SPARC: Reliable Spatial Annotations from Robot Demonstrations at Scale

Nils Blank*, Paul Mattes*, Maximilian Xiling Li*, Jakub Suliga*,
Thomas Roth*, Moritz Reuss†, Pankhuri Vanjani*, and Rudolf Lioutikov*‡
*Karlsruhe Institute of Technology †NVIDIA ‡Robotics Institute Germany

***Abstract*—Spatial annotations of robot demonstrations, capturing which object is manipulated, where it is, and how it moves, are a key ingredient for grounded policies and embodied vision-language models, yet producing them reliably at scale remains hard, especially in cluttered scenes with visually similar objects. Automated pipelines chain a detector and a tracker and rank annotations by detector confidence, which measures *recognition*, not whether the detected object is the one actually being manipulated. We present SPARC, an auto-labeling pipeline that scores each candidate with physically grounded interaction cues (phase-aware motion, 3D gripper proximity, and a robot-overlap filter) and aggregates them into a per-annotation reliability score, turning the quality-versus-coverage tradeoff into a single controllable threshold without human review. On IA-Bench, a new 1.7k-demonstration benchmark with video and proprioception across 12 embodiments, SPARC reaches 80.2% interacted-object localization accuracy (vs. 58.1% detector-only) and retains $3\times$ more data at high-precision operating points. VLMs fine-tuned on fully auto-labeled SPARC data match or beat human-annotated supervision on grounding and pointing, and reasoning policies trained on SPARC data more than double the success of policies in cluttered, high-ambiguity real-world scenes.**

I. Introduction

Spatial annotations of robot demonstrations identify which object is being manipulated, where it is located, and how it moves through time. These annotations are a key ingredient in modern robot-learning pipelines, including object-centric reasoning in vision-language-action models [37, 38, 1], pixel-level scene understanding for visually grounded policies [20], and embodied VLM mid-training [10, 16, 11]. As robot learning moves from clean laboratory setups toward production-like manipulation, reliable grounding becomes increasingly important: many objects may look similar, workspaces are cluttered, and a single object identity error can invalidate an otherwise feasible action. As demonstration datasets grow, the bottleneck therefore shifts from collecting trajectories to labeling them with reliable spatial structure, where label quality rather than sheer volume limits what downstream models can learn.

Producing such annotations reliably at scale remains difficult, and across pipelines the way they break is remarkably consistent. Real demonstrations contain clutter, occlusion, repeated objects, and large viewpoint variation, and existing automated pipelines typically chain a detector with a tracker [37, 1, 20], making the detector the sole source of object identity. Detector confidence measures recognition confidence, not whether the detected object is the one actually being manipulated; in high-ambiguity scenes such pipelines lock onto a plausible but incorrect object with high confidence and propagate that error through time. This mismatch creates a direct scale-quality tradeoff: hard thresholds on detector confidence either retain many incorrect annotations or discard many valid ones. Human review at annotation time is reliable but expensive [26, 16], and simulation provides ground truth [35, 39] but struggles to transfer to real demonstrations.

We address this with **Sp**atial **A**nnotations from **R**obot Demonstrations with Reliability **C**alibration (**SPARC**), an automatic pipeline that produces spatio-temporal annotations from robot demonstrations and a reliability score for each one. For each object-centric subtask, SPARC identifies the interacted object and generates its start and target locations, its movement over time as a 3D object trajectory, and a continuous reliability score. Instead of trusting detector confidence for object identity, SPARC performs *interaction-aware candidate annotation*: it isolates the manipulated object using physical cues, namely phase-aligned motion and 3D gripper proximity, while a spatial filter suppresses the robot's own geometry. The same interaction evidence yields a *composite reliability score* that acts as a per-annotation quality signal, letting practitioners monitor and trade annotation quality against dataset scale without human review.

To evaluate annotation quality directly, we introduce IA-Bench, a benchmark of 1.7k hand-annotated object start/target locations spanning 12 embodiments, including bimanual setups, with video and proprioception. SPARC achieves 80.2% interacted-object localization accuracy at high confidence versus 58.1% for a detection-only baseline, and stays viable under aggressive filtering where detection-only collapses to near-zero usable data. Downstream, thresholding 280k auto-labeled trajectories for 95% precision yields 167k high-quality ones versus only 30k from detector confidence, and models fine-tuned on them match or exceed human-annotated supervision without any manual verification.

Contributions. (i) We reframe automatic spatio-temporal annotation of demonstrations as a *selective-prediction* problem and introduce a per-annotation reliability score grounded in physical interaction evidence rather than detector confidence, targeting cluttered and high-ambiguity scenes where visually similar objects make recognition confidence unreliable. (ii) We release IA-Bench, an interaction-aware benchmark with video and proprioception across diverse embodiments, including settings with clutter, occlusion, and bimanual manipulation. (iii) We generate a large-scale, high-quality dataset of 167k

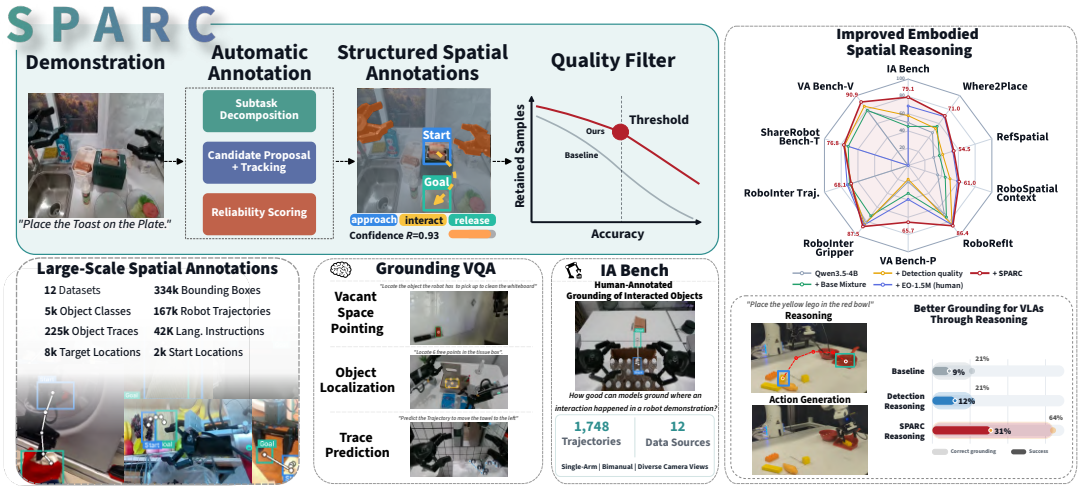

Fig. 1: **SPARC** auto-labels robot demonstrations with object-centric spatial annotations and a per-annotation reliability score derived from interaction evidence: phase-aware motion, gripper proximity, and a robot-overlap filter. A single threshold on this score controls the quality-coverage tradeoff without human review, producing large-scale annotations (*bottom*) that improve downstream embodied reasoning and policy learning (*right*).

annotated trajectories and show that fully automatic SPARC supervision matches or exceeds human-annotated data on embodied grounding and yields reasoning policies more robust in cluttered real-world manipulation, a common failure mode when moving toward production-like settings.

II. Method

Given a demonstration video, language instruction, and gripper proprio signal, SPARC produces object-centric annotations for each interaction segment: a subtask instruction, the manipulated object name, initial and target locations, an object trajectory, gripper phase boundaries, and a scalar reliability score. The pipeline has three stages: subtask decomposition, candidate proposal and tracking, and reliability scoring.

Stage 1: Subtask Decomposition. SPARC decomposes a long-horizon demonstration into short, object-centric segments, each capturing a single interaction. We extract closed-gripper intervals from the normalized gripper signal, remove short spurious closures with an adaptive duration threshold, and apply this per arm for bimanual setups. Adjacent open-gripper intervals give the approach and release phases. A VLM [31] then aligns the grasp cycles with the language instruction, returning a structured record per subtask (manipulated object name, subtask instruction, initial/target location descriptions, phase intervals) that guides the later stages.

Stage 2: Candidate Proposal and Tracking. SPARC compiles candidate objects and lifts their motion to 3D. We run an open-vocabulary detector [12] at a keyframe chosen halfway through the first grasp phase (to reduce gripper-object occlusion), using a very low threshold so it acts as a region proposer while retaining grounding scores, and segment each proposal with SAM2 [27]. We then run dense point tracking [14] over the video and lift tracks inside each object mask to 3D with a monocular geometry model [33], yielding a per-candidate 3D trajectory.

Stage 3: Reliability Scoring. Not all tracked candidates are the manipulated object, and detector confidence, an appearance signal, aligns poorly in clutter. We therefore score each candidate with three physically grounded cues. *Phase-aware motion* compares a candidate's displacement during the interaction phase against its motion outside it, favoring objects that move when manipulation actually happens while suppressing background jitter. *3D gripper proximity* measures the fraction of a candidate's 3D points that fall inside an adaptive sphere around the gripper, remaining informative even for constrained interactions with little object motion. A *robot-overlap soft penalty* suppresses the common failure of selecting the gripper itself, using overlap with a robot-segmentation mask attenuated by 3D depth separation. The three cues are normalized and combined with detector confidence into one reliability score that down-weights appearance when gripper-proximity evidence is strong. The top-scoring candidate becomes the annotation; its score is the per-annotation quality signal, and the target object is localized from where its tracks land. SPARC annotates a trajectory in $\sim$5 s on a single GPU and parallelizes across workers for a 24$\times$ wall-clock speedup over human labeling.

III. Experiments

Benchmark. We evaluate annotation quality on IA-Bench, which contains 1,748 human-annotated start/target boxes from four sources, namely AgiBotWorld [2], DROID [18], BridgeData [32], and Open X-Embodiment [25], covering 12 embodiments, single-arm and bimanual setups, clutter, occlusion, and diverse views. While IA-Bench is not a factory benchmark, it captures the grounding ambiguities that often make production-oriented manipulation difficult: visually similar objects, repeated instances, and object identity errors under clutter. We split it into 473 validation and 1,275 held-out test annotations (X-Embodiment held out zero-shot); validation

Category	Scoring rule	Acc.↑	Cov@90↑	AURC↓	E-AURC↓
Baselines	Det. conf. (OWLv2)	0.622	0.002	0.297	0.002
	Motion magnitude	0.463	0.000	0.657	0.476
	FSD: Traj. [36]	0.386	0.331	0.279	0.032
Ablation	Det. conf. (LLMDet)	0.581	0.266	0.219	0.115
	+ Mean movement	0.697	0.514	0.117	0.066
	+ Robot filter	0.727	0.549	0.101	0.059
	+ Phase-aware motion	0.778	0.735	0.064	0.037
External	Gemini Robotics ER 1.6 [30]	0.717	n/a	n/a	n/a
	Qwen3.6-30B-A3B [31]	0.060	n/a	n/a	n/a
	Ours (Final)	**0.802**	**0.776**	**0.056**	**0.035**

TABLE I: **Interaction-aware filtering ablation on IA-Bench.** *Baselines*: prior-style confidence and motion filters. *Ablation*: starting from LLMDet detection confidence, we sequentially add each component. *External*: general-purpose models for reference. Coverage is at the 90% precision operating point.

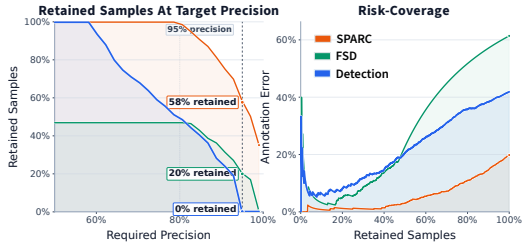

Fig. 2: Selective annotation. *Left*: retained coverage at increasing target precision. *Right*: risk-coverage curves. The reliability score retains more data at high precision and lowers risk at all levels.

only picks thresholds. An annotation is correct if its box matches ground truth at IoU > 0.4. We treat reliability as a *selective-prediction* problem: a threshold τ retains annotations with score $\geq \tau$, and we report accuracy, coverage at the 90% precision point (Cov@90), and risk-coverage area.

Results. Table I shows a practical failure mode for curating demonstrations in cluttered, high-ambiguity scenes: detector confidence and generic motion are insufficient quality signals. Detector confidence is only moderate, raw motion fails at high precision, and trajectory-length filtering gets a low E-AURC only through conservative abstention. Each interaction-aware cue improves accuracy and calibration (object motion gives the largest first gain, then the robot filter and phase-aware motion), reaching **80.2% accuracy**, **77.6% Cov@90**, and the lowest selective-prediction error (AURC 0.056, E-AURC 0.035). This exceeds even a large proprietary embodied model [30] (71.7%) while also producing a reliability score. The score is well calibrated for filtering (Fig. 2): at a 95% precision target SPARC retains 58% of samples versus 20% for the best trajectory-filtering baseline.

Does Risk-Aware Filtering Improve Embodied Spatial Reasoning? We test whether the reliability score yields better supervision. From SPARC annotations we build a 511k-sample VQA dataset (target-location, vacant-space pointing, trace prediction) and fine-tune Qwen3.5-4B, co-training with general instruction data to retain instruction following capabilities. As Table II shows, across pointing and grounding benchmarks (Where2Place [35], RefSpatial [39], RoboRefIt [22], VABench [36], ShareRobot [16]), SPARC supervision reaches a 69.6% pointing average, above the same model trained on human-annotated EO-1.5M [26] (62.6%) and far above the detection-annotated baseline (50.5%), without manually verified training data. It also improves trajectory prediction, while gains are smaller on language-only MCQA.

Training data	IA-Bench↑	Point. Avg↑	Traj. RMSE↓
Qwen3.5-4B (zero-shot)	49.7	41.9	n/a
+ Base mixture	45.2	48.2	0.268
+ Detection-annotated	57.7	50.5	0.218
+ EO-1.5M (human) [26]	68.9	62.6	n/a
+ SPARC (ours)	**79.1**	**69.6**	**0.192**

TABLE II: **Downstream VLM training.** Qwen3.5-4B fine-tuned on each data mixture, averaged over pointing/grounding benchmarks (Point. Avg) and object-trajectory benchmarks (Traj. RMSE). SPARC supervision surpasses both the human-annotated EO-1.5M and the detection baseline with no manual verification.

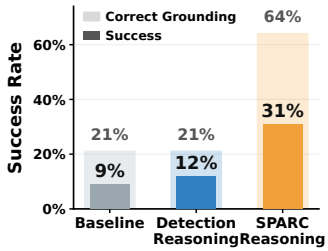

Fig. 3: Real-world policy success over 100 rollouts across 10 cluttered grounding tasks.

Do High-Quality Spatial Annotations Improve Robustness in Clutter? We build a cluttered tabletop setting where the robot must move visually similar objects to targets. Success depends on grounding rather than low-level control. We train three VLAs on 250 demonstrations across 10 tasks (shared backbone, flow-matching action head [34]): a next-action baseline and two reasoning policies that predict trajectories as language before acting, supervised by the detection baseline and by SPARC. Over 100 rollouts (Fig. 3), SPARC reasoning more than doubles the detection-annotated policy's success and triples the no-reasoning baseline under identical training.

IV. Conclusion

We introduce SPARC, an automatic spatial annotation pipeline that treats object grounding quality as a controllable curation problem. The main practical lesson is that detector confidence is a poor quality signal in cluttered, high-ambiguity scenes, because it scores recognition rather than interaction correctness. Interaction cues such as phase-aware motion, 3D gripper proximity, and robot overlap provide a more useful reliability signal, making the quality-scale trade-off controllable without human review. Models trained on SPARC annotations improve embodied grounding and real-world reasoning policies, suggesting that reliable annotation curation is an important step toward robust manipulation in production-like settings.

V. Limitations

SPARC is not evaluated in a real factory cell, and IA-Bench only captures production-like failure modes such as clutter, occlusion, repeated objects, and ambiguous grounding. The pipeline is also bounded by its off-the-shelf components: detectors limit part-level annotations, trackers can switch identity between similar objects, 3D lifting struggles with transparent or reflective surfaces, and the method assumes a mostly static camera with distinct grasp phases.

AI Usage Statement

AI assistants were used solely for language polishing and coding assistance. All technical content, experiments, and analysis are the authors' own.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 448648559. The authors gratefully acknowledge the computing time provided on the high- performance computer HoreKa by the National High- Performance Computing Center at KIT (NHR@KIT). This center is jointly supported by the Federal Ministry of Education and Research and the Ministry of Science, Research and the Arts of Baden-Württemberg, as part of the National High-Performance Computing (NHR) joint funding program (https://www.nhr-verein.de/en/our-partners). HoreKa is partly funded by the German Research Foundation (DFG). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for supporting this project by providing computing time on the GCS Supercomputer JUPITER at Jülich Supercomputing Centre (JSC). The authors gratefully acknowledge the support of the Robotics Institute Germany (RIG).

References

[1] Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning vla: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026.

[2] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[3] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[4] Kaiyuan Chen, Shuangyu Xie, Zehan Ma, Pannag R Sanketi, and Ken Goldberg. Robo2VLM: Improving visual question answering using large-scale robot manipulation data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=OChorZcZnY.

[5] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22831–22840, 2025.

[6] William Chen, Suneel Belkhale, Suvir Mirchandani, Karl Pertsch, Danny Driess, Oier Mees, and Sergey Levine. Training strategies for efficient embodied reasoning. In *Conference on Robot Learning*, pages 365–391. PMLR, 2025.

[7] Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, et al. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

[8] Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, et al. Rynnbrain: Open embodied foundation models. *arXiv preprint arXiv:2602.14979*, 2026.

[9] Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Heming Cui, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. In *9th Annual Conference on Robot Learning*.

[10] Yiyang Du, Zhanqiu Guo, Xin Ye, Liu Ren, and Chenyan Xiong. Embodiedmidtrain: Bridging the gap between vision-language models and vision-language-action models via mid-training, 2026. URL https://arxiv.org/abs/2604.20012.

[11] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL https://arxiv.org/abs/2605.02881.

[12] Shenghao Fu, Qize Yang, Qijie Mo, Junkai Yan, Xihan Wei, Jingke Meng, Xiaohua Xie, and Wei-Shi Zheng. Llmdet: Learning strong open-vocabulary object detectors under the supervision of large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14987–14997, 2025.

[13] Yulu Gan, Ligeng Zhu, Dandan Shan, Baifeng Shi, Hongxu Yin, Boris Ivanovic, Song Han, Trevor Darrell, Jitendra Malik, Marco Pavone, et al. Foundationmotion: Auto-labeling and reasoning about spatial movement in videos. *arXiv preprint arXiv:2512.10927*, 2025.

[14] Adam W Harley, Yang You, Xinglong Sun, Yang Zheng,

Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Suya You, et al. Alltracker: Efficient dense point tracking at high resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5253–5262, 2025.

[15] Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-thinkact: Efficient vision-language-action reasoning via verbalizable latent planning. *arXiv preprint arXiv:2601.09708*, 2026.

[16] Yuheng Ji et al. RoboBrain: A unified brain model for robotic manipulation from abstract to concrete. 2025.

[17] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *European Conference on Computer Vision (ECCV)*, 2024.

[18] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

[19] Hao Li, Ziqin Wang, Zi-han Ding, Shuai Yang, Yilun Chen, Yang Tian, Xiaolin Hu, Tai Wang, Dahua Lin, Feng Zhao, et al. Robointer: A holistic intermediate representation suite towards robotic manipulation. In *The Fourteenth International Conference on Learning Representations*.

[20] Wenqi Liang, Gan Sun, Yao He, Jiahua Dong, Suyan Dai, Ivan Laptev, Salman Khan, and Yang Cong. Pixelvla: Advancing pixel-level understanding in vision-language-action model, 2026. URL https://arxiv.org/abs/2511.01571.

[21] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[22] Yuhao Lu, Yixuan Fan, Beixing Deng, Fangfu Liu, Yali Li, and Shengjin Wang. Vl-grasp: a 6-dof interactive grasp policy for language-oriented objects in cluttered indoor scenes. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 976–983. IEEE, 2023.

[23] Gen Luo, Ganlin Yang, Ziyang Gong, Guanzhou Chen, Haonan Duan, Erfei Cui, Ronglei Tong, Zhi Hou, Tianyi Zhang, Zhe Chen, et al. Visual embodied brain: Let multimodal large language models see, think, and control in spaces. *arXiv preprint arXiv:2506.00123*, 2025.

[24] Haiyang Mei, Qiming Huang, Hai Ci, and Mike Zheng Shou. Robotseg: A model and dataset for segmenting robots in image and video. *arXiv preprint arXiv:2511.22950*, 2025.

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

[26] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[27] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pages 28085–28128, 2025.

[28] Arash Rocky and Q. M. Jonathan Wu. Sam2auto: Auto annotation using flash, 2025. URL https://arxiv.org/abs/2506.07850.

[29] Quanxin Shou, Fangqi Zhu, Shawn Chen, Puxin Yan, Zhengyang Yan, Yongping Miao, Xiaoyi Pang, Zicong Hong, Rui Shi, Haochen Huang, Jie Zhang, and Song Guo. Halo: A unified vision-language-action model for embodied multimodal chain-of-thought reasoning. *ArXiv*, abs/2602.21157, 2026. URL https://api.semanticscholar.org/CorpusID:286001130.

[30] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[31] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[32] Homer Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridge-Data V2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[33] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *Advances in Neural Information Processing Systems*, 38:35928–35959, 2026.

[34] Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026.

[35] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *8th Annual Conference on Robot Learning*.

[36] Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei

Ni, Longxin Kou, Jinyi Liu, Pengyi Li, Yan Zheng, and Jianye Hao. From seeing to doing: Bridging reasoning and decision for robotic manipulation, 2026. URL https://arxiv.org/abs/2505.08548.

[37] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, pages 3157–3181. PMLR, 2025.

[38] Qingqing Zhao et al. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[39] Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *Advances in Neural Information Processing Systems*, 38:28404–28481, 2026.

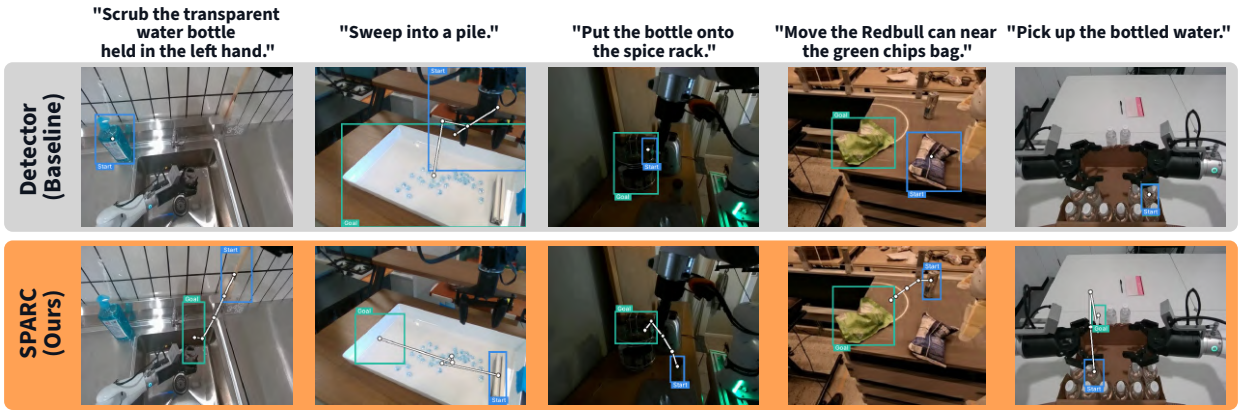

Fig. 4: Qualitative comparison on diverse robot demonstrations examples. Columns show different language-conditioned manipulation tasks, while rows compare annotations generated by the detector baseline and Spatial Annotations from Robot Demonstrations with Reliability Calibration (SPARC). Detector confidence often selects visually plausible distractors or robot parts, while SPARC selects the physically interacted object by combining phase-aware motion, 3D gripper proximity, and robot-overlap filtering.

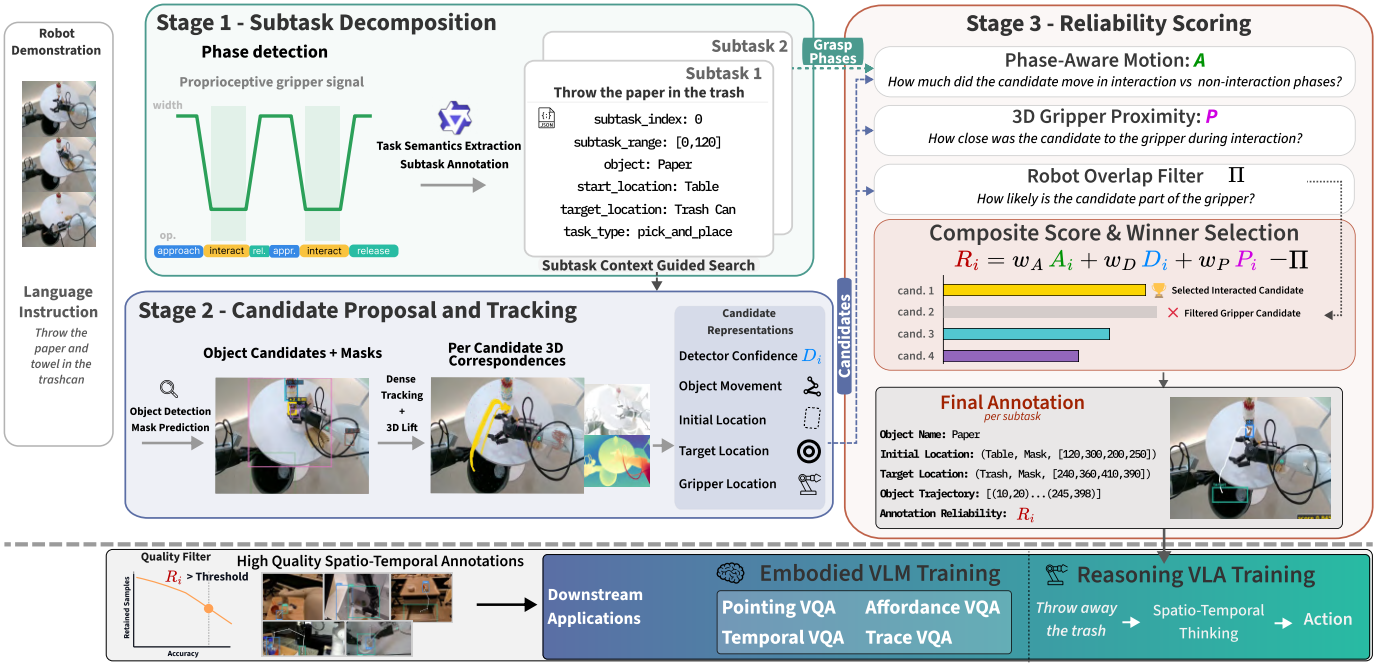

Fig. 5: **Overview of the SPARC annotation pipeline.** Stage 1 segments a demonstration into object-centric subtasks via gripper-phase detection and language parsing; Stage 2 proposes, tracks, and 3D-lifts object candidates; Stage 3 scores each candidate with phase-aware motion A, 3D gripper proximity P, and a robot-overlap filter, combined with detector confidence D into a composite reliability score R. The top-scoring candidate yields the annotation, and thresholding R controls the quality-coverage tradeoff without human review.

Table III shows existing works and how they generate and filter spatial annotations from robot demonstrations. Notably, almost all of them use a detection model as main confidence estimate, while some additionally use tracking as a signal. However, as discussed previously, the initial detector confidence is not calibrated well, thus relying on its confidence followed by track length filtering results in a significantly reduced coverage. On the other hand, SPARC combines spatio-temporal properties of robot object manipulation to obtain a robust and well calibrated filtering signal.

The overall pipeline is depicted in Fig. 5.

Type	Method / dataset	Ann.	Filtering signal			Labels
			Det.	Track	Robot	
Human	RoboInter [19] / RynnBrain [8]	H	✗	✗	✗	Box / VQA / contact
	EO-1M [26]	H+A	✗	✗	✗	Box / VQA / contact
	AgiBot-World [2] / VeBrain [23]	H	✗	✗	✗	Task / scene / object
	Gemini Robotics [30]	?	?	?	?	Trace / Mask
Sim.	RoboPoint [35] / InterVLA-A1 [7]	A	✓	✗	✗	Box / CoT
	RoboRefer [39] / GraspVLA [9]	A	✓	✗	✗	Box / CoT
Automatic filtering	LaRA-VLA [1] / ECoT [37, 6]	A	✓	✗	✗	Box / CoT
	Fast-Think-Act / Molmo-Act [15]	A	✓	✗	✗	Box / CoT
	PixelVLA [20]	A	✓	✗	✗	Box / mask
	SAM2Auto [28] / FoundationMotion [13] /FSD [36]	A	✓	✓	✗	Track / motion
	HALO [29]	A	✗	✗	✗	CoT / affordance
	Robo2VLM [4]	A	✗	✗	✓	Grip. phases
	Ours	**A**	✓	✓	✓	**Box / Mask / Grip / Trace**

TABLE III: Comparison of annotation pipelines by annotation source and filtering signal. **Type** groups methods by annotations type: *Human* (manually labelled real-world data), *Sim.* Simulation-based, and *Automatic filtering*. **Ann.**: H = human, A = automatic. Det. denotes detector or mask confidence. Ours is the only pipeline combining all three filtering signals with the richest label set.

A. SPARC Implementation Details

1) Stage 1 - Subtask Decomposition and Phase Detection: **Interaction segmentation.** Given a long-horizon demonstration $\tau = \{(x_t, q_t)\}_{t=1}^{T}$, with RGB frames x_t, proprioceptive state q_t, and language instruction ℓ, SPARC decomposes τ into a sequence of short object-centric subtasks. Each subtask is intended to isolate one coherent interaction with a single manipulated entity, so that later proposal, tracking, and scoring operate on temporally localized clips rather than the full demonstration. For tool-use behaviors, the manipulated tool is treated as the primary interaction object, while the acted-on receptacle or surface is retained as the target context. The output of this stage is a structured subtask set

$$\mathcal{S}(\tau) = \left\{ \left(u_j, \ell_j, q_j^{\text{init}}, q_j^{\text{target}}, \mathcal{P}_j \right) \right\}_{j=1}^{M},$$

where u_j is the subtask instruction, ℓ_j the manipulated object name, q_j^{init} and q_j^{target} textual descriptions of the initial and target locations, and $\mathcal{P}_j$ the associated grasp-phase sequence.

Gripper phase extraction. SPARC derives the temporal support of each interaction from the normalized gripper signal $a_t \in [0, 1]$, where smaller values correspond to a more closed gripper. The implementation uses a *dual-pass* procedure with hysteresis to robustly separate true grasps from teleoperation noise, short slips, and transient re-openings. We first normalize the raw gripper trace per trajectory and define asymmetric closing and opening thresholds

$$\tau_{\text{enter}} = \tau_c - \delta, \qquad \tau_{\text{exit}} = \tau_c + \delta,$$

so the state enters a closed interaction only when $a_t < \tau_{\text{enter}}$ and exits it only when $a_t \geq \tau_{\text{exit}}$. This hysteresis prevents oscillatory threshold crossings from fragmenting a single interaction into many short phases. If the gripper signal shows almost no range, the trajectory is treated as a single coarse interaction span.

In the *first pass*, SPARC scans the trajectory with a finite-state machine over `grasp`, `interact`, and `release`. A transition is only accepted after several consecutive frames satisfy the new state condition, which suppresses one-frame fluctuations. This pass does *not* yet decide whether a closure is valid or failed; instead, it simply records candidate closed-gripper spans and their durations

$$d_k = e_k - s_k + 1,$$

where $[s_k, e_k]$ is the k-th candidate interaction interval. From these provisional spans, the algorithm estimates a robust per-trajectory interaction scale. Concretely, it computes the median candidate duration and combines it with a per-attempt interaction budget derived from the trajectory length and the number of detected attempts, yielding a minimum valid interaction duration

$$d_{\min} = \max\left(0.5\,\frac{B}{K},\ 0.4\,\text{median}(\{d_k\}_{k=1}^{K}) \right),$$

where B is the expected total interaction budget for the trajectory and K is the number of candidate closures. This makes the threshold adaptive: in short or repetitive demonstrations, valid short grasps are preserved, while unusually brief closures remain identifiable as outliers.

In the *second pass*, the same state machine is run again, now using $d_{\min}$ to classify each closed interval. Closures whose interact duration exceeds $d_{\min}$ are emitted as standard `grasp` $\rightarrow$ `interact` $\rightarrow$ `release` cycles, whereas shorter closures are relabeled as `grasp_failure` and do not create full interaction segments. The implementation additionally repairs boundary artifacts in two ways. First, if a new grasp begins too soon after a release, a small number of frames may be reassigned from the preceding release to the new grasp so that the new cycle retains a plausible approach phase. Second, release phases are allowed to terminate early when the gripper re-closes quickly, ensuring back-to-back pick attempts are not merged. After phase extraction, SPARC applies a small forward offset to each phase boundary, borrowing a fraction of the following phase, so that downstream keyframes better capture contact transitions and early object motion. For bimanual trajectories, this entire dual-pass procedure is applied independently to each arm before the two phase streams are merged temporally.

Language-grounded subtask parsing. Gripper phases indicate when interaction occurs, but not which object is involved or how the interaction should be described. SPARC therefore conditions a language model on the original instruction ℓ together with the ordered phase list and asks it to recover a structured description of the executed subtasks. For each subtask, the model returns the manipulated object name, action type, start and target locations, and a phase-level natural-language description aligned with the detected phase sequence. This language grounding step is cached by instruction together with the detected phase-sequence key, so repeated trajectories with the same language-phase pattern do not trigger repeated model queries. In bimanual clips, the prompt is arm-aware and predicts arm-specific object assignments before the two streams are merged back into a unified subtask list.

Subtask boundary alignment. After parsing, the symbolic phase descriptions are aligned back to the measured trajectory timeline. Concretely, each predicted phase entry inherits its start and end frames from the ordered detected phase list, producing a temporally grounded subtask clip for downstream annotation. SPARC then expands each subtask clip slightly beyond the raw closed-gripper interval using the offset phases, so proposal and tracking can observe approach motion before contact and release motion after placement. The result is a set of short-horizon interaction records that jointly specify *what* object to look for, *when* to look for it, and *where* it is expected to begin and end.

2) Stage 2 - Candidate Proposal, Segmentation, and Tracking: **Object proposal at interaction keyframes.** For each subtask j, SPARC extracts the corresponding video clip and selects a grasp-conditioned detection keyframe near the onset of object interaction. The manipulated object is then proposed with LLMDet [12] using a low confidence threshold, so the detector acts primarily as a broad region proposal mechanism rather than a strict final classifier. To calculate the grounding scores, we average the token logits for each box. In parallel, robot-arm and gripper detections are collected at the same keyframes, which later support candidate filtering, robot-aware scoring, and failure recovery. This proposal stage intentionally favors recall: it is preferable to include several plausible object boxes and defer disambiguation to later tracking and scoring.

Target proposal. The final object location is not inferred from motion alone. Instead, SPARC also proposes target boxes using the target-location phrase returned in Stage 1. The implementation queries target candidates from early and late frames of the subtask clip, then merges and suppresses them with size filtering and non-maximum suppression. For pick-and-place-like behaviors, this gives a set of plausible end locations against which tracked object motion can be matched. If the language model did not predict an explicit target location, the pipeline falls back to the object or tool name as a weak target descriptor and detects on the last frame rather than discarding the subtask.

Segmentation and candidate refinement. Each detected object proposal is refined into an instance mask with SAM2 [27], yielding a set of candidates

$$c_i = (b_i, m_i, \ell_i, D_i)\,,$$

where b_i is the initial box, m_i the segmentation mask, ℓ_i the text-grounded label, and D_i the detector confidence. Mask refinement is important because later tracking operates on object-support regions rather than on coarse boxes alone. For DROID-like data, the implementation may crop the clip around the union of detected object and target boxes before tracking, which reduces wasted image area and improves tracking stability in wide views; all coordinates are restored to the original image frame before the final annotation is written.

Temporal tracking and 3D lifting. Given the segmented candidates, SPARC propagates object evidence over time and ranks candidates by whether their trajectories match the intended interaction. The current implementation supports several tracking variants: a point-based tracker, a SAM2-video propagation mode, and a hybrid mode in which an initial point-tracker ranking is re-evaluated with SAM2 on the top candidates. At a high level, all variants produce temporally extended candidate representations of the form

$$o_i = \left(c_i, \Gamma_i^{\mathrm{2D}}, \Gamma_i^{\mathrm{3D}}\right),$$

where Γ_i^{2D} denotes the candidate's propagated image-space tracks or masks across the subtask clip, and Γ_i^{3D} denotes the corresponding lifted 3D tracks when geometry is available. The 3D lifting step is optional in the sense that it is only applied when dense geometry prediction succeeds, but when available it provides embodiment-aware evidence for later scoring. Intermediate detections at additional keyframes are also retained, allowing the tracker to prefer candidates whose propagated support remains compatible with re-detections throughout the clip rather than only at the start and end.

3) Stage 3 - Reliability Scoring and Final Annotation Selection: Stage 2 yields a set of tracked object candidates $\{o_i\}_{i=1}^{N}$, where each candidate consists of an initial detection, a segmentation mask, and a set of tracked object points over time. Stage 3 ranks these candidates by combining appearance reliability, phase-aware motion, 3D gripper proximity, and a robot-overlap penalty. The score is designed so that the selected object is not merely visually plausible, but also exhibits the temporal and spatial signatures expected of a manipulated object.

Per-candidate tracked support. For each candidate o_i, tracking starts from points sampled inside the first-frame object mask m_i. Let $\Gamma_i = \{\mathbf{u}_{i,p}^t\}$ denote the resulting 2D tracks, where p indexes sampled points and t indexes frames in the subtask clip. Each point also has a visibility indicator $v_{i,p}^t \in \{0, 1\}$. Only points that remain visible for a sufficient fraction of the clip are retained for scoring, which suppresses unstable or spuriously short tracks. All motion, target, and proximity signals are then computed from this filtered point set.

Detector confidence. Each candidate o_i inherits a language-grounded detector confidence D_i from the initial proposal stage. This term captures appearance-level compatibility between the visual region and the queried object description. On its own, D_i is often insufficient for selecting the manipulated object, since visually salient distractors or robot parts can also receive high scores. It is therefore used as an appearance prior that is later reweighted by geometric interaction evidence rather than as a standalone selection rule.

Phase-aware motion signal. The first signal measures whether a candidate moves primarily during the interaction phase rather than throughout the full clip. For each visible point, we compute frame-to-frame displacement and then aggregate it at the candidate level. Concretely, for candidate i, the instantaneous 2D motion at frame transition $t \to t+1$ is

$$m_{i,t} = \text{median}_{p:\, v_{i,p}^t = v_{i,p}^{t+1} = 1} \left\| \mathbf{u}_{i,p}^{t+1} - \mathbf{u}_{i,p}^t \right\|_2 \cdot \text{fps},$$

that is, the median displacement of all visible tracked points, converted to pixels per second. A short temporal smoothing window is then applied to reduce jitter. Let $\mathcal{T}_{\text{int}}$ denote the set of interaction-phase frame transitions and $\mathcal{T}_{\text{non}}$ the remaining valid transitions. Their average motion is

$$\mu_i^{\text{int}} = \frac{1}{|\mathcal{T}_{\text{int}}|} \sum_{t \in \mathcal{T}_{\text{int}}} m_{i,t}, \qquad \mu_i^{\text{non}} = \frac{1}{|\mathcal{T}_{\text{non}}|} \sum_{t \in \mathcal{T}_{\text{non}}} m_{i,t}.$$

The raw phase-aware motion score is then computed as a soft signal-to-noise ratio,

$$A_i = \frac{(\mu_i^{\text{int}})^{0.6}}{(\mu_i^{\text{non}} + 1.0)^{0.2}}.$$

This favors candidates that move persistently when the gripper is engaged, but penalizes candidates whose motion is equally strong before grasp or after release, which is typical of robot parts, background clutter, or unstable tracks. The set $\{A_i\}$ is min-max normalized across candidates in the same clip to obtain $\widehat{A}_i$.

3D gripper-proximity signal. The second signal measures whether the candidate occupies the local 3D neighborhood of the active gripper during manipulation. When dense geometry is available, each tracked 2D point is lifted to 3D, giving per-candidate 3D tracks $\mathbf{x}_{i,p}^t \in \mathbb{R}^3$. A gripper reference point is then estimated for each frame from the tracked gripper points. Rather than averaging all gripper points, which is unstable under partial visibility and self-occlusion, the reference is chosen as the gripper point whose 2D projection lies closest to the image center among the visible gripper points in that frame. This choice produces a stable central proxy for the fingertip/TCP region. We found that selecting a consistent track is unreliable becuase trackers struggle significantly with tracking a point on the gripper consistently, likely due to visual similarity accross the gripper.

Around this reference point g_t, an adaptive 3D sphere $B(g_t, r_t)$ is constructed. Its radius is determined from the local gripper geometry together with an estimate of object scale, so the same criterion remains meaningful across small and large objects. $Q_{i,t}$ is the set of visible 3D points of candidate i at frame t. The per-frame proximity fraction is

$$p_{i,t} = \frac{1}{|Q_{i,t}|} \sum_{\mathbf{x} \in Q_{i,t}} \mathbf{1}[\mathbf{x} \in B(g_t, r_t)]\,.$$

Averaging over valid frames yields the 3D proximity score

$$P_i = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} p_{i,t}.$$

In the final implementation, the mean point-fraction variant is used, and the sphere radius is slightly enlarged relative to the raw gripper scale so that nearby in-contact objects are not undercounted. The resulting $\{P_i\}$ are min-max normalized across candidates to obtain $\widehat{P}_i$.

Robot-overlap penalty. A major failure mode is selecting the gripper or robot arm itself. This is addressed with a continuous penalty rather than a hard rejection rule. For bimanual data, the RobotSeg mask is restricted to the active arm component before scoring, which prevents the inactive arm from suppressing candidates on the correct side of the workspace. Active arms are determined by activity in the proprio gripper signal.

At each aligned keyframe, the candidate's visible tracked points are rasterized against the RobotSeg mask, producing a per-frame overlap fraction. These per-frame values are then robustly aggregated, emphasizing the strongest overlap frames rather than averaging uniformly over the entire clip. This yields a candidate-level overlap score $O_i \in [0, 1]$. High overlap should decrease the final score, but not all overlap is equally harmful: true objects frequently touch or partially occlude the gripper. To distinguish contact from identity confusion, the penalty is modulated by a 3D depth-relief term. Specifically, the median absolute depth difference between the gripper tip region and the candidate tip region is measured across valid frames. Large depth separation weakens the overlap penalty, while near-zero separation preserves it.

The final robot penalty Π_i is therefore a continuous quadratic function of RobotSeg overlap, attenuated by this depth-relief factor. This is crucial in practice: earlier hard overlap filtering removed many correct contact-rich candidates, whereas the continuous penalty suppresses obvious robot regions without catastrophically rejecting true manipulated objects.

Composite score. The final reliability score follows the notation used in the main paper. We min-max normalize A_i and P_i across candidates in the same clip, giving $\widehat{A}_i$ and $\widehat{P}_i$, and combine all four signals:

$$R_i = w_A \, \widehat{A}_i + w_D(\widehat{P}_i) \, D_i + w_P \, \widehat{P}_i - \Pi_i,$$

where $w_A = 0.5$, $w_P = 0.3$, and

$$w_D(\widehat{P}_i) = 0.75 - 0.15 \, \widehat{P}_i.$$

Thus, detector confidence D_i contributes strongly when 3D proximity evidence is weak, but its influence is reduced once the candidate is already well supported by embodiment-aware context. The selected manipulated object is

$$i^\star = \arg\max_i R_i.$$

Final target selection and target position estimation. The selected manipulated object is $i^\star = \arg\max_i R_i$. After selecting $i^\star$, SPARC resolves the target position from a set of detected target candidates $\mathcal{C}^{\text{tar}} = \{c_j^{\text{tar}}\}_{j=1}^{M}$, obtained from language-conditioned target detection on early and late frames of the subtask clip. In the main pipeline, these candidates are scored jointly with the selected object hypothesis using motion, temporal alignment, and target agreement, so target resolution remains tied to the same interaction evidence used for manipulated-object selection.

Concretely, SPARC measures how many tracked points from the selected object candidate terminate inside each target candidate and prefers targets that receive a dense concentration of transported points. In downstream target remapping, this support is normalized by target-box size to avoid favoring large diffuse boxes. Let $S_{i^\star j}^{\text{ungated}}$ denote the fraction of visible final tracked points from candidate $i^\star$ that lie inside target candidate c_j^{tar}, and let A_j denote its box area. The density-normalized target score is

$$\widetilde{S}_{i^\star j} = \frac{S_{i^\star j}^{\text{ungated}}}{\sqrt{A_j / A_{\max}}}, \qquad A_{\max} = \max_j A_j.$$

This favors compact target boxes that capture a high density of transported points. To avoid degenerate tiny boxes, candidates whose area is less than half the selected start-box area are excluded before ranking.

If tracking-supported target resolution is unavailable, SPARC falls back to the best detected target candidate; for tool-use tasks, it further falls back to the acted-on object or tool context when no separate target box is available. Note that since the best interacted object is already fixed, which includes movement scoring, this approach does not degenerate to static points, which would dominate this selection logic.

B. Runtime

SPARC runs several large foundation models in sequence for each trajectory, resulting in roughly 5 seconds per annotation on a single NVIDIA GH200 GPU. To reduce this further, we subsample tracking frames to 6 fps. We keep the native resolution, since lowering it degraded annotation quality, especially for small objects. For reference, human annotators labeled the 1,275 trajectories in the Interaction-Aware Bench (IA-Bench) test set in roughly 12 hours, corresponding to approximately 34 seconds per annotation. Crucially, SPARC is highly parallelizable, as trajectories can be processed independently. To maximize GPU utilization, each GPU hosts a dedicated inference server that loads all models once at startup and exposes them to multiple CPU workers through an asynchronous request queue. Each CPU worker processes a separate trajectory and issues inference requests concurrently, keeping the GPU continuously occupied across detection, segmentation, and tracking. On a single node with four GH200 GPUs and four CPU workers per GPU, 16 workers in total, annotating the same 1,275 trajectories takes approximately 30 minutes. This yields a $24\times$ wall-clock speedup over human annotation, and the workload distributes across

multiple nodes without modification. Furthermore, all annotations generated by our pipeline can be used downstream for several applications.

C. Foundation Model Choices

SPARC relies on off-the-shelf foundation models to generate interaction-aware signals used for annotation and reliability scoring. We will explain the choice for each individual model in this section. Note that SPARC allows changing the individual models as better models for embodied settings emerge, and the method is not tied to a specific set of models. Table IV lists all the foundation models used for generating the different candidate representations. For task decomposition in stage 1 SPARC leverages Qwen3.6 due to fast inference speed and good out-of-the-box embodied and physical reasoning capabilities. For intial candidate proposal with open-vocabulary object detection, we rely on LLMDet. Although OWLv2 is slightly better in grounding objects in robotic settings, we found that its grounding score is generally less calibrated compared to LLMDet. Similar behavior was observed with GroundingDino [21]. Furthermore, LLMDet produces less noisy boxes at very low thresholds. Since SPARC uses the initial detector mainly as a region proposal network, the proposal network has to be robust at low threshold regimes, and should actually output boxes around objects and not empty space.

Component	**Model**
Task parsing	Qwen3.6-30B-MOE [31]
Object detection	LLMDet-Base [12]
Gripper detection	RobotSeg [24]
Object tracking	AllTracker [14]
Segmentation	SAM2.1 Hiera-Small [27]
Depth/Geometry Estimation	MoGe-2[33]

TABLE IV: Overview of Foundation Models used in SPARC

For gripper detection, we use RobotSeg, a model that builds on SAM2 and is trained specifically to segment gripper masks in video, producing far more reliable gripper masks than a standard pipeline that follows object detection and segmentation. Existing object detectors and segmentation models struggle to consistently localize and segment the same parts of the gripper.

For object tracking, we rely on a dense tracker instead of sparse trackers such as [17]. We found that the tracks are more robust than those from sparse tracking. Furthermore, since we track many possible candidate objects, CoTracker runtime was significantly larger than AllTracker, which despite tracking all pixels in the scene has proven to be very efficient.

SPARC uses SAM2.1 for segmentation instead of the newer SAM3 [3] since SAM2.1 is significantly faster than SAM3. Furthermore, SPARC does not require the advantages coming with SAM3, such as grounding or better masks.

Finally, we rely on a monocular depth prediction model MoGe-2 for 3D lifting. MoGe-2 predicts scene geometry individually per frame from a single RGB observation and outputs a depth map, normals, and point cloud without requiring camera intrinsics. If intrinsics are available, MoGe-2 can use those to improve geometry reconstruction. Because MoGe does not incorporate temporal information, the resulting depth maps differ slightly in scale across frames but remain consistent within each frame. Since the signals used by SPARC operate on a per-frame basis, we do not require consistent depth maps on a temporal axis. While there are methods specifically tailored at predicting consistent depth across time [5], they often require known intrinsics or have significantly higher runtime. Furthermore, MoGe operates at a higher native resolution, which is important for accurately capturing gripper-object boundaries necessary for robust gripper overlap calculation.

D. Pipeline Hyperparameters and Thresholds

In Table V, we show the hyperparameters used in SPARC to perform all evaluations. Note that we did tune some of these hyperparameters on the hold-out validation set, but never on the test set. Furthermore, we did not tune the hyperparameters on the OXE dataset. These parameters have proven robust when transferring to the test set and new embodiments.

Name	Symbol	Value
Stage 1 – Gripper phase detection		
Closure threshold	τ_c	0.6
Hysteresis offset	δ	0.05
Min. phase duration	$T_{\min}$	7 frames
Phase start offset	r_{off}	0.3
Stage 2 – Detection keyframe		
Keyframe position	t^*	midpoint of subtask start and first grasp end
Detector Threshold	–	0.05
Stage 3 – Phase-aware motion signal		
Interact exponent	α	0.6
Non-interact exponent	β	0.2
Noise floor	ε	1.0 px/s
Motion term weight	w_A	0.5
Stage 3 – Adaptive detector and sphere proximity		
Detector base weight	w_D^0	0.75
Adaptive detector slope	w_D^{slope}	0.15
Sphere proximity weight	w_P	0.3
Sphere radius scale	k_r	8
Stage 3 – Robot-overlap soft gate		
Overlap onset	τ_{ov}	0.3
Penalty scale	λ	1.45
Depth attenuation	γ	0.85
Full-overlap spike coeff.	λ_{sp}	0.20
Full-overlap spike thresh.	τ_{sp}	0.98
Depth relief lower bound	z_ℓ	0.05 m
Depth relief upper bound	z_h	0.25 m
Evaluation		
IoU match threshold	τ_{IoU}	0.4
Containment fallback IoU	τ_{ct}	0.1
Containment fraction	f_{ct}	0.8

TABLE V: SPARC pipeline hyperparameters. All values are fixed across datasets and embodiments.

E. Reliability Score Sensitivity Analysis

Parameter	Value	Acc ↑	Cov@90 ↑	AURC ↓
w_A (motion weight)	0.3	0.794	0.760	0.0613
	0.4	0.803	0.770	0.0568
	0.5	**0.802**	**0.776**	**0.0563**
	0.6	0.799	0.768	0.0569
	0.7	0.795	0.765	0.0571
w_P (sphere weight)	0.1	0.798	0.765	0.0589
	0.2	0.803	0.778	0.0567
	0.3	**0.802**	**0.776**	**0.0563**
	0.4	0.803	0.759	0.0570
	0.5	0.801	0.748	0.0583
w_D slope	0.0	0.803	0.776	0.0569
	0.1	0.807	0.775	0.0553
	0.15	**0.802**	**0.776**	**0.0563**
	0.2	0.801	0.771	0.0564
τ_c (overlap onset)	0.2	0.803	0.775	0.0559
	0.3	**0.802**	**0.776**	**0.0563**
	0.4	0.802	0.768	0.0565
α/β (SNR exp.)	0.4/0.1	0.802	0.764	0.0578
	0.6/0.2	**0.802**	**0.776**	**0.0563**
	1.0/0.5	0.799	0.773	0.0562
	2.0/1.0	0.789	0.757	0.0585

TABLE VI: Reliability-score sensitivity analysis. Each block varies one hyperparameter while holding all others at their default (**bold**). Results on the IA-Bench test split; all values are fixed across datasets and embodiments.

F. IoU-threshold Robustness

G. Robot-overlap Filter False-Suppression Analysis

H. Per Dataset Results

Table IX reports the full ablation across all four IA-Bench datasets. FSD abstentions are counted as incorrect throughout. Agibot proves to be the most challenging dataset, compared to Bridge, which consists mostly of simple pick-and-place in visually simple environments.

I. Per Task Results

We further show the performance of SPARC on different task groups.

Method	IoU >0.4	IoU >0.5	IoU >0.75
Det. conf. only	0.581	0.572	0.564
+Mean motion	0.697	0.689	0.681
+Robot filter (hard)	0.727	0.716	0.706
+Soft-SNR motion	0.778	0.766	0.759
+Adaptive det. (sphere)	0.785	0.776	0.768
SPARC (ours)	**0.802**	**0.793**	**0.783**

TABLE VII: Accuracy at stricter IoU match thresholds. Correctness at each threshold: IoU $>$ threshold, *or* predicted box $\geq$80% contained within GT box with IoU >0.1 (containment fallback identical across columns). SPARC retains its margin at all thresholds.

Statistic	Count	% of solvable
Solvable demonstrations	1171	100.0
GT candidate hard-suppressed (≥ 0.3 overlap)	72	6.1
, of which false suppressions	34	2.9
True suppressions (wrong $\rightarrow$ correct)	60	5.1
Acc without filter (+Mean motion)	0.697	
Acc with hard filter	0.727	
Acc SPARC (quadratic penalty, ours)	**0.802**	

TABLE VIII: Robot-overlap filter false-suppression analysis on the IA-Bench test split. Correctness uses the IoU+containment criterion throughout. "GT hard-suppressed" counts demonstrations where the correct candidate received overlap ≥ 0.3 and was gated to $-\infty$ in the ablation row *+Robot filter (hard)*. "False suppression" is the subset where *+Mean motion* (no filter) would have selected the correct candidate. SPARC's quadratic penalty avoids hard gating and lifts accuracy from 0.727 to 0.802.

Task groups are derived from the action label produced by the LLM subtask parser (Sec. II): we match the action string against keyword lists for each group (e.g. *pick*, *place*, *lift* $\rightarrow$ Relocate; *open*, *close* $\rightarrow$ Open/Close) and assign unmatched actions to *Other*.

SPARC achieves its strongest results on the two categories that have the clearest motion signal: *relocate* (0.861) and *fluid* (0.857). Both involve sustained, large-amplitude end-effector displacement that the SNR-based motion score is well-suited to capture. *Tool/shape* tasks (0.896) benefit additionally from the dedicated tool-detection branch, which explicitly segments the held instrument and transfers its tracks to the manipulated object. Performance is weakest on *press/switch* (0.385) and *open/close* (0.581), tasks characterized by small spatial displacement or purely rotational motion; here the motion signal carries less discriminative power, and the sphere-shape prior is rarely applicable. *Articulate* tasks show a modest regression when the sphere bonus is added, suggesting that the spherical-object heuristic occasionally fires on non-articualted articulated objects such as drawer handles. Across every group, SPARC substantially outperforms FSD. FSD's coverage-adjusted accuracy collapses to 0.000 on press/switch because nearly all of those objects are small and briefly tracked, causing the area-and-duration filter to abstain on the entire group. This confirms that a purely detector-driven, coverage-agnostic baseline cannot generalise across the diversity of manipulation task types present in IA-Bench.

J. Hyperparameters

Table XI shows the hyperparameters used to train all VLM models on the respective data mixes.

K. VQA Dataset Generation

To generate the VQA dataset, we use templated question-answer pairs. We consider three task types. In object grounding, the VLM points to a specified object from either its name or the demonstration instruction. In vacant location pointing, the VLM points to the start or target location of the demonstration. In trajectory prediction, the VLM predicts the object trajectory for a given task instruction. We use the same templates and prompts for all variants, and only replace the underlying spatial annotations with either SPARC or detection-generated annotations.

For target point supervision, we sample a point from the winner candidate mask. For vacant pointing, we use the extracted start or target location and query the first or last observation frame in which the object is absent at that location. We then sample points at the corresponding vacant location. This provides a proxy for pointing to a vacant location without requiring explicit location annotations, thereby allowing supervision from the language instruction alone. For trajectory tasks, we subsample five equally spaced points from the object trajectory. All spatial coordinates are normalized to the range $[0, 1000]$.

L. Downstream Dataset Scaling Experiments

We study how annotation selection quality affects downstream scaling. For each scoring rule, we rank automatically generated annotations and train models on the top-scoring 50K, 200K, 500K, and 838,211 samples. We do not mix other datasets with

Dataset	Scoring rule	Acc.↑	Cov@90↑	Cov@95↑	AURC↓	E-AURC↓
Bridge (N=311)	FSD [36]	0.598	0.624	0.460	0.113	0.018
	Det. confidence	0.698	0.486	0.347	0.123	0.071
	+ Mean movement	0.826	0.711	0.450	0.060	0.044
	+ Robot filter	0.871	0.894	0.643	0.041	0.032
	+ Phase aware movement	0.907	1.000	0.894	0.021	0.016
	Ours (Final)	**0.913**	**1.000**	**0.904**	**0.020**	**0.016**
AgiBotWorld (N=258)	FSD [36]	0.357	0.225	0.116	0.330	0.053
	Det. confidence	0.450	0.039	0.008	0.376	0.184
	+ Mean movement	0.597	0.271	0.167	0.222	0.127
	+ Robot filter	0.578	0.275	0.159	0.223	0.117
	+ Phase aware movement	0.655	0.360	0.236	0.153	0.084
	Ours (Final)	**0.674**	**0.372**	**0.244**	**0.145**	**0.085**
DROID (N=263)	FSD [36]	0.456	0.437	0.335	0.225	0.038
	Det. confidence	0.563	0.000	0.000	0.267	0.152
	+ Mean movement	0.646	0.498	0.338	0.131	0.059
	+ Robot filter	0.692	0.494	0.338	0.121	0.067
	+ Phase aware movement	0.741	0.654	0.570	0.077	0.039
	Ours (Final)	**0.741**	**0.654**	**0.570**	**0.077**	**0.040**
OXE (N=424)	FSD [36]	0.229	0.250	0.208	0.439	0.004
	Det. confidence	0.587	0.380	0.302	0.192	0.091
	+ Mean movement	0.696	0.557	0.486	0.110	0.058
	+ Robot filter	0.733	0.623	0.524	0.083	0.044
	+ Phase aware movement	0.802	0.828	0.757	0.043	0.022
	Ours (Final)	**0.835**	**0.863**	**0.762**	**0.039**	**0.024**

TABLE IX: **Per-dataset ablation on IA-Bench.** Each ablation stage is applied cumulatively from detection confidence alone up to the full SPARC score. FSD abstentions (53% of samples on OXE) are counted as incorrect. Cov@90/95 = coverage at 90%/95% precision operating point.

TABLE X: Accuracy broken down by task group across all datasets (IA-Bench test split, n=1,256). Correctness uses the IoU+containment criterion. FSD abstains on 53% of demonstrations; abstentions are counted as incorrect, consistent with the requirement of full-corpus annotation coverage.

		Method					
Task group	n	Det. conf.	+Motion	+Filter	+Soft-SNR	SPARC	FSD
Relocate	884	0.629	0.761	0.776	0.828	**0.861**	0.436
Open / Close	117	0.368	0.530	0.590	0.615	**0.581**	0.222
Articulate	47	0.574	0.532	0.596	**0.681**	0.638	0.340
Press / Switch	13	0.308	0.154	0.231	0.385	**0.385**	0.000
Fluid	49	0.388	0.592	0.735	0.776	**0.857**	0.306
Tool / Shape	48	0.854	0.896	0.896	**0.917**	0.896	0.625
Other	98	0.408	0.429	0.490	0.551	**0.592**	0.235
All	**1256**	0.581	0.697	0.727	0.778	**0.802**	0.394

the respective data subsets. SPARC ranks samples using the proposed interaction-aware reliability score, while the baseline ranks samples by detector confidence. We additionally report a quality-filtered reference for each method using the same fixed quality threshold as in the main experiments.

SPARC provides a stronger scaling signal than detector confidence on both the overall benchmark suite and IA-Bench. On the aggregate score, SPARC improves from 0.554 at 50K to 0.583 at 200K, then decreases to 0.563 at 500K and 0.557 at 838K. This suggests that scaling is beneficial until lower-ranked annotations enter a noisier data regime. Note that the drop in performance on other general benchmarks can also be attributed to the model overfitting on our domain when not mixing in general VQA data. However, on IA-Bench, SPARC continues to improve with scale, rising from 0.716 at 50K to 0.785 at 838K, close to the quality-filtered reference of 0.789. In contrast, detector confidence degrades as more samples are added in both settings. This indicates that detector confidence increasingly admits noisy supervision, while SPARC preserves useful supervision over a much larger annotation budget.

Hyperparameter	Ours + Mix + Threshold	Det + Mix + Threshold	Mix (FSD+RoboPoint+OV2)	EO-1.5M + Mix
Model	Qwen3.5-4B	Qwen3.5-4B	Qwen3.5-4B	Qwen3.5-4B
Per-device batch size	8	8	8	8
Gradient accumulation	1	1	1	1
Effective global batch size	128	128	128	128
Epochs	1	1	1	1
Learning rate	2×10^{-5}	2×10^{-5}	2×10^{-5}	2×10^{-5}
LR scheduler	cosine w/ min LR	cosine w/ min LR	cosine w/ min LR	cosine w/ min LR
Warmup ratio	0.03	0.03	0.03	0.03
Min LR ratio	0.1	0.1	0.1	0.1
Weight decay	0.0	0.0	0.0	0.0
Max grad norm	1.0	1.0	1.0	1.0
Precision	bf16	bf16	bf16	bf16
Gradient checkpointing	true	true	true	true
Vision encoder frozen	true	true	true	true
Vision projector frozen	false	false	false	false
Max sequence length	5600	5600	5600	5600
Candidate selector	SPARC	Detection Confidence	–	–
Max samples	all	all	all	all
Quality threshold	0.97	0.85	–	–
Max per object	700	700	–	–
Data mix	Ours + fsd + robopoint + llavaov2	Ours + fsd + robopoint + llavaov2	fsd + robopoint + llavaov2	EO-1.5M + fsd + robopoint + llavaov2
Dataset Size	1,159,047	941,642	793,580	1,063,630

TABLE XI: Training hyperparameters for the main mixture-based models used in our downstream VLM training experiments.

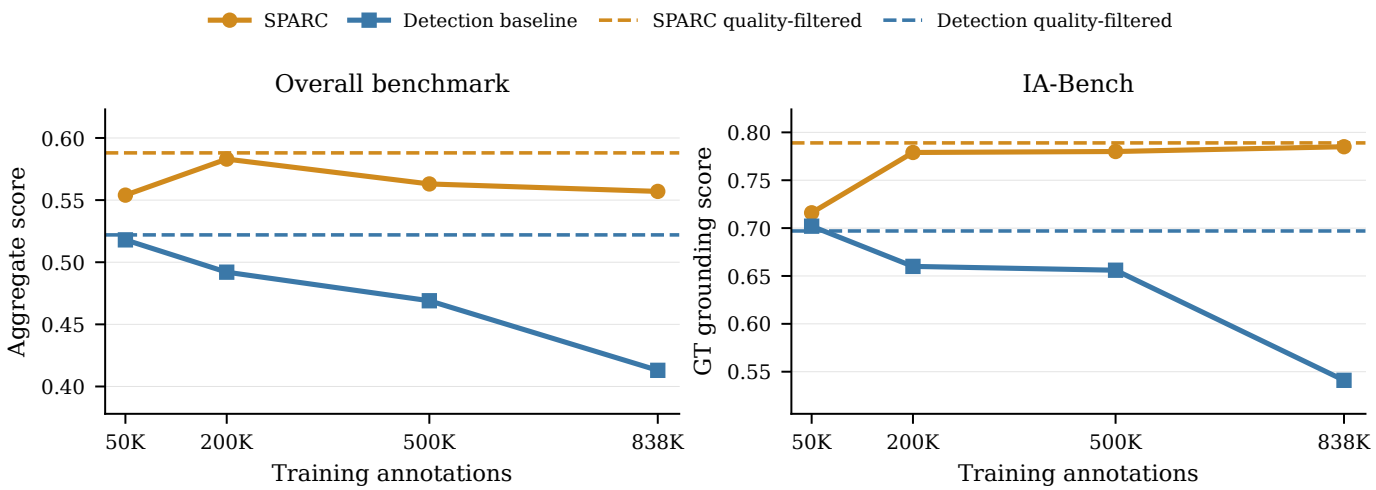

Fig. 6: **Scaling behavior across downstream benchmarks.** We train on the top-scoring 50K, 200K, 500K, and 838K annotations selected by SPARC or detector confidence. Dashed lines show the quality-filtered setting using the fixed threshold from the main experiments. Left: average performance across all downstream benchmarks. Right: performance on IA-Bench.

M. Comparison to Embodied Foundation Models

Method	Pointing and VQA ↑									Trajectory benchmarks ↓				
	IA Bench	Where2 Place	Ref Spatial	ERQA	Robo Spatial	Robo RefIt	EO Bench	VA Bench-P	**Avg.**	RoboInter Gripper	RoboInter Traj.	ShareRobot Bench-T	VA Bench-V	**Avg.**
Qwen3.5-4B (Zero-shot)	49.7	47.0	30.5	30.3	48.4	70.1	30.5	19.0	42.6	0.180	0.282	∞	0.219	∞
Open Source Brain Models														
RoboBrain 2.0 (7B)	-	63.5	54.0	30.3	54.2	70.4	-	26.67	-	-	-	0.236	-	-
MiMo-Embodied (7B)	-	63.6	48.0	46.7	61.8	82.3	-	46.9	-	-	-	-	-	-
RoboPoint (13B)	-	46.8	16.1	-	-	49.8	-	19.1	-	-	-	-	-	-
RoboRefer (2B SFT)	-	66.0	33.8	-	**66.4**	72.8	-	24.7	-	-	-	-	-	-
Gemini 2.5 Pro	-	53.0	29.2	48.3	-	49.5	-	21.7	-	-	-	-	-	-
Qwen2.5-VL (72B)	-	37.2	20.9	-	-	78.5	-	23.3	-	-	-	-	-	-
EO-1 (3B)	-	-	-	45.5	-	-	36.4	-	-	-	-	-	-	-
FSD (13B) [36]	-	45.8	-	-	-	56.7		61.8	-	-	-	-	-	-
RynnBrain (8B)	-	-	59.2	46.8	**73.1**	-	-	-	-	-	-	0.35	-	-
RynnBrain (2B)	-	-	52.7	42.3	65.7	-	-	-	-	-	-	0.34	-	-
Ours (4B)	**79.1**	**71.0**	**54.5**	**44.9**	**61.0**	**86.4**	35.3	**65.7**	**62.7**	**0.125**	0.319	**0.232**	**0.091**	**0.192**
Ours (0.8B, VT-FT)	76.7	58.0	37.1	40.4	53.5	81.0	30.3	48.3	53.2	0.175	0.429	0.354	0.216	0.294
Ours (0.8B)	75.4	62.0	33.1	36.9	45.5	81.0	34.6	56.3	53.1	0.157	0.425	0.358	0.208	0.287

TABLE XII: Comparison of Qwen3.5-4B trained on SPARC data against several recent state-of-the-art embodied foundation models. All values are taken from the respective papers. Although our VLM is smaller in size and trained on vastly less data without any human annotation, it remains competitive with larger models while even outperforming most of them consistently across several benchmarks.

N. Real Robot Setup and Policy Training

Fig. 7: Illustration of our real-world robot evaluation setup.

We conduct our real robot experiments in a tabletop manipulation setting. Specifically, we use a Franka-Panda manipulator with a Robotiq gripper and one external and one in-hand camera. We conduct the 10 tasks shown in XIII. The setup and the objects the robot has to manipulate are shown in Figure 7.

For policy training, we employ the following logic: For each trajectory (o_i, ℓ_i, a_i), we optionally retrieve a step-aligned reasoning trace c_i either generated from the detection baseline or SPARC and train the policy to produce this reasoning trace before predicting the low-level action. The reasoning trace is generated and filtered with the respective method (SPARC and Detection Baseline). We simplify each object's trajectory by uniformly resampling it into 5 waypoints along its arc length. This preserves the coarse geometric path while removing redundant high-frequency tracking samples. The trace is prompted as: *"Generate the 2D trajectory the object should follow to complete the task. Output exactly 5 points."* The VLM is supervised only on the assistant reasoning tokens using a masked language-modeling loss,

$$\mathcal{L}_{\text{cot}} = -\frac{1}{|\mathcal{M}|} \sum_{i} \sum_{t \in \mathcal{M}_i} \log p_\theta(c_{i,t} \mid o_i, \ell_i, c_{i,<t}), \tag{1}$$

where $\mathcal{M}_i$ denotes the unmasked CoT answer tokens. The action expert is a diffusion-transformer policy that predicts a horizon of $H = 24$ continuous 8-DoF absolute end-effector actions,

$$a_i = (a_i^1, \ldots, a_i^H), \qquad a_i^t \in \mathbb{R}^8. \tag{2}$$

It conditions on the last VLM hidden states h_i and learns to denoise noisy action trajectories. The final objective is

$$\mathcal{L} = \mathcal{L}_{\text{act}} + 0.1\, \mathcal{L}_{\text{cot}}. \tag{3}$$

At inference time, we first generate the reasoning trace using at most 160 new tokens, then re-encode the prompt and generated trace to obtain h_i and condition the action expert.

Cotraining with good spatial annotations gives the VLMs the capability to ground visually very similar objects. Notably, training on low quality annotations results in a significant performance drop.

O. Per Task Results

Table XIV shows policy success rates for each tasks. Reasoning VLAs trained on annotations generated by SPARC consistently outperform the baselines by a large margin on most tasks, highlighting the improved grounding and spatial localization capabilities resulting from higher quality spatial annotations.

P. Hyperparameters

ID	Language instruction
1	Put the yellow cube into the red bowl
2	Put the yellow cube into the red dustpan
3	Put the yellow cuboid into the red bowl
4	Put the yellow cuboid into the red dustpan
5	Put the yellow lego block into the red bowl
6	Put the yellow lego block into the red dustpan
7	Put the orange carrot into the red bowl
8	Put the orange carrot into the red dustpan
9	Put the orange lego block into the red bowl
10	Put the orange lego block into the red dustpan

TABLE XIII: Language instructions used in the real-robot ambiguity evaluation.

Method	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Total
Baseline	1	2	0	0	0	1	1	0	3	1	9/100
Detection	2	1	2	2	0	1	0	0	1	3	12/100
SPARC	3	7	4	4	1	1	1	2	3	5	31/100

TABLE XIV: Per-task real-robot success counts. Each entry denotes successful rollouts out of 10.

(a) Stage 2a: Single-arm task and object extraction from detected gripper phases

```
[SYSTEM] You are a robot manipulation expert. Given a task instruction and detected gripper phases with frame indices, extract object interactions (subtasks), assign phases to subtasks, and generate phase descriptions.
         Each phase has start_frame, end_frame, and phase_type in {"grasp","interact","release","grasp_failure"}. Output a JSON list with keys: object, start_location, target_location, action, tool_name_required,
        tool_usage_description, grasp_phases. Each grasp_phases item has: start_frame, end_frame, phase_type, description. Use present-tense instruction phrasing. Split multi-object tasks by frame order and task
        semantics. Keep repeated grasp cycles for the same object in one entry.

[EXAMPLE]
Task: Place the bag of chips in the bottom drawer
Detected phases: [{"start_frame":0,"end_frame":25,"phase_type":"grasp"},{"start_frame":26,"end_frame":80,"phase_type":"interact"},{"start_frame":81,"end_frame":100,"phase_type":"release"}]
Output: {"object":"bag of chips","start_location":null,"target_location":"bottom drawer","action":"pick and place","tool_name_required":null,"tool_usage_description":null,"grasp_phases":[{"start_frame":0,"end_frame
        ":25,"phase_type":"grasp","description":"Move towards the bag, lower the gripper, and close the gripper to grasp the bag."},{"start_frame":26,"end_frame":80,"phase_type":"interact","description":"Lift the bag
        and move it toward the bottom drawer."},{"start_frame":81,"end_frame":100,"phase_type":"release","description":"Place the grasped bag in the drawer."}]}

[QUERY TEMPLATE]
Task: <language_instruction>
Detected phases: <json_serialized_phases>
```

(b) Stage 2b: Bimanual task and object extraction from left/right phase streams

```
[SYSTEM] You are a robot manipulation expert. Given a task instruction and detected gripper phases for a bimanual robot with separate left-arm and right-arm timelines, identify which object each arm interacts with,
        assign phases to the corresponding subtask, and generate phase descriptions. Each phase has start_frame, end_frame, and phase_type in {"grasp","interact","release","grasp_failure","null"}. Output a JSON list
        with keys: arm, object, start_location, target_location, action, tool_name_required, tool_usage_description, grasp_phases. Each entry must specify arm as left or right. Do not merge arms into one entry. If an
        arm has no valid interaction, emit an entry with an empty grasp_phases list.

[EXAMPLE]
Task: Pick up the red block with the left arm and place the blue cup with the right arm
Left-arm phases: [{"start_frame":0,"end_frame":20,"phase_type":"grasp"},{"start_frame":21,"end_frame":60,"phase_type":"interact"},{"start_frame":61,"end_frame":70,"phase_type":"release"}]
Right-arm phases: [{"start_frame":5,"end_frame":25,"phase_type":"grasp"},{"start_frame":26,"end_frame":65,"phase_type":"interact"},{"start_frame":66,"end_frame":75,"phase_type":"release"}]
Output: [{"arm":"left","object":"red block","start_location":null,"target_location":null,"action":"pick up","tool_name_required":null,"tool_usage_description":null,"grasp_phases":[{"start_frame":0,"end_frame":20,"
        phase_type":"grasp","description":"Move the left gripper to the red block and grasp it."},{"start_frame":21,"end_frame":60,"phase_type":"interact","description":"Lift the red block."},{"start_frame":61,"
        end_frame":70,"phase_type":"release","description":"Release the red block."}]},{"arm":"right","object":"blue cup","start_location":null,"target_location":null,"action":"place","tool_name_required":null,"
        tool_usage_description":null,"grasp_phases":[{"start_frame":5,"end_frame":25,"phase_type":"grasp","description":"Move the right gripper to the blue cup and grasp it."},{"start_frame":26,"end_frame":65,"
        phase_type":"interact","description":"Move the blue cup to the target location."},{"start_frame":66,"end_frame":75,"phase_type":"release","description":"Place the blue cup down."}]}]

[QUERY TEMPLATE]
Task: <language_instruction>
Left-arm phases: <json_serialized_left_phases>
Right-arm phases: <json_serialized_right_phases>
```

Fig. 8: Prompt templates used for Stage 2 task/object extraction in the annotation pipeline. The pipeline uses the single-arm prompt for standard trajectories and the bimanual prompt when separate left/right gripper phase streams are available.