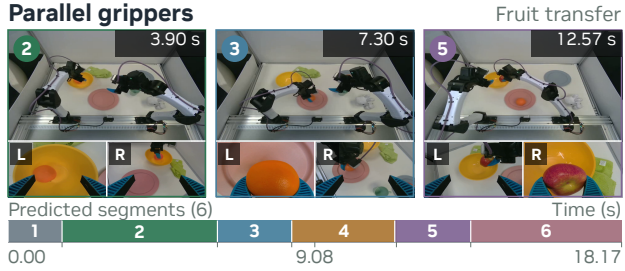
Human



Selected captions

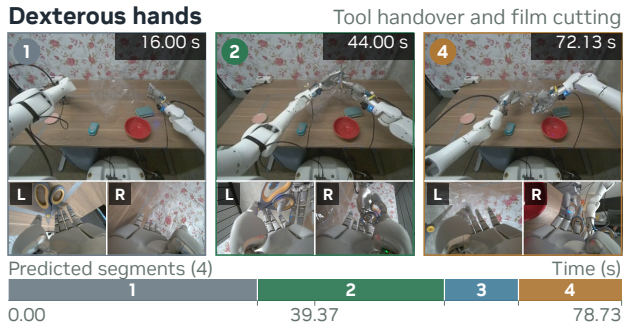
- 2 3.05–32.65 s
hold shoe with left hand, dip brush in bowl of liquid with right hand
- 3 32.65–64.25 s
hold shoe with left hand, scrub shoe sole with brush in right hand

Parallel grippers



- 2 1.60–6.20 s
Pick up the orange from the yellow bowl with the left hand.
- 3 6.20–8.40 s
Put the orange on the pink plate with the left hand.
- 5 11.45–13.70 s
Put the apple into the yellow bowl with the right hand.

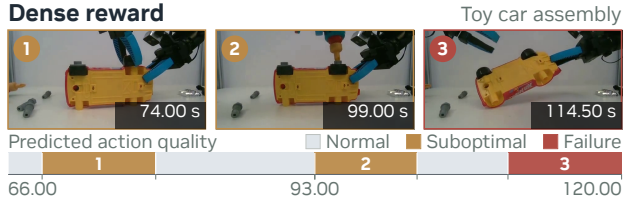
Dexterous hands



- 1 0.00–32.00 s
Pick up the scissors from the table with the left hand.
- 2 32.00–56.00 s
Pass the scissors from the left hand to the right hand.
- 4 65.50–78.73 s
Snip the cellophane with the scissors in the right hand while holding the sheet steady with the left hand.

Episode frames and rewards

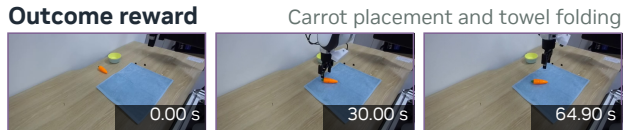
Dense reward



Selected rationales

- 2 93.00–102.00 s · Suboptimal
the toy power drill is colliding with the gray toy screw in the beginning
- 3 110.00–120.00 s · Failure
the right hand has a poor grasp on the toy car body

Outcome reward



- Outcome score 3 / 5 ●●●○○
The robot picked up the carrot and placed it on the towel, but it did not fold the towel.

Figure 10: **Example episode outputs across embodiments.** Timestamped intended-action captions appear above dense execution-quality and outcome labels. Robot mosaics retain synchronized scene and wrist views. Selected captions and rationales illustrate the structured records generated by the engine.

We qualitatively compare Flora to GPT-6 Astra on captioning and reward labeling. For annotation, we use paired episode diagnostics and a reference-informed sampled review of 12 oracle windows. Figure 11 shows two temporal examples and four caption examples using fixed references and fixed-protocol judgments.

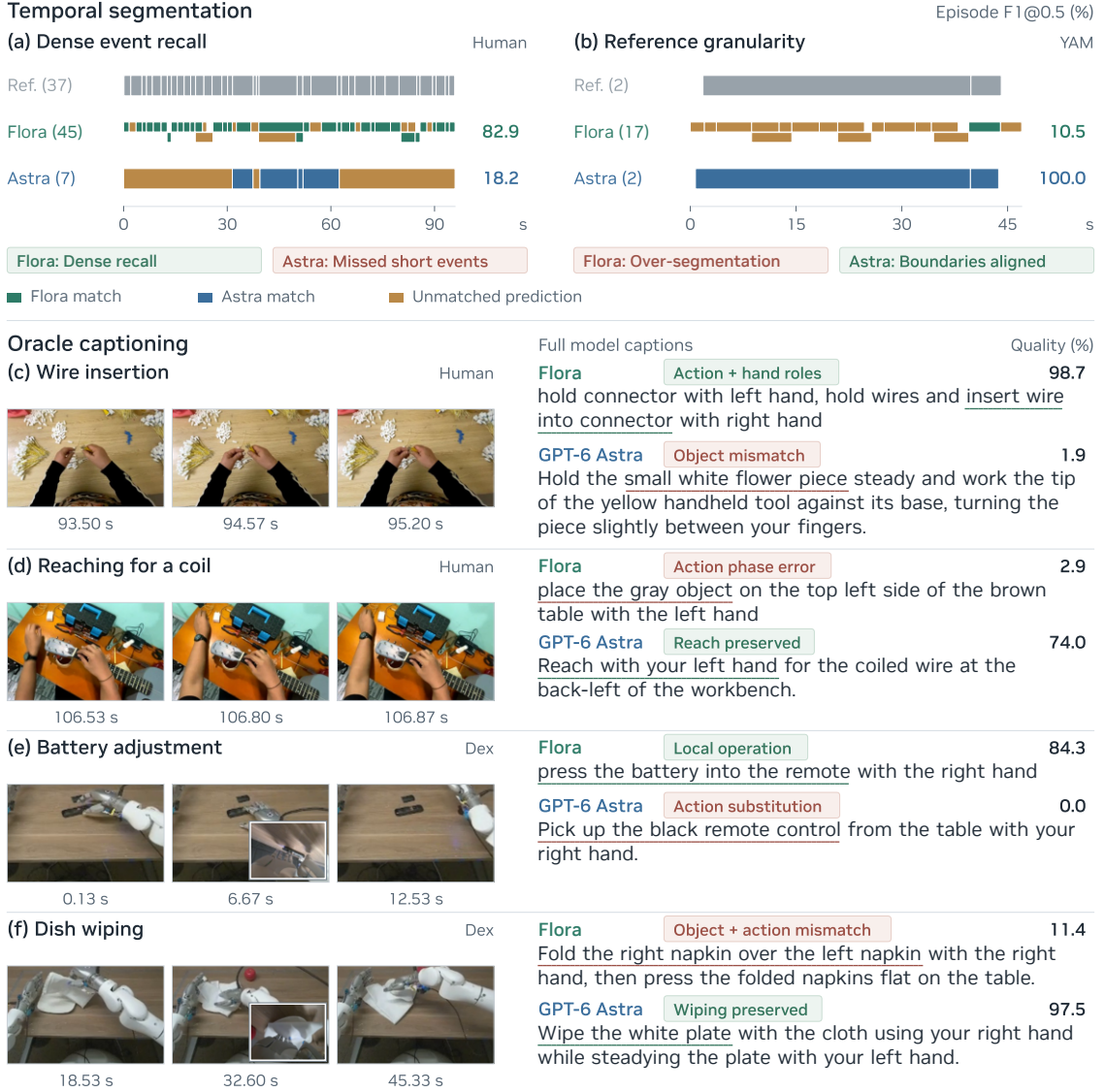


Figure 11: **Qualitative comparison with GPT-6 Astra.** Flora captures better dense actions (a) and local manipulation details (c,e), but can over-segment coarse events (b) and confuse action phases or objects (d,f). Parentheses denote segment counts. Temporal and caption scores are F1@0.5 and point quality on reference intervals, respectively. Robot insets show right-wrist views.

Temporal coverage and granularity. Panel (a) illustrates the advantage on dense actions: Flora matches nearly all reference intervals, whereas Astra matches only a few. The additional segments can also fragment coarse reference events: in panel (b), Flora splits two references into many short segments and matches only one, while Astra matches both. On mutually observed dexterous-hand episodes, recall is similar, but Flora is less precise. The aggregate parallel-gripper ordering reverses on mutually observed episodes.

Local operations and object identity. Flora preserves wire insertion in panel (c), where Astra misidentifies a connector as a flower piece, and battery adjustment in panel (e), where Astra substitutes remote pickup. Conversely, Flora substitutes placement for reaching in panel (d) and napkin folding for dish wiping in panel (f). On jointly judged human windows, Flora covers coordination requirements far more often than Astra (58.3% versus 21.2%), while action coverage is similar. Different input pipelines and reference types limit causal attribution.



Figure 12: **Flora-Rewarder flags failures and credits progress that GPT-6 Astra misjudges.** Selected examples. (a–d) DenseRewardBench: Astra calls a falling wing and a missed grasp suboptimal, although its own rationales describe both. (e,f) RoboRewardBench: Astra scores a partial and a full success too low. (a,b) 50 s excerpts with per-second labels. (c,d) the dashed 16 s windows. Right-hand numbers give agreement with the reference (%).

Failure severity. Astra understates severity, even when its own rationale describes a failure. In panel (c), it states that the wing falls onto the table but labels the drop suboptimal, and in panel (d), it labels gripper contacts “without securing a grasp” as suboptimal. Flora-Rewarder labels both events as failures. Over the longer excerpts in panels (a) and (b), its labels follow the reference segments. Higher failure recall has a cost: Flora-Rewarder also labels about 11% of normal steps as failures.

Partial outcomes. Flora-Rewarder’s advantage comes from partially completed tasks. On complete failures and successes, Astra is at least as accurate. In panels (e) and (f), Astra gives the lowest score to a bottle that was toppled but not knocked off the table, and scores a completed fold 3 instead of 5. Flora-Rewarder’s rationales name the progress behind each score.

F. Timestamped Captioning: Implementation and Training

F.1. Temporal reconstruction and visual context

The segmenter uses overlapping windows with a central emission core and context guards (Section 4.1). Window-local timestamps are projected into the episode clock. Stitching associates compatible observations using temporal overlap and continuation flags, while preserving independent concurrent actions. Guard-only fragments do not create an event for that window. A bounded set of shifted windows refines unresolved boundaries. Unobserved endpoints retain the observed extent without inventing a transition.

Window-local predictions. For sampled frames X_{W_k} and a window prompt q_k , the segmenter predicts local intervals and continuation flags ℓ_{kj}, r_{kj} :

$$\mathcal{F}_k = f_\theta(X_{W_k}, q_k) = \{(u_{kj}, v_{kj}, \ell_{kj}, r_{kj})\}_{j=1}^{n_k}, \quad (12)$$

The projection Φ_k converts local timestamps to episode time. Stitching $\mathcal{S}$ joins compatible fragments while preserving concurrent actions:

$$\mathcal{I} = \{I_i\}_{i=1}^N = \mathcal{S}(\{(W_k, \Phi_k(\mathcal{F}_k))\}_{k=1}^K). \quad (13)$$

Caption context. After stitching, the nearest non-overlapping events provide context for a target interval $I_i = [s_i, e_i]$:

$$j_i^- \in \arg \max_{j: e_j \leq s_i} e_j, \quad j_i^+ \in \arg \min_{j: s_j \geq e_i} s_j. \quad (14)$$

The corresponding visual blocks in Eq. (2) are $X_i^- = X_{I_{j_i^-}}$, $X_i = X_{I_i}$, and $X_i^+ = X_{I_{j_i^+}}$. Ties favor the shorter neighboring interval. Unavailable neighbors are omitted. Each block has its own frame budget and retains source timestamps. These adjacent-event blocks are used for training and production captioning. Evaluation on DenseCaptionBench instead restricts context to at most 2 seconds on either side of the target, as specified in Appendix B.2.

Multiple views. At each sampled source time, the main camera occupies the upper panel of a mosaic, with the left and right wrist cameras below. This arrangement is fixed across time and across the before, target, and after blocks. The prompt identifies the camera roles and requests one account of the intended action across views. Single-view inputs preserve the same temporal block structure.

F.2. Supervised fine-tuning

We initialize the segmenter f_θ and captioner g_ϕ from Qwen3.5-9B and Qwen3.5-27B [23], respectively. Supervision comprises timestamped action annotations from heterogeneous human and robot manipulation recordings. Segmentation targets serialize local intervals and continuation flags. Captioning targets pair an action instruction with its annotated interval and visual context. Both models minimize cross-entropy on output tokens only:

$$\mathcal{L}_\eta(\mathcal{B}) = - \frac{\sum_{(x,y) \in \mathcal{B}} \sum_{t=1}^{|y|} \log p_\eta(y_t \mid x, y_{<t})}{\sum_{(x,y) \in \mathcal{B}} |y|}, \quad \eta \in \{\theta, \phi, \psi\}. \quad (15)$$

Here x contains the visual evidence and task prompt, and y is the segmentation output or action instruction. The segmenter updates rank-32 LoRA adapters with scaling $\alpha = 64$ in the language model’s attention and MLP projections. The captioner updates the language model and visual merger, while freezing the visual patch embedding and transformer blocks. SFT uses AdamW with gradient clipping. Training uses sharding and activation checkpointing for the captioner.

F.3. Temporal segmentation reinforcement learning

We optimize the segmenter with GRPO [25] using a verifiable reward. Predictions and references are clipped to the emission core, and guard-only fragments are discarded. Maximum-cardinality one-to-one matching at temporal-IoU threshold τ defines F1 F_τ and precision P_τ . The reward is

$$r_{\text{seg}} = 0.45F_{0.5} + 0.15F_{0.75} + 0.25P_{0.5} + 0.15B. \quad (16)$$

For each matched pair at tIoU 0.5, let d be the sum of absolute endpoint errors, augmented by 0.5 seconds for each incorrect continuation flag. The boundary term B averages $\max(0, 1 - d/(1\text{ s}))$ over these pairs. Matching first maximizes cardinality, then minimizes total clamped error. If no pair matches, $B = 0$, except when both sets are empty, for which $B = 1$. Both-empty sets receive reward one. Invalid or truncated outputs receive zero.

Rollout advantages subtract the group mean without standard-deviation normalization. Only LoRA adapters are updated, with KL regularization to the fixed SFT policy.

Checkpoint selection. A separate validation set determines the reported checkpoint. Selection maximizes a prespecified weighted F1@0.5, subject to at most a two-percentage-point drop in each guarded validation group. Benchmark results are not used for selection.

F.4. Captioner reinforcement learning

The captioner uses a reference-grounded reward based on intended action, object identity, manipulator coordination, and relevant detail. Source video and reference requirements define supported mass R , covered mass M , and factual-error penalty F :

$$r_{\text{cap}} = \frac{2M}{R + M + F}. \quad (17)$$

The group weights and phase gates follow the point-quality formulation in Appendix B.3. Reference instruction text is never part of the captioner’s generation input.

Rewards are centered within each rollout group without standard-deviation normalization. The language model is updated while the visual encoder and merger remain frozen. Reference instructions remain unavailable to the generation input.

Checkpoint selection. Validation equally weights the three embodiments and precedes benchmark evaluation. We maximize mean quality among checkpoints without per-track regression. If none qualifies, we maximize the worst-track change. The segmenter and captioner are selected independently.

F.5. Pipeline schematics

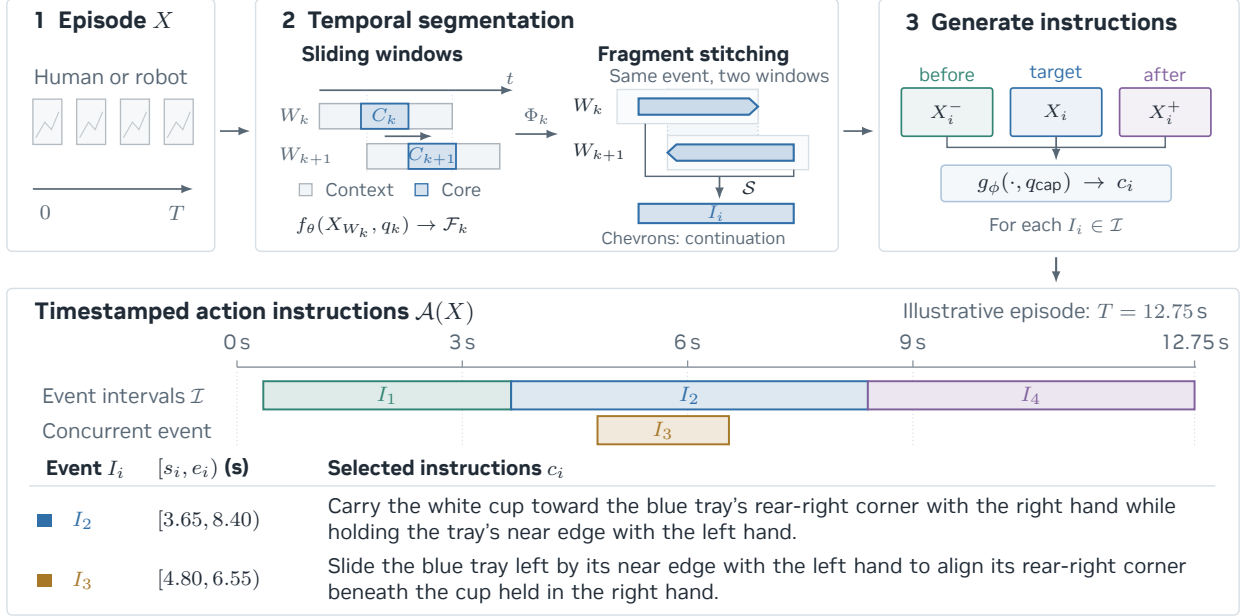


Figure 13: **Windowed localization and contextual captioning.** Overlapping windows emit fragments from their central regions. Timestamp projection and stitching reconstruct episode intervals, which are captioned independently with before, target, and after visual evidence.

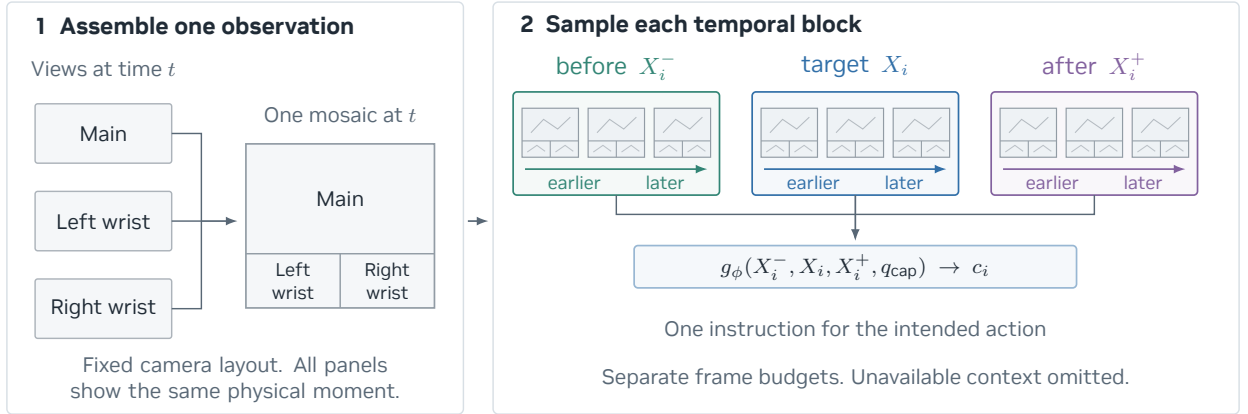


Figure 14: **Synchronized views and temporal context.** Each mosaic aligns cameras at a source time. Mosaics retain their temporal order within the before, target, and after blocks.

G. Reward Modeling Implementation and Training

This appendix details the inputs, data, prompts, and training of Flora-Rewarder (Section 4.2). Benchmark protocols and results appear in Section 2.2.

Reward interfaces. With a task instruction ℓ , the outcome and dense interfaces return

$$\mathcal{R}(X) = (\{a_t\}_t, s), \quad a_t \in \{1, 2, 3\}, \quad s \in \{1, \dots, 5\}, \quad (18)$$

Here a_t is execution quality for a 0.5-second group, and s is the episode outcome. Score-token probabilities define the continuous outcome reward:

$$p_k = p_\psi(s = k \mid X, \ell, q_{\text{out}}), \quad \mathbb{E}[s] = \sum_{k=1}^5 k p_k, \quad \hat{y} = \frac{\mathbb{E}[s] - 1}{4} \in [0, 1]. \quad (19)$$

For each dense window, the output contains its worst label, ordered quality sequence, and rationales for non-normal spans:

$$r_\psi(X_{W_k}, \ell, q_{\text{dense}}) = (w_k, (a_{k,1}, \dots, a_{k,32}), \{(u_{kj}, v_{kj}, c_{kj})\}_j), \quad w_k = \min_g a_{k,g}, \quad (20)$$

Given a rubric ρ , grading is constrained to allowed values and compared with human grades by L1 error:

$$\hat{\mathbf{y}} = r_\psi(X, \ell, \rho, q_{\text{rub}}) \in V_1 \times \dots \times V_n, \quad R_\rho(X) = \sum_{j=1}^n \hat{y}_j, \quad l(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^n |\hat{y}_j - y_j|, \quad (21)$$

G.1. Video Input

Multi-view mosaics. Outcome and dense prompts receive one video in which each sampled time is a single mosaic of up to four synchronized views. The layout is 1×1 , 1×2 , 1×3 , or 2×2 for one to four views. Camera roles determine the placement. A 1×3 mosaic orders the left, center, and right views. A 2×2 mosaic places the scene (top or front) view at the upper left, a second scene or wrist view at the upper right, and the fixed left and right cameras in the lower row. When more than four views exist, scene and fixed side cameras take precedence over wrist views, and downscaled duplicates of a stream are dropped. Each view is resized into its cell without cropping or padding. Cell sides are multiples of 32 pixels and target about 10^5 pixels per view, and the row height is chosen to minimize aspect-ratio distortion. Mosaics carry no gutters, borders, or text. A 64-frame clip is capped at 14,336 visual tokens. The bimanual recordings used for dense rewards produce a 1152×288 mosaic for three views (left, top, right) and a 768×576 mosaic for four views (top, bottom, left, right), each with 384×288 cells. A 64-frame clip of the latter occupies 13,824 visual tokens.

Timestamps. For outcome and dense prompts, we pass the source frame indices and frame rate of every sampled frame to the model’s video processor instead of resampling inside it. The timestamp tokens of Qwen3.5 therefore refer to absolute episode seconds in both training and inference, for every source frame rate.

Rubric inputs. Rubric prompts instead receive each camera as a separate video, labeled Video 1, Video 2, and so on in the text. Frames are sampled at 1 FPS, uniformly subsampled to at most 82 frames per camera, and resized to between 65,536 and 131,072 pixels each.

G.2. Dense Reward

Annotation protocol. Annotators fill one table per episode. Each row names an action, gives its start and end times in seconds, and assigns an outcome of success, normal, suboptimal, or failure. Success and normal both map to the normal label. A free-text description of the visible issue is required for suboptimal and failure actions. We convert the table to frame labels on the native 30 FPS clock. Every frame starts as normal, each action

assigns its label to its frames, and overlapping actions resolve to the more severe label (failure > suboptimal > normal). The result is stored as contiguous run-length segments with a label and issue text. A new segment begins whenever either changes. Normal segments carry no text.

Supervision. Dense labels come from human-annotated bimanual robot episodes with synchronized cameras. Training covers normal execution, recoverable inefficiencies, and failures. Episodes used for evaluation are excluded from training and validation.

Window geometry. A window is defined by an anchor time α_k . Its 64 positions sample the source video at $\alpha_k + (i - 9)/4$ s for $i = 1, \dots, 64$, so the window spans $[\alpha_k - 2, \alpha_k + 14)$ s. Positions outside the episode repeat the nearest frame. Group g contains positions $2g - 1$ and $2g$ and is valid if either of its frames lies inside the episode. Inference anchors are $\alpha_k = 10k$ s while α_k is below $\alpha_{\max} = t_{\text{last}} - 9.75$ s, followed by a final window at $\alpha_{\max}$, where t_{last} is the time of the last frame. The last frame of the episode thus falls at position 48, the end of group 24, and the final window's last 4 s repeat it. Every window has the same geometry and output length. Each group takes the more severe label of its two frames. Padded groups are supervised with the label of the boundary frame.

Training examples. Windows are sampled to cover all three quality categories and temporal transitions. The split is made at the episode level. Non-normal labels include short evidence-grounded rationales from the human annotations.

Task conditioning. Every dense prompt conditions on the episode's global task instruction, never on the sub-task annotations. Words that reveal the outcome, such as *failure*, *success*, *retry*, and *recovery*, are removed. Placeholders and opaque assembly codes make an episode ineligible. Task descriptions are normalized to identify the requested operation without revealing its outcome.

Prompt and answer. The system prompt defines the score scale and the answer grammar:

Dense reward: system prompt

```
You are a dense action-quality reward model for robot manipulation.
You watch a synchronized multiview clip and predict action quality at the finest temporal resolution visible to the model.

The clip contains exactly 64 sampled frames at 4 FPS. The vision encoder combines each consecutive pair of frames into one
temporal group, so you must predict exactly 32 ordered groups. Group 1 contains frames 1-2, group 2 contains frames 3-4,
and group 32 contains frames 63-64.
The clip is retrospective: a later recovery must not improve an earlier group and a later unrelated mistake must not
degrade it.

Use exactly this score scale:
1 = failure: the action causes or clearly constitutes a major task failure, unsafe event, destructive mistake, or loss of
task viability.
2 = suboptimal: the action is noticeably hesitant, inefficient, imprecise, or locally incorrect, but the task remains
recoverable.
3 = normal: the action is appropriate, safe, and reasonably efficient for the visible situation.

Return the worst score before the temporal sequence. Worst must equal the minimum of all 32 quality digits. Quality must
contain exactly 32 characters from 1, 2, or 3 with no spaces; character i is the score for group i.
For every contiguous group range scored 1 or 2, add one short rationale grounded only in visible evidence. Groups scored 3
get no rationale. Rationale ranges must exactly match the contiguous non-normal runs in Quality.

Return exactly:
Worst: <1, 2, or 3>
Quality:
<32 digits with no spaces>
Rationales:
<first group>-<last group> <short visible-evidence rationale>
<one line per contiguous group range scored 1 or 2, in time order>
```

The user prompt states the task and the clip's absolute time span. A training example and its target are shown below:

Dense reward: user prompt and target

```

Task: assemble the duplofish

Synchronized multiview clip:
<video>

The clip contains 64 sampled frame positions at 4 FPS (16 seconds nominal span) and 32 visible temporal groups of 2 frames
each. The sampled positions run from t=353.000 to t=368.733 seconds of the episode. Episode boundaries repeat the nearest
available frame.
Predict Worst, all 32 group scores, and rationales for non-normal runs.
-----
Worst: 1
Quality:
11111111111113333332222222222233
Rationales:
1-12 fail to place red lego block on the left edge of the lego toy vehicle due to misalignment
19-30 the lego toy vehicle slides while grasping
    
```

An all-normal window has the target Worst: 3, a string of 32 threes, and an empty Rationales: section.

Inference and stitching. Decoding is greedy and constrained by a regular expression for the answer grammar, with at most 640 new tokens. A parsed answer whose Worst value disagrees with its Quality string is rejected, and an unparseable answer is replaced by an all-normal window and flagged. Each non-final window retains groups 1–24 and discards groups 25–32. The final window retains all valid groups. Let $\mathcal{C}$ be the retained valid groups (k, g) , each contributing the source times $\tau_{k,g}$ of its two frames. A frame at time t takes the label

$$a_t = a_{k^*, g^*}, \quad (k^*, g^*) = \operatorname{lexmin}_{(k, g) \in \mathcal{C}} \left(\min_{\tau \in \tau_{k, g}} |t - \tau|, -\min(g - 1, 32 - g), k, g \right), \quad (22)$$

that is, the nearest retained group, then the group farthest from its window’s edges, then the earlier window. Consecutive frames with equal labels form segments, and each non-normal segment reports its most frequent rationale. Tail trimming keeps at least 4 s of later video for every retained group except near the episode end. The reward is retrospective and intended for offline labeling, analysis, and data filtering.

G.3. Outcome Reward

Supervision. Outcome supervision combines public, licensed, and internally collected robot recordings. Native five-point labels are retained. Binary success and failure map to 5 and 1. Partial-success values $p \in [0, 1]$ are mapped to $s = 1 + \operatorname{round}(4p)$. Failure categories are mapped only when their outcomes are unambiguous. Training pairs scores with short evidence-grounded rationales when such explanations are available.

Frame sampling. For an episode with N frames at frame rate f , we sample $m = \min(64, \max(2, \lfloor 10N/f \rfloor))$ frames, rounded down to an even number, at indices $\operatorname{round}(i(N - 1)/(m - 1))$ for $i = 0, \dots, m - 1$, so the first and last frames are always included.

Prompts. The two system prompts differ only in their last lines:

Outcome reward: system prompt

```

You are a terminal-success reward model for robot trajectories.
Given the task instruction and the complete trajectory video, predict a score from 1 to 5.
A complete failure is 1. A complete success is 5.
Judge only what is visible in the trajectory and follow the requested output format exactly.
[with rationale]
Provide one short, compact rationale followed by the score. Return exactly:
Rationale: <rationale>

Score: <1-5>
[score only]
Do not provide a rationale. Return exactly:
Score: <1-5>
    
```

Outcome reward: user prompt and target (rationale mode)

Task: Slide the pink block a short distance toward the robot on the pegboard.

Complete trajectory video:
<video>

Evaluate the terminal success of the task.

Rationale: The robot contacted the pink block and nudged it slightly toward the robot base, but the displacement was small and insufficient to complete the requested movement.

Score: 2

Score distribution. Decoding is greedy and constrained to the answer format, with at most 12 new tokens in score-only mode and 160 with a rationale. At the generated score token, we take the 20 most likely alternatives, keep those that decode to a digit from 1 to 5, and renormalize their probabilities to obtain p_k in Eq. (19). RoboRewardBench uses the decoded score directly. Trajectory ranking uses $\mathbb{E}[s]$.

G.4. Rubric-Derived Reward

Rubric format. A rubric lists task steps in order. Each step has an action description, a maximum score, and two to five labeled score levels. Human grades always take one of these levels, and the maxima sum to one. Table 11 shows an example. Rubrics for internal policy-evaluation episodes contain 2–10 steps, and rubrics for external episodes 1–4 steps.

Supervision. Rubrics describe task steps and allowed levels of partial completion. Human grades define the target scores. Training examples pair each video and rubric with a reasoning and answer sequence. The RubricBench evaluation episodes and task names are held out from this pool.

Prompt. The system prompt is a generic assistant prompt. The user turn lists the videos, followed by the template below for internal policy-evaluation episodes. For external sources, the first paragraph describes the cameras actually present.

Rubric reward: user prompt template (internal policy evaluation)

Video 1: <video>Video 2: <video>Video 3: <video>

You are an expert evaluator of robot manipulation. You are shown three time-synchronized videos of a bimanual robot performing one task: Video 1 is a top (scene) camera, Video 2 a left-wrist camera, and Video 3 a right-wrist camera.

Score how well the robot performed, using the rubric below. The rubric lists the task’s steps; each step has its own maximum number of points and an explicit list of the point values you may award for it. For every step, award exactly one of the values listed for that step -- not a value in between.

Task: {task name}

Instruction to the robot: "{instruction}"

Scoring rubric -- award one of the listed values for each step:

{i}. {action} (max {maximum})
award one of: {level} = {points} | ...

Respond with your scores as a JSON array of numbers -- one score per step, in the exact order listed above (a length-{n} array). Each number must be one of the values allowed for that step. Output the array on a single line prefixed with 'ANSWER:' and nothing after it, e.g.
ANSWER: {list of the step maxima}

The answer is parsed from the last ANSWER: line, and each value is clipped to $[0, \max_j]$.

G.5. Co-Training

Shared supervised adaptation. The released reward model uses SFT only. Outcome, dense, and rubric examples are shuffled together without task-family reweighting. Rubric answers include reasoning, while outcome and dense answers use direct structured outputs. A shared Qwen3.5-27B model updates the language model, visual merger, and visual encoder, using a lower learning rate for the visual encoder. Training minimizes

output-token cross-entropy (Eq. (15)). The input budget accommodates the longer rubric answers.

G.6. Interface Examples

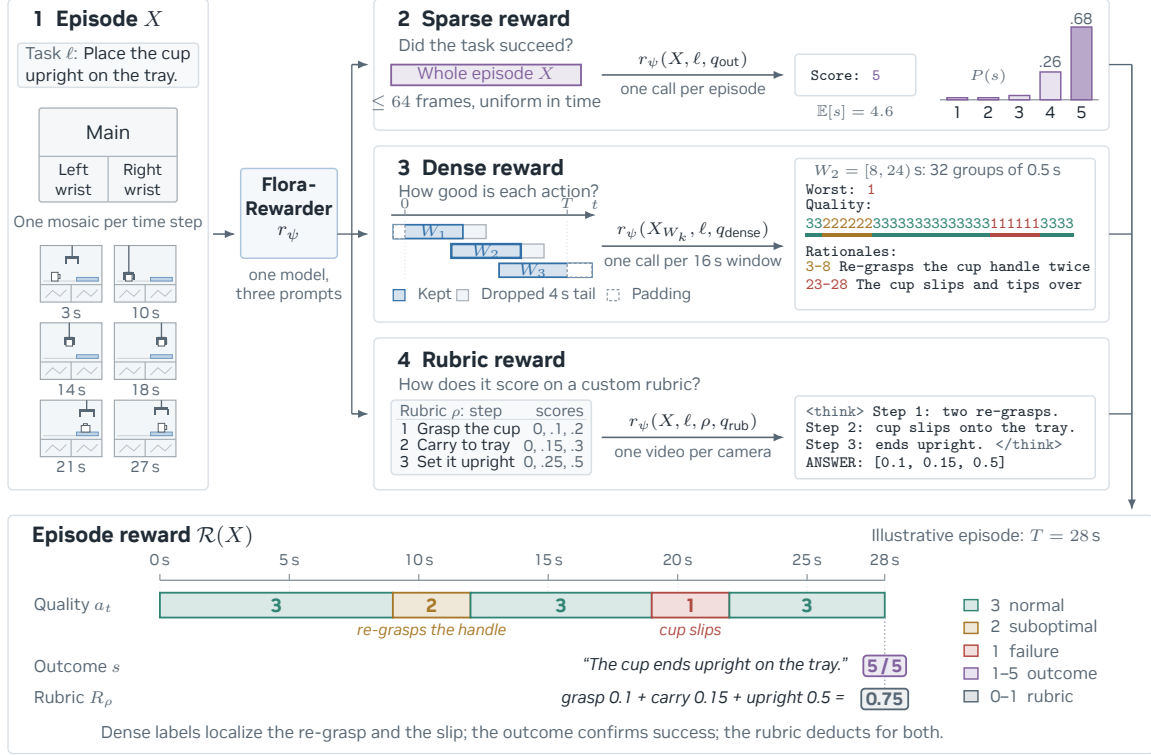


Figure 15: **Three interfaces, one reward model.** Outcome prompts score episode completion, dense prompts label temporal groups with rationales, and rubric prompts assign discrete credit to user-defined task steps.

Step	Score	Criterion
1. Firmly grasp apple	0.0	Did not grasp the apple
	0.2	Touched but did not grab the apple
	0.4	Firmly grasped the apple
2. Pick up knife	0.0	Did not pick up the knife
	0.15	Grasped the knife but dropped it
	0.3	Picked up the knife with a secure grip
3. Cut apple in half	0.0	Did not cut the apple
	0.15	Partially cut the apple
	0.3	Cut the apple fully in half
Maximum total	1.0	

Table 11: Sample scoring rubric for an apple-cutting task.

H. Related Work

Captioning for robotics. Commercial services such as Scale AI’s Physical AI Data Engine [37] provide data collection and model-assisted annotation for robotics. Open-source alternatives have recently emerged. Macrodata Labs’ Refiner [18] provides subtask segmentation and reward scoring along with the WGO-Bench benchmark [38]. LeRobot [19, 39] uses an off-the-shelf VLM to write subtasks, plans, and VQA pairs into datasets, and uses pretrained reward models such as Robometer [22] to label per-frame progress. Their captioning pipelines compose existing VLMs with task-specific prompts and processing. General video-language models such as Pegasus [40] support video understanding and summarization, while DIAL and ECoT generate language supervision for robot policy learning [6, 41].

Open-weight VLMs as annotators. Open-weight VLMs provide backbones for locally deployed annotators. General-purpose candidates include Gemma [42, 43], Qwen3-VL [44], InternVL3.5 [45], GLM-4.5V [46], and Kimi [47, 48]. CapRL [49, 50] shows that captioners can be optimized with RL for downstream utility. Flora builds on these advances but targets a narrower goal: annotations of robot and human data that are accurate, consistent, and cheap enough to cover large pretraining corpora.

Reward models for robotics. Robot reward modeling has progressed from visual representations [51, 52] and zero-shot VLM similarity [53, 54], to in-context progress estimation [55], and most recently to fine-tuned reward models. RoboReward [21] synthesizes failures to train episode-level rewards, Robometer [22] combines frame-level progress with trajectory preferences, and Robo-Dopamine [33] targets process rewards for high-precision manipulation.