
What matters in building vision–language– action models for generalist robots

In the format provided by the
authors and unedited

Supplementary

APPENDIX A DISCUSSION

This empirical study mainly focuses on what matters in building Visual-Language-Action models (VLAs). We raise four essential questions for building a VLA: **Why** do we need VLAs instead of other generalist policies, and by outperforming the existing methods by a large margin, we illustrate the necessity of studying VLAs. For the next step, we describe the key components for building a VLM-based VLA: **Which** kind of VLM backbone to utilize, **How** to train the model to generate action, and **When** should we add cross-embodiment data into training stages. To answer these questions, we built a unified framework for a fair comparison of VLAs and designed a series of bottom-up systematic experiments. To answer these questions, we conduct extensive experiments across three simulators and more than 240 rollouts within 20 tasks in real-world scenarios, and we can conclude from the experiments that: For the **Why** question, VLAs could achieve high performance and generalization, and is a promising path to generalist robotics policy; For the **Which** problem, we find that VLMs with sufficient vision-language pre-training over large scale vision-language datasets is suitable for constructing VLAs. For the **How** problem, we can investigate the performance, generalization, and data efficiency of different VLA structures, and find that integrating history observations is essential for VLAs, and policy head is a more effective and efficient history aggregating method compared with interleaved; For the **When** problem, we compare three training recipes with cross-embodiment integrated at different stages, and conclude that extra in-domain data shows beneficial, and large-scale cross-embodiment pre-training further improves overall as well as few-shot performance. As a byproduct of answers to the raised questions, we built an easy-to-use framework for easily integrating arbitrary VLMs and turning them into VLAs, named **RoboVLMs**.

Under investigated observations. During our experiments, we found that VLAs built upon Qwen-VL and LLaVA, the performance is surprisingly low, compared with their original performance on vision-language tasks. After adding a perceiver resampler after the vision encoder, we found that the VLAs based on Qwen-VL and LLaVA could obtain great performance gains and reach reasonable performance. We hypothesize that the performance gain is related to the image resolution and the number of vision tokens in the input token sequence.

Limitations. Although we make every effort to investigate the key challenges in building Vision-Language Agents (VLAs), this work remains preliminary and has several limitations at the current stage. (1) The action tokenization, policy head, and corresponding training objectives are not fully explored in this work. For example, techniques like VQ-VAE, VQGAN, and FAST action tokenizer remain under-explored in the context of VLAs. (2) The set of VLM backbones considered in this study is limited and can be actively expanded. (3) Deploying such large models for real-time robotic control remains a significant challenge.

Future works. For future work, we envision several potential directions for advancing generalist robot policies. Beyond semantic generalization, an ideal generalist robot policy should be capable of handling long-horizon, complex task instructions (e.g., `make breakfast`), reasoning through executable actions step by step, and generating meaningful physical interactions with its environment. In our future work, we aim to explore the key elements required to develop policies with these advanced capabilities.

We have open-sourced our codebase with detailed guidelines, model weights of the strongest VLAs built by RoboVLMs, along with the real-world dataset used in our experiments. We anticipate that our research will bolster the community and expedite progress in the realms of vision-language learning and foundational models for robotics.

APPENDIX B IMPLEMENTATION DETAILS

a) Hyper-parameters and Training Details.: With different formulations, the best setting of hyperparameters like batch size, weight decay, and learning rate could be varied. Although OpenVLA suggests utilizing the same hyperparameters as in the VLM pretrain phase, we find that a varied setting of the hyperparameters could improve the performance.

The hyperparameters for fine-tuning VLAs are mainly derived from the VLMs training setups, for example, we select the weight decay from $[0, 1e - 1]$, and the learning rate as one of $[1e - 4, 5e - 5, 2e - 5, 1e - 5]$. We conduct a grid search and select the one with the best performance. We set the global batch size as 128 and the warmup ratio is 0.25 epoch (5K steps for OpenX Embodiment pre-train). All models included in this paper are trained on a cluster of 4 x 8 A100-80G GPUs.

b) Checkpoint selection.: We find out that, normally, the performance of robot policies does not fully depend on offline evaluation metrics, such as the validation loss, due to the compounding error in long-horizon rollouts. Therefore, it is challenging to select the best checkpoint during training. For fair comparisons, we train all VLAs for a fixed number of epochs or timesteps. Concretely, on CALVIN, we train each model for 5 epochs with a batch size of 128 truncated trajectories and report the performance of the final model. For SimplerEnv, we train the model for 100K iterations with a batch size of 512 truncated trajectories and report the best-performing model with a 10K-iteration training interval. In real-world experiments, we train the model for 5 epochs with a batch size of 512 truncated trajectories, and we only report the performance of the last model.

Supplementary Table 1: Hyper Parameters setup for different experiments. “EP” denotes the epoch. “Iters” represents iterations.

Experiment Name	Backbone	Window Size	Chunk Size	Input View	Batch Size	Warmup	Scheduler	Optimizer	Learning Rate	Total Epochs/Iters
CALVIN Perform (Extended Table. 2)	KosMos-2	16	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
SimplerEnv Training Recipe (Supplementary Fig. 1, Extended Fig.4)	PaliGemma (Co-Train)	1	10	Side	1024	500 Iters	Constant	AdamW	2e-5	20K Iters
	PaliGemma (Post-Train & Finetune)	1	10	Side	128	0.25 Ep	Constant	AdamW	5e-5	5 Ep
CALVIN VL Pre-train (Fig. 4b)	Flamingo & KosMos-2	16	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
Real Perform (Fig. 4d)	KosMos-2	8	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
VLA Structure (Table. I)	LLaVA	8	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	2e-5	5 Ep
	Flamingo & KosMos-2 & PaliGemma	16	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
CALVIN Generalization (Fig. 4c)	All	16	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
CALVIN Data Efficiency (Extended Table. 4)	All	16	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	1e-4	5 Ep
CALVIN Backbone (Extended Table. 5)	All	8	10	Side	128	0.25 Ep	Constant	AdamW	2e-5	5 Ep
CALVIN Training Objective (Table. IIa)	All	1	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	2e-5	5 Ep
CALVIN MoE (Table. IIb)	All	1	10	Side+Wrist	128	0.25 Ep	Constant	AdamW	2e-5	5 Ep
CALVIN few-shot (Extended Fig. 5)	All	16	10	Side	128	0 Iter	Constant	AdamW	2e-5	5K Iters

APPENDIX C BENCHMARK DETAILS

CALVIN is a simulated benchmark for multi-task table-top manipulation. It provides 24k human-teleoperated demonstrations annotated with language instruction. Each trajectory is less than 64-time steps, which includes 34 pre-defined basic skills: rotate blue block right, move slider right, lift red block slider, place slider, turn off light bulb, turn off led light, push in drawer, lift blue block drawer, close drawer, lift pink block slider, lift pink block table, move slider left, open drawer, turn on light bulb, rotate blue block left, push blue block left, rotate red block right, turn on led light, push pink block right, push red block left, lift blue block table, place in drawer, rotate red block left, push pink block left, lift stacked blocks, lift blue block slider, push red block right. The dataset contains four splits as scene A, B, C, and D. We train and test VLAs on different training/test splits to fully analyze the capabilities, data and training efficiencies. During evaluation, the robot is required to complete a set of 5 consecutive tasks. The metrics are the success rates of finishing these sequential tasks and the average length of completed tasks. All evaluations are implemented on D split, with 1000 rollouts and 5 consecutive sub-tasks for each rollout.

SimplerEnv are designed as a suite of real-to-sim environments, which enables evaluating robotic policies in simulation as efficient, scalable, and informative alternative to real-world evaluation. SimplerEnv created a comparable arena for benchmarking robotics policies on private real-world setups as Google and BridgeData V2.

We adopt the following tasks in the Google Robot setting:

- **pick coke can.** The task assigned to the robot is to pick up an empty Coke can from the table and lift it. Under the standard configuration, the environment is kept free of any distracting elements. The Coke can is arranged in three distinct positions: lying flat horizontally, lying flat vertically, and standing upright. For each of these positions, the can is placed at 25 specific grid points within a defined rectangular area on the table. This setup results in 25 experiments per position, totaling 75 trials across all orientations.
- **move {obj1} near {obj2}.** In the experiment, a set of three objects was arranged on the table in a triangular formation. For each trial, one object was assigned the role of the source, another was designated as the target, and the third served as a distractor. This setup resulted in six distinct trials for each triplet and triangular configuration. From a total of eight objects—blue plastic bottle, Pepsi can, orange, 7up can, apple, sponge, Coke can, and Redbull can—five triplets were randomly selected. Additionally, two triangular patterns, upright and inverted, were employed. This design produced a total of 60 trials.
- **(open/close) (top / middle/bottom) drawer.** In this setup, the robot is placed facing a cabinet equipped with three drawers and tasked with opening or closing a specific drawer. This experiment evaluates the robot’s capability to handle articulated objects. The robot is positioned at nine distinct locations on a predefined grid within a rectangular area on the floor. With three drawers and two possible actions (opening or closing), the setup results in a total of 54 trials.
- **open top drawer; place apple into top drawer.** In this experiment, the robot is tasked with opening the top drawer and transferring an apple from the surface of the cabinet into the drawer. This setup evaluates the robot’s ability to execute tasks that require multiple sequential actions. The robot is positioned in three distinct locations on the floor, while the apple is placed at nine specific grid points on the cabinet surface, resulting in a total of 27 trials. At the start, the robot operates under the instruction to open the top drawer. Once the robot either signals task completion with a "terminate" token or reaches the midpoint of the allotted time, the instruction transitions to directing the robot to place the apple into the drawer.

For the WidowX + Bridge setting, we test the following tasks:

- **put the spoon on the towel.** In this setup, the spoon is positioned at one corner of a square on the tabletop, with the towel placed at a different corner. The square has sides measuring 15 cm in length. The orientation of the spoon

alternates between horizontal and vertical, requiring the robot to adjust the orientation of its gripper accordingly. This configuration results in a total of 24 trials.

- **put carrot on plate.** This setup is similar to put the spoon on the towel, but the spoon is replaced with a carrot and the towel is substituted with a plate.
- **stack the green block on the yellow block.** In this experiment, a green block is positioned at one corner of a square on the tabletop, while a yellow block is placed at a different corner. Both blocks measure 3 cm in size. Two square configurations with 10 cm and 20 cm side lengths are used. This setup results in a total of 24 trials.
- **put eggplant into yellow basket.** An eggplant is positioned randomly within the right basin of a sink, while a yellow basket is placed in the left basin. The eggplant's placement varies in both location and orientation but is carefully arranged to remain easily graspable, avoiding proximity to the sink's edges. A total of 24 trials are conducted under this setup.

109

110

APPENDIX D
DETAILED PERFORMANCE ON CALVIN

Supplementary Table 2: Generalization and data efficiency of VLAs with or without vision-language pre-train on different settings of the CALVIN benchmark. “Inter.” denotes interleaved modeling. “P.H.” denotes policy head. “No VL” suggests models without VL pre-training. “5x” represents training with 5x re-generated training data. The results for 5x data are the model performance at 1 epoch, and the best-behaved model checkpoints within 5 epochs for the others.

Architecture	Train	Consecutive tasks success rates					Avg. Len.
		1	2	3	4	5	
KosMos Inter. No VL	ABCD	0.692	0.382	0.189	0.085	0.036	1.38
KosMos Inter.		0.987	0.915	0.824	0.737	0.660	4.12
KosMos P.H. No VL		0.815	0.626	0.473	0.349	0.245	2.51
KosMos P.H.		0.967	0.930	0.899	0.865	0.826	4.49
Flamingo P.H. 9B No VL		0.733	0.444	0.270	0.161	0.077	1.69
Flamingo P.H. 9B		0.955	0.879	0.784	0.714	0.634	3.97
KosMos Inter. No VL	5x ABCD	0.833	0.588	0.421	0.297	0.209	2.35
KosMos Inter.		0.989	0.940	0.892	0.842	0.795	4.46
KosMos P.H. No VL		0.893	0.755	0.644	0.564	0.462	3.32
KosMos P.H.		0.968	0.937	0.903	0.872	0.830	4.51
KosMos Inter. No VL	ABC	0.432	0.162	0.044	0.007	0.002	0.65
KosMos Inter.		0.824	0.684	0.524	0.376	0.296	2.70
KosMos P.H. No VL		0.389	0.121	0.038	0.008	0.001	0.56
KosMos P.H.		0.980	0.936	0.854	0.778	0.704	4.25

APPENDIX E
DIVERSE BACKBONE

Supplementary Table 3: Sub-task level success rates by tasks in CALVIN under different VLM backbones. All models trained with 5 epochs on CALVIN ABCD training splits.

Task Name	Flamingo-3B	Flamingo-4B	Flamingo-9B	KosMos	MoonDrean	Paligemma	Qwen	Uform
rotate blue block right	0.222	0.300	0.322	0.493	0.234	0.816	0.057	0.689
move slider right	0.736	0.989	0.805	0.987	0.988	1.000	0.442	1.000
lift red block slider	0.681	0.620	0.301	0.856	0.500	0.975	0.051	0.387
place in slider	0.683	0.813	0.765	0.874	0.361	0.957	0.308	0.490
turn off lightbulb	0.935	0.920	0.857	0.927	0.988	0.992	0.051	0.907
turn off led	0.990	0.925	0.966	0.970	0.990	0.987	0.793	0.754
push into drawer	0.422	0.318	0.296	0.705	0.688	0.814	0.000	0.743
lift blue block drawer	1.000	1.000	0.250	0.917	0.833	1.000	–	0.917
close drawer	0.118	1.00	0.990	0.986	0.991	1.000	0.160	0.950
lift pink block slider	0.754	0.633	0.471	0.861	0.444	0.945	0.059	0.703
lift pink block table	0.708	0.541	0.630	0.543	0.544	0.936	0.229	0.892
move slider left	0.952	0.952	0.961	0.970	1.000	1.000	0.787	0.994
open drawer	0.986	0.796	0.983	0.980	1.000	1.000	0.231	0.992
turn on lightbulb	0.737	0.225	0.561	0.949	0.990	1.000	0.170	0.775
rotate blue block left	0.358	0.588	0.596	0.636	0.500	0.955	0.000	0.964
push blue block left	0.554	0.509	0.696	0.677	0.424	0.727	0.000	0.661
rotate red block right	0.267	0.317	0.355	0.591	0.444	0.853	0.162	0.610
turn on led	0.962	0.864	0.902	0.985	0.973	0.970	0.562	0.526
push pink block right	0.365	0.574	0.538	0.627	0.271	0.559	0.027	0.412
push red block left	0.383	0.542	0.727	0.562	0.338	0.653	0.043	0.435
lift blue block table	0.470	0.474	0.564	0.611	0.691	0.955	0.111	0.889
place in drawer	0.969	0.932	0.928	0.971	0.958	0.994	0.750	0.931
rotate red block left	0.469	0.562	0.588	0.677	0.327	0.937	0.000	0.978
push pink block left	0.661	0.672	0.697	0.747	0.338	0.857	0.000	0.554
stack block	0.130	0.029	0.070	0.569	0.448	0.646	0.000	0.495
lift blue block slider	0.458	0.405	0.282	0.769	0.867	0.953	0.211	0.367
push red block right	0.233	0.561	0.557	0.457	0.194	0.414	0.103	0.350
lift red block table	0.629	0.608	0.554	0.606	0.640	0.967	0.091	0.835
lift pink block drawer	0.800	0.667	0.333	0.778	0.667	0.929	–	0.400
rotate pink block right	0.182	0.358	0.377	0.478	0.462	0.620	0.179	0.431
unstack block	1.000	0.500	1.000	0.946	0.950	0.935	–	0.880
rotate pink block left	0.578	0.442	0.479	0.698	0.146	0.964	0.034	1.000
push blue block right	0.091	0.421	0.320	0.400	0.066	0.324	0.000	0.500
lift red block drawer	0.400	0.625	0.800	0.769	0.667	1.000	–	0.556

APPENDIX F
DIVERSE PH

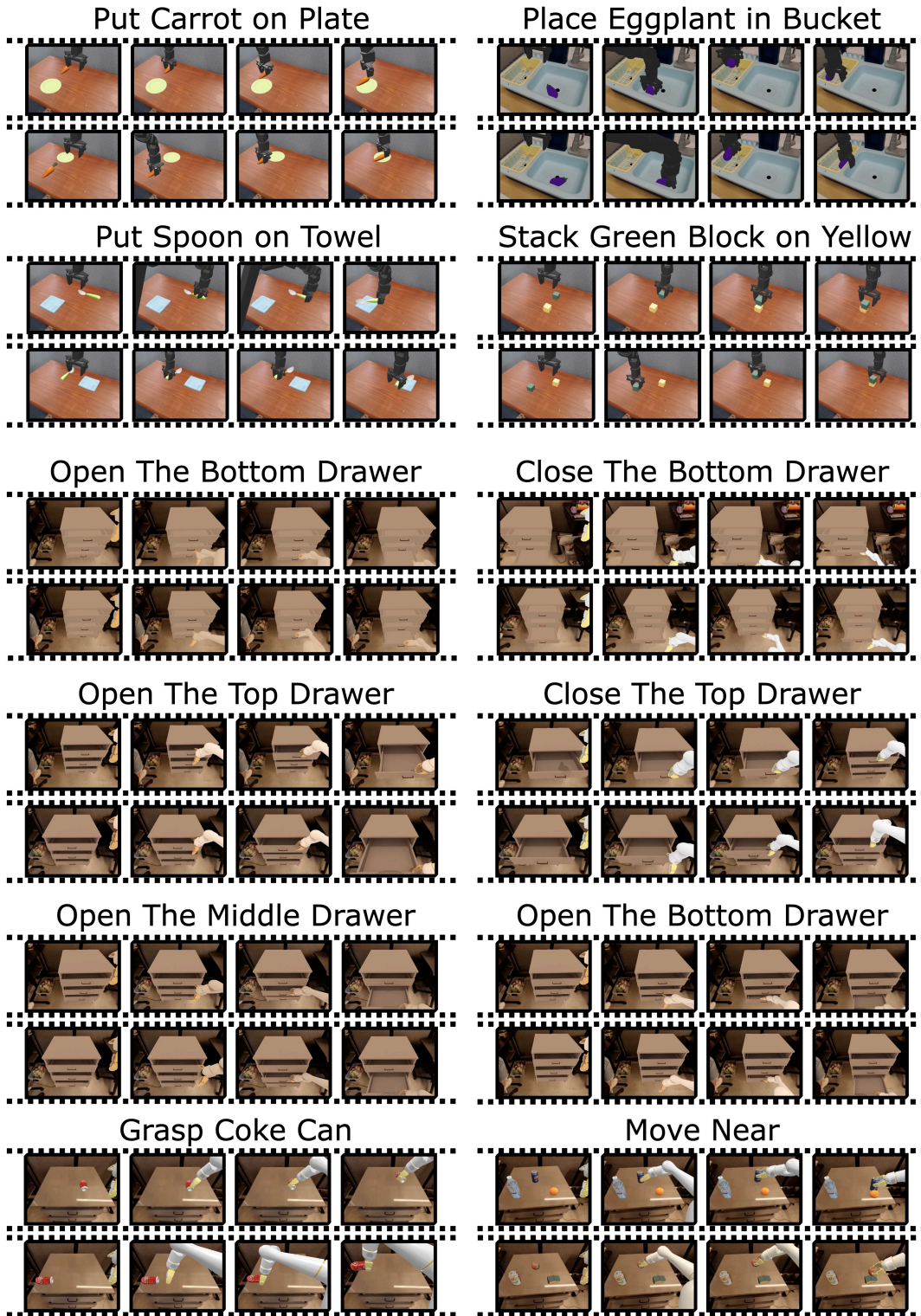
Supplementary Table 4: Sub-task level success rates by tasks in CALVIN under different training splits and VLM backbones. All models are trained with maximal 5 epochs.

Task Name	flamingo-3b-abc	flamingo-3b-abcd	flamingo-4b-abcd	flamingo-9b-abc	flamingo-9b-abcd	kosmos-abc	kosmos-abcd
rotate blue block right	0.831	0.893	0.770	0.722	0.882	0.960	0.974
move slider right	0.163	0.993	0.992	0.471	0.996	1.000	1.000
lift red block slider	0.642	0.970	0.858	0.610	0.927	0.906	0.993
place in slider	0.739	0.828	0.911	0.654	0.910	0.744	0.960
turn off lightbulb	0.852	1.000	0.956	0.135	0.964	1.000	1.000
turn off led	0.835	1.000	0.994	0.765	0.981	0.970	1.000
push into drawer	0.656	0.821	0.770	0.688	0.703	0.805	0.849
lift blue block drawer	0.818	0.950	1.000	0.800	0.737	0.944	1.000
close drawer	1.000	1.000	1.000	0.978	0.995	1.000	1.000
lift pink block slider	0.674	0.971	0.862	0.733	0.918	0.925	0.993
lift pink block table	0.871	0.851	0.892	0.872	0.899	0.913	0.961
move slider left	0.517	0.996	1.000	0.798	0.996	0.992	1.000
open drawer	1.000	0.997	0.982	0.991	0.997	1.000	1.000
turn on lightbulb	0.922	0.994	0.988	0.241	0.988	0.989	1.000
rotate blue block left	0.915	0.939	0.925	0.755	0.848	0.985	1.000
push blue block left	0.909	0.955	0.836	0.736	0.909	0.914	0.826
rotate red block right	0.893	0.972	0.905	0.727	0.959	1.000	0.973
turn on led	0.956	0.988	0.976	0.777	0.994	0.977	0.995
push pink block right	0.706	0.754	0.651	0.652	0.750	0.708	0.779
push red block left	0.983	0.920	0.949	0.911	0.908	0.910	0.848
lift blue block table	0.936	0.956	0.927	0.876	0.931	0.995	0.995
place in drawer	1.000	0.989	0.975	0.969	0.976	1.000	1.000
rotate red block left	0.942	0.908	0.953	0.961	0.952	1.000	1.000
push pink block left	0.889	0.920	0.973	0.959	0.933	0.959	0.883
stack block	0.489	0.641	0.595	0.588	0.604	0.801	0.908
lift blue block slider	0.607	0.963	0.826	0.595	0.869	0.835	0.978
push red block right	0.825	0.732	0.451	0.764	0.653	0.871	0.681
lift red block table	0.891	0.939	0.975	0.958	0.989	0.989	0.984
lift pink block drawer	0.444	0.800	0.714	0.333	0.923	0.846	0.857
rotate pink block right	0.768	0.896	0.794	0.787	0.789	0.985	0.986
unstack block	0.880	0.982	0.980	0.909	0.979	0.985	1.000
rotate pink block left	0.974	0.839	0.818	0.947	0.927	1.000	0.965
push blue block right	0.620	0.597	0.478	0.511	0.594	0.870	0.657
lift red block drawer	0.714	1.000	1.000	0.833	0.933	0.950	1.000

115

116

APPENDIX G
ROLLOUT EXAMPLES IN SIMPLERENV



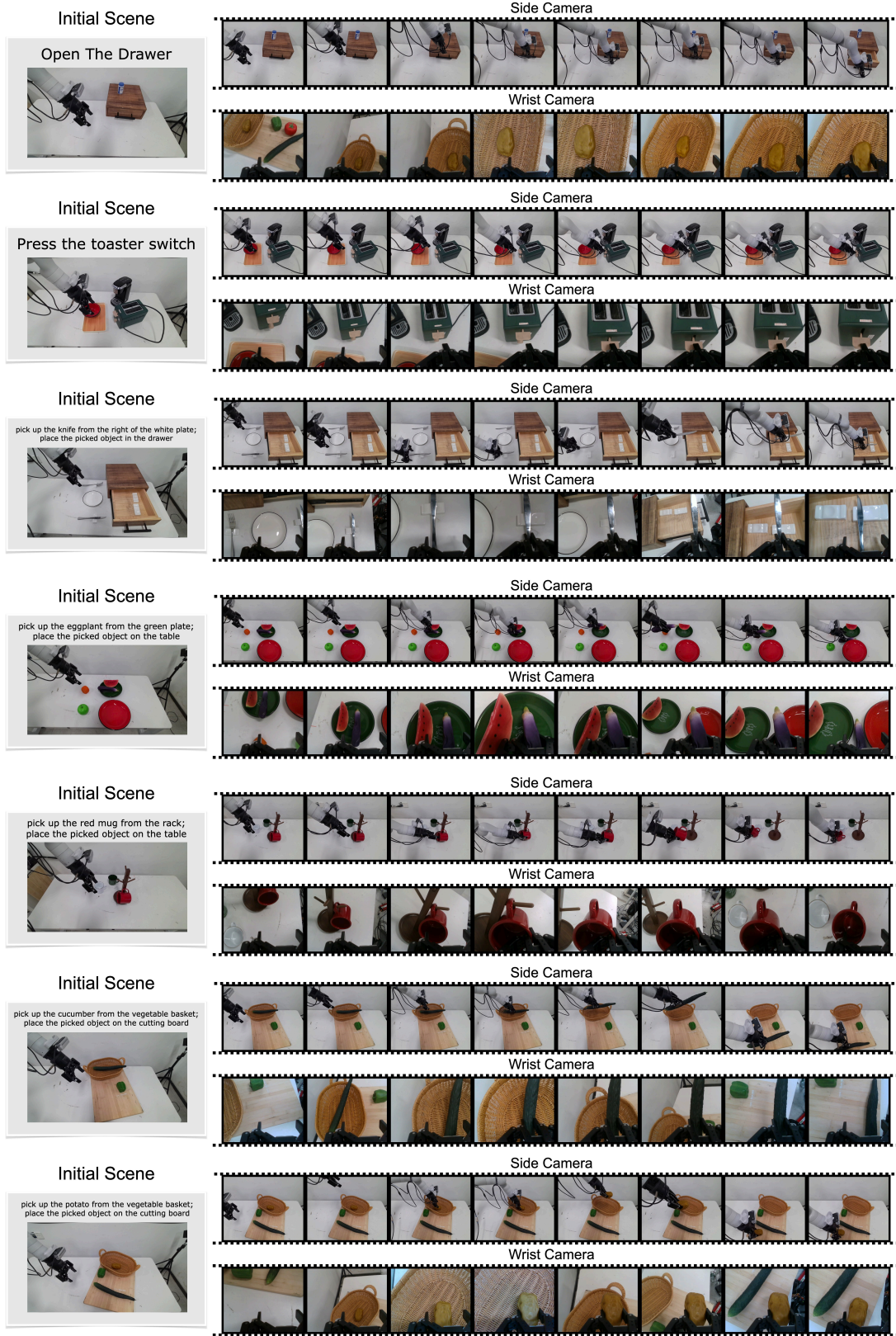
Supplementary Figure 1: Qualitative results for SimplerEnv tasks.

117
118

APPENDIX H
ROLLOUT EXAMPLES IN REAL-WORLD EXPERIMENTS



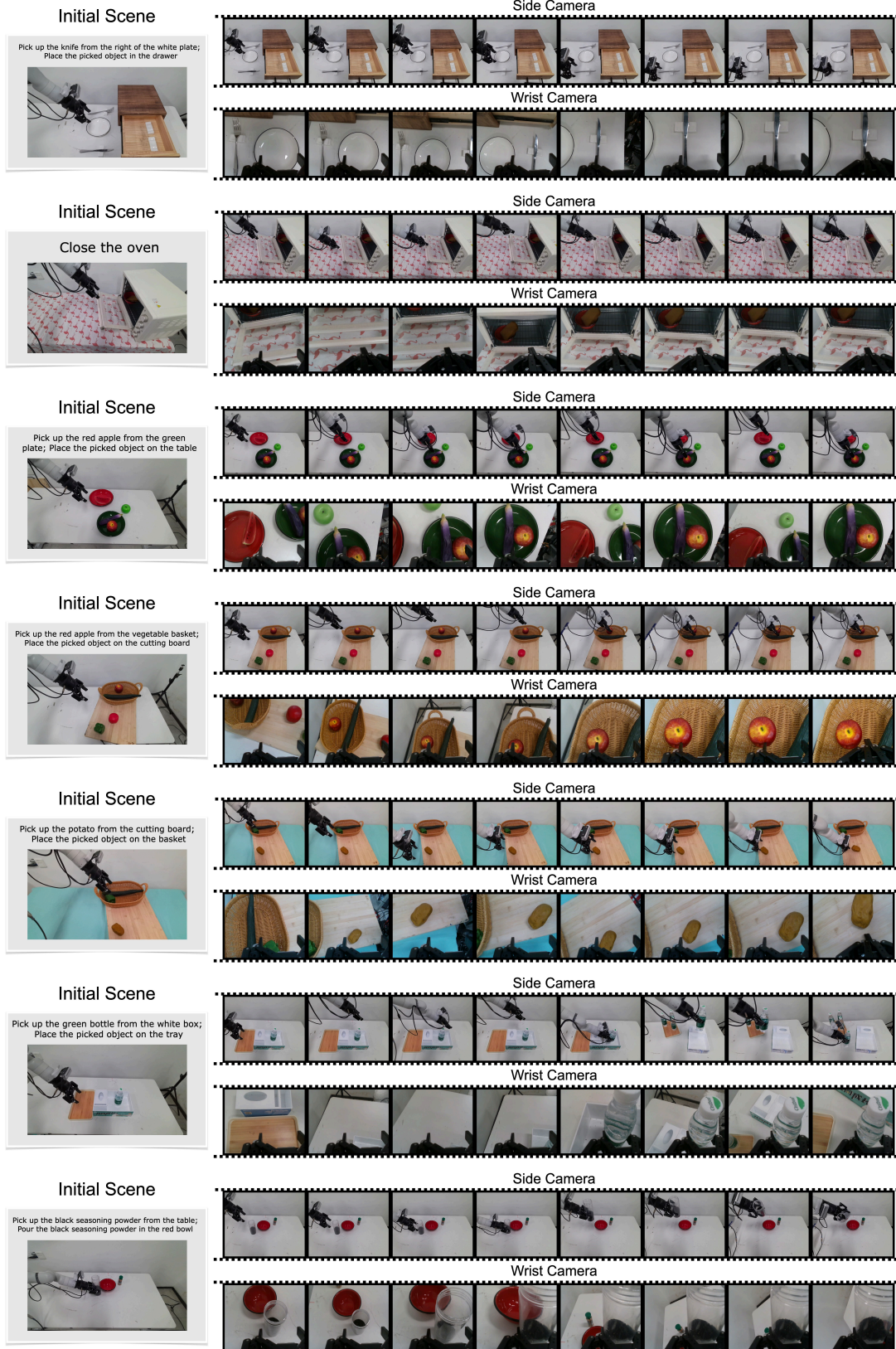
Supplementary Figure 2: This figure illustrates the experimental setup of some real-world tasks. The models are evaluated across 20 tasks, each with 5 rollouts, involving unseen distractors, unseen backgrounds, unseen target objects, and novel skill descriptions. Note that some tasks exclude the unseen target object setting due to the lack of suitable alternative unseen objects.



Supplementary Figure 3: Qualitative results for basic setting in real-world experiments.

Supplementary Figure 4: Qualitative results for unseen background.

Supplementary Figure 5: Qualitative results for unseen distractors and objects.



Supplementary Figure 6: Our model exhibits several typical failure cases. For instance, it might prematurely close the gripper, fail to accurately grasp the target object, exhibit repeated oscillations, or successfully pick up an object but cannot place it in the correct location.