

Action-Sketcher: From Reasoning to Action via Visual Sketches for Robotic Manipulation

Supplementary Material

Appendix

This supplementary material provides more details of the proposed method and experiment results that are omitted from the manuscript due to the page limit. Sec. A presents additional details of the models, training strategies and evaluation settings. Sec. B presents details of training dataset. Sec. C presents the pipeline of data construction. Sec. D complements more experiment results and analysis. Sec. E shows more visualization results to prove the effectiveness of Action-Sketcher. Sec. F discusses potential future research directions and limitations for Action-Sketcher.

A. Details of Model, Training and Evaluation

In this section, we provide comprehensive implementation details of the Action-Sketcher framework. We first elaborate on the specific model architecture based on the π_0 VLA policy and define the distinct input-output flows that facilitate the cyclic See-Think-Sketch-Act loop. Subsequently, we specify the detailed hyperparameters, computational configurations, and optimization strategies utilized across our three-stage curriculum training.

A.1. Model Structure

The Action-Sketcher framework is model-agnostic and can be integrated with any Vision-Language-Action (VLA) model. For our implementation and experiments, we employ π_0 [9] as the base VLA policy. π_0 is a generalist robot policy that leverages a VLM backbone for semantic understanding and a Flow Matching policy for continuous action generation, which consists of three primary components:

- **Vision Encoder:** a pre-trained Vision Transformer (*i.e.*, SigLIP-SO400M [83]) to process visual inputs. This encoder maps high-resolution images from multiple camera views (*e.g.*, wrist and ego-centric views) and the rendered *Visual Sketch* image into a sequence of visual tokens.
- **Auto-Regressive LLM Backbone:** The core reasoning unit is a Transformer-based LLM (*i.e.*, Gemma-3B). It takes a concatenated sequence of tokens as input, including the visual tokens, tokenized task instructions, and text-based history sub-tasks tokens. This backbone operates auto-regressively to perform high-level reasoning, generate text-based sketch parameters and sub-task description in the Reasoning Mode.
- **Flow Matching Action Expert:** For low-level control (*i.e.*, Action Mode), the model utilizes a Flow Matching head. Instead of auto-regressively discretizing actions,

this head acts as a conditional generative model. It takes the robot’s proprioceptive state and the latent embeddings from the LLM backbone as conditioning to generate continuous, high-frequency action chunks via solving a differential equation (ODE).

A.2. Inputs and Outputs

Based on the VLA model, Action-Sketcher operates in a cyclic loop with distinct input-output flows depending on the active mode (*i.e.*, Reasoning Mode and Action Mode). The inputs and outputs are as follows:

- **Reasoning Mode Inputs:** The model receives: (1) Multi-view RGB images $I_t^{1:n}$ capturing the current scene (*i.e.*, left-wrist, left-wrist and ego-centric views); (2) The task instruction ℓ (*e.g.*, “Clean the objects on the table”); (3) Textual history of completed sub-tasks. (4) If the current sub-task is not the first in the episode, the rendered *Visual Sketch* image produced for the immediately preceding sub-task.
- **Reasoning Mode Outputs:** Triggered by the $\langle \text{BOR} \rangle$ (Begin-of-Reasoning) token, the LLM auto-regressively generates a textual spatial and temporal rationale followed by the *Sub-Task* and the structured definition of the *Visual Sketch*, including coordinates for bounding boxes, keypoints, and arrows (*i.e.*, `{"bbox": [...], "points": [...], "arrow": [...] }`). This mode is terminated by $\langle \text{EOR} \rangle$ and texture *Visual Sketch* is then rendered into an image.
- **Action Mode Inputs:** Once the *Sub-Task* and *Visual Sketch* is generated and rendered, the model inputs are updated to include: (1) The current observation images $I_t^{1:n}$; (2) The rendered *Visual Sketch* image, which explicitly overlays the spatial intent on the reference view; (3) The robot’s current proprioceptive state s_t (*e.g.*, joint angles or end-effector pose).
- **Action Mode Outputs:** Once triggered by the $\langle \text{BOA} \rangle$ (Begin-of-Action) token from LLM, the Action Expert predicts a sequence of future actions (an action chunk) A_t . Specifically, it predicts the flow vector field to denoise a random noise sample into a valid trajectory of end-effector poses/joint angles and gripper states.

A.3. Training Setting

In the main text of the paper, we employed a staged training strategy, with complete settings presented in Tab. 5. During the entire training phase, we conducted all experiments on a cluster of servers, each equipped with $8 \times \text{H100}$ GPUs. We

Table 5. **Detailed configuration for each training stage of Action-Sketcher.** The table presents the training statistics and hyperparameters across the three-stage curriculum. Stage 1 focuses on foundational spatiotemporal representations, Stage 2 on reasoning-to-sketch generation, and Stage 3 on action policy alignment and mode adaptation.

		Stage 1 (Foundational Spatiotemporal Learning)	Stage 2 (Reasoning-to-Sketch)	Stage 3 (Sketch-to-Action / Mode Adaptation)
Data	Dataset Source	Grounding / Pointing / Planning	Reasoning-to-Sketch Dataset	Reasoning-Sketch-Action Corpus
	#Samples	$\sim 3.4\text{M}$ (Spatial) + 870k (Temp.)	$\sim 21\text{k}$ Samples	$\sim 6\text{k}$ Episodes (2.8M Samples)
	Data Modality	Image + Text	Image + Text \rightarrow Sketch	Image + Text + Sketch \rightarrow Action
Model	Base Architecture	π_0 (PaliGemma-3B)	π_0 (PaliGemma-3B)	π_0 (PaliGemma-3B)
	Trainable Parts	Vision Encoder + LLM	Vision Encoder + LLM	LLM + Action Expert
	Action Head	Frozen	Frozen	Flow Matching
Training	Global Batch Size	256	32	32
	Gradient Accumulation	4	2	2
	LR: $\{\psi_v^{\text{VIT}}, \phi_v^{\text{LLM}}, \phi_v^{\text{AE}}\}$	$\{1 \times 10^{-6}, 1 \times 10^{-4}, N/A\}$	$\{1 \times 10^{-6}, 5 \times 10^{-5}, N/A\}$	$\{N/A, 1 \times 10^{-5}, 1 \times 10^{-4}\}$
	LR Schedule	Cosine Decay	Cosine Decay	Constant with Warmup
	Warmup Ratio	0.03	0.03	0.05
	Optimizer	AdamW	AdamW	AdamW
	Weight Decay	0.05	0.05	0.05
	Image Resolution	224×224	224×224	224×224
	Max Seq. Length	4096	8192	8192
	Precision	bfloat16	bfloat16	bfloat16
	Training Epochs	1	1	20
	Hardware	$32 \times \text{H100}$	$8 \times \text{H100}$	$8 \times \text{H100}$

implement our models using PyTorch π_0 codebase. We utilize the AdamW optimizer with a cosine decay learning rate schedule and a weight decay of 0.05 across all stages. To ensure training stability and efficiency with high-resolution inputs (224×224), we employ bfloat16 mixed-precision training. The specific curriculum details are as follows:

- **Stage 1: Foundational Spatiotemporal Learning.** To handle the large-scale pre-training corpus ($\sim 4.3\text{M}$ samples combined), we scale the training to 32 H100 GPUs. We use a global batch size of 256 to efficiently digest the diverse visual grounding and temporal planning data. In this stage, the Action Head is not initialized; we focus on tuning the Vision Encoder (LR 1×10^{-6}) and the LLM backbone (LR 1×10^{-4}) to master spatiotemporal reasoning within a context length of 4096 tokens.
- **Stage 2: Reasoning-to-Sketch Alignment.** We fine-tune the model on the curated 21k-sample dataset to specialize in generating *Visual Sketches*. The context window is expanded to 8192 tokens to accommodate long-horizon reasoning history and detailed sketch definitions. We reduce the global batch size to 32 and lower the LLM learning rate to 5×10^{-5} to preserve the pre-trained knowledge while aligning the text output with the specific JSON-based sketch format.
- **Stage 3: Sketch-to-Action and Mode Adaptation.** In the final stage, we jointly train the full model, including the previously frozen components and the Flow Matching

action head. To support the complex multimodal mapping from sketches to continuous actions, we employ a differential learning rate strategy: the Vision Encoder remains frozen to maintain feature stability, the LLM is fine-tuned with a reduced learning rate (1×10^{-5}), and the Action Expert is trained with a higher rate (1×10^{-4}). This stage runs for 20 epochs to ensure the action policy fully converges with the sketch-conditioned inputs.

A.4. Evaluation Details

We provide comprehensive details about our evaluation protocols, metrics, and experimental setup across both simulation and real-world environments. For real-world experiments, we deploy our trained policies on two robotic platforms: Aloha AgileX and Galaxea R1.

- **RoboTwin 2.0 (Sim).** We evaluate on five custom manipulation tasks in the RoboTwin 2.0 simulation environment. Each task is evaluated over 100 rollouts using the same random seed across all models to ensure fair comparison. Success is determined by the simulator’s built-in success condition, which checks whether the episode meets the task-specific completion criteria.
- **Long-Horizon Tasks (Real).** For the two long-horizon manipulation tasks (Tidy Table and Pour Tea), we evaluate on 10 rollouts per model. To test robustness, we modify the scene every 2 rollouts by changing object positions and introducing new object types. All models are evalu-

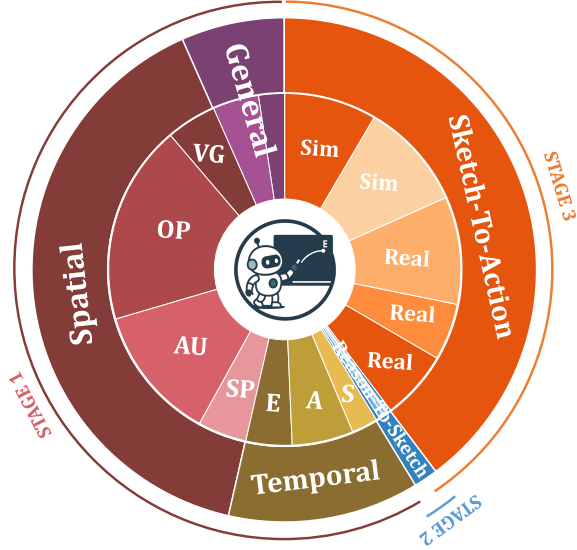


Figure 5. The distribution of the entire training dataset.

ated on the same scenario before transitioning to the next scene configuration.

- **General Pick and Place (Real).** We evaluate on 30 roll-outs per model, modifying the scenario every 5 rollouts to test generalization across diverse spatial arrangements and distractor objects. As with the long-horizon tasks, all models complete evaluation on each scenario before scene changes.

B. Details of Training Dataset

In the main body of the paper, we emphasize the importance of the training data and the proportion of robotic data. In this section, we will provide a detailed overview of the training data and its sources. The distribution of the entire training dataset is illustrated in Fig. 5.

- **General MLLM VQA (470k).** To establish a robust foundation for multimodal understanding, we construct a general visual question-answering dataset comprising 470k high-quality samples. This subset is primarily derived from two sets: LLaVA-665k [48] and LRV-400k [47], encompassing diverse tasks such as standard VQA, OCR-based queries, and visual dialogues.
- **Visual Grounding (327k).** We enhance object-level localization capabilities by leveraging extensive annotations from LVIS [23], Ref-L4 [12], OV-VG [2, 35], Ref-COCO [56, 81], and Visual Genome [33]. In this subset, we include 327k high-resolution images with the standardized bounding box coordinates (x_1, y_1, x_2, y_2) .
- **Object Pointing (1.3M).** This module focuses on precise coordinate identification in embodied contexts. We process the Pixmo-Points [17] dataset through a two-step

filtering mechanism, *i.e.*, limiting point counts and utilizing GPT-4o [27] to select indoor-relevant scenes, resulting in 190k QA pairs optimized for reduced clutter. Additionally, we incorporate 347k object reference samples from RoboPoint [82] and extend capabilities to physical-world interactions using the Spatial Referring Dataset [86] (802k samples), ensuring the model can generalize across diverse pointing tasks.

- **Affordance Understanding (881k).** We address both functional and spatial affordances to support object interaction. For functional reasoning, we utilize part-level annotations from PACO-LVIS [64] to generate 561k QA pairs regarding object functionality and part usage. For spatial affordance, we integrate 320k region-reference pairs from RoboPoint, where coordinate annotations are optimized to help the model identify valid placement areas and spatial relationships in real-world settings.
- **Spatial Perception (318k).** To enable fine-grained 3D environmental understanding, we integrate multiple 3D-centric datasets: MMScan-224k [54] for segmentation and description, 3RScan-43k [78] for semantic reconstruction, as well as ScanQA-25k [4] and SQA3D-26k [55] for spatially grounded question answering.
- **Temporal Planning (870k).** To empower the model with robust temporal reasoning and execution capabilities, we construct a unified planning dataset comprising 870k samples. This collection synthesizes diverse embodied scenarios from three primary sources: EgoPlan-IT [15] for egocentric task planning, ShareRobot [28] to support fine-grained manipulation, and AgiBot-World [11] for continuous visual sequences from diverse household environments. This subset facilitates long-horizon task decomposition and sequential reasoning.
- **Reasoning-To-Visual (21k).** To bridge high-level semantic reasoning with precise visual grounding, we construct a specialized dataset containing 21k high-quality instruction-to-sketch pairs. Data generation employs a dual-pipeline strategy: an automatic pipeline leveraging SAM [32] and GPT-4o for scalable annotation, and a human-in-the-loop pipeline utilizing a custom interface to ensure geometric accuracy. Each sample pairs a textual sub-task instruction with a structured Visual-Sketch (bounding boxes, keypoints, and directional arrows), augmented by GPT-4o-generated reasoning chains that justify both the temporal necessity of the action and the spatial rationale behind the visual cues.
- **Visual-To-Action (2.8M).** To translate visual plans into executable motor controls, we compile a large-scale trajectory dataset aligning Visual-Sketches with low-level action sequences. This corpus spans 6.1k episodes, resulting in 2.8M state-action pairs derived from LIBERO [46], a modified complex version of RoboTwin 2.0 [14], and real-world teleoperation data. Each sample integrates

multi-view observations, the generated Visual-Sketch, and reasoning context to predict precise action chunks, ensuring the model learns to ground its manipulation policy effectively in the provided visual guidance.

C. Details of Data Construction

For Stage 2, to equip the model with downstream task-reasoning proficiency and the capability to generate semantically grounded Visual Sketches, we instantiate three independent annotation pipelines: (i) an automatic Visual-Sketch pipeline, (ii) a human Visual-Sketch pipeline, and (iii) an automatic Reasoning-generation pipeline that is grounded on the Visual-Sketch data produced by the first two pipelines. This stage is correspondingly organized into two complementary subsections: Visual-Sketch generation (Sec. C.1) and Reasoning generation (Sec. C.2). For Stage 3, to construct the full Reasoning-Sketch-Action training corpus, we integrate the Visual-Sketches and reasoning annotations from Stage 2 with the corresponding action sequences in the demonstration trajectories (Sec. C.3).

C.1. Visual-Sketch Generation

- **Automatic annotation pipeline.** Each demonstration is temporally decomposed into sub-tasks by detecting consecutive gripper open/close transitions. SAM [32] is applied to the start and end frames of every segment to produce instance masks; the target object is designated as the mask exhibiting maximal centroid displacement. The egocentric image pair, the target’s bounding box, and its 2-D start and end coordinates are forwarded to GPT-4o with a deterministic prompt that requests (i) a concise sub-task instruction and (ii) a JSON-compliant Visual-Sketch record containing bounding box, keypoints, and arrows.
- **Human annotation pipeline.** We implement a PyQt interface (Fig. 6) that streams HDF5, MP4, JPG, or PNG sequences. Annotators navigate to any frame, overlay the same JSON primitives, and validate or rewrite the sub-task description; the interface enforces schema consistency and averages 2.3 subtask annotations per minute. The identical tool is optionally deployed for human-in-the-loop refinement during inference, allowing real-time correction of model-generated sketches.

Quality assessment. Ablations in Sec. D reveal that both pipelines improve success rates over baseline in simulation, but the human-curated set yields a further +7.6% absolute gain, primarily ascribed to fewer inaccurate sketches. Consequently, all main-results models are trained on the human-refined split, while the automatic pipeline will be released as an open-source tool to enable scalable future expansion.

C.2. Reasoning Generation

Reasoning data are produced along two complementary axes: temporal and spatial. Both branches leverage GPT-4o conditioned on the Visual-Sketch annotations.

- **Temporal reasoning.** Given an episode-level list of sub-tasks and the Visual-Sketch image for the current key-frame, we prompt GPT-4o to produce a concise chain-of-thought that (i) recalls the high-level instruction, (ii) summarises the scene state, (iii) states the previously completed sub-task, and (iv) justifies the current sub-task as the next necessary action. The complete prompt template is provided in Fig. 8.
- **Spatial reasoning.** For every key-frame we additionally query GPT-4o to explain *why* the ground-truth visual prompts are the minimal set required to accomplish the current sub-task. The prompt conditions on the sub-task string and the JSON-encoded sketch; the complete template is provided in Fig. 9.

C.3. Sketch-Action Generation

For each episode, we align the generated sub-tasks and Visual-Sketches with the corresponding action sequences. Crucially, all action samples belonging to the same sub-task share the same reference Visual-Sketch, ensuring consistent spatial grounding throughout sub-task execution. This results in approximately 2.8M action prediction samples across 6.1k episodes, with each sample containing multi-view observations, the rendered Visual-Sketch for the current sub-task, reasoning text, and ground-truth action chunks. Our training corpus combines simulation and real-world data to ensure robust generalization:

- **LIBERO [46].** We utilize the open-sourced training sets from four task suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, which provide diverse manipulation scenarios with varying spatial reasoning and long-horizon planning requirements. We use third-person and wrist camera images along with robot proprioceptive state and language instructions.
- **RoboTwin 2.0 [14].** We select 5 tasks from RoboTwin 2.0 and introduce modifications to increase complexity, including random object clutter, increased camera height to reduce object size, and extended episode length. These modifications ensure tasks are sufficiently long (≥ 4 sub-tasks) and test spatial understanding. Task details are summarized in Tab. 6. All tasks use the Aloha Agile-X embodiment with egocentric and left/right wrist views, controlled via absolute joint positions.
- **Real-World Data.** To validate real-world transfer, we collect two long-horizon manipulation tasks and one spatial understanding task in laboratory settings. Details are provided in Tab. 6. All tasks use absolute joint position control with egocentric and wrist camera views.

Table 6. RoboTwin 2.0 and Real-world task specifications.

Task	Description	Robot	#Episodes
RoboTwin 2.0 Tasks			
Stack Blocks	Stack three blocks with varying color orders (RGB/GBR)	AgileX Piper	550
Hanging Mug	Grasp and hang mug on rack	AgileX Piper	200
Place Cups	Place two cups to nearest plates among randomly positioned plates	AgileX Piper	200
Place A2B	Place object relative to another’s left or right side	AgileX Piper	550
Real-world Tasks			
Tidy Table	Place diverse objects in containers (16 sub-tasks)	AgileX Piper	342
Pour Tea	Pour tea into two teacups (7–8 sub-tasks)	AgileX Piper	749
Pick & Place	Place object amid distractors	Galaxea R1	1,500

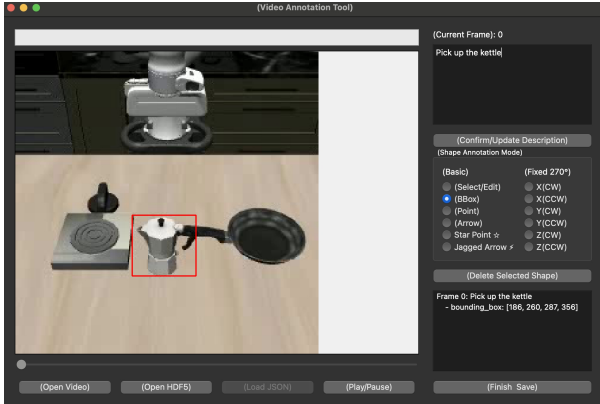


Figure 6. **Human annotation interface.** Screenshot of our PyQt-based annotation tool showing the interface for creating Visual-Sketch annotations. Annotators can navigate frames, overlay bounding boxes, keypoints, and directional arrows, and edit sub-task descriptions in real-time.

D. Complementary Experiments

In this section, we present the complete experiments and results that are omitted from the main manuscript due to page limitations. We present supplementary experiments that provide further insights into our approach but were omitted from the main manuscript due to space constraints.

D.1. Ablation Experiments in Detail

In the main manuscript, we conducted several ablation studies to validate key design choices of Action-Sketcher. Here we provide additional details and deeper analysis of those experiments. To validate the necessity of each component in our Action-Sketcher framework, we conducted controlled ablations by removing or modifying specific modules while keeping all other training and evaluation protocols identical.

w/o Spatial Reasoning. We trained a model to directly generate Visual-Sketches without the intermediate spatial

reasoning step. Specifically, the model predicts the JSON-encoded sketch parameters directly after temporal reasoning, bypassing the chain-of-thought explanation about why specific visual prompts are necessary. This ablation isolates the impact of explicit spatial reasoning on generating accurately grounded Visual-Sketches.

w/o Visual Sketches. We trained a model that undergoes the same reasoning steps as the original Action-Sketcher (generating both temporal and spatial reasoning), but Visual-Sketches are not rendered or overlaid on the reference images during action execution. Instead, only the original observation images are provided to the action prediction module. This ablation isolates the impact of using Visual-Sketches as explicit spatial reference during action execution, as opposed to simply providing additional reference images.

The results demonstrate that both spatial reasoning and Visual-Sketch conditioning contribute significantly to task performance, with the full Action-Sketcher framework achieving the best results across all evaluation metrics.

D.2. Sub-task vs. Task-level Visual Sketching

Our first supplementary experiment studies the effect of the granularity of visual sketching. We compare task-level sketches, where a single Visual-Sketch is provided for the entire task, against sub-task-level sketches, where a separate Visual-Sketch is given for each annotated sub-task (Action-Sketcher). Both variants are evaluated on the same RoboTwin 2.0 tasks in simulation, and use identical training and evaluation protocols except for the sketching level. The results are reported in Tab. 7.

Analysis. The results reveal a clear advantage of sub-task-level sketching across all tasks, with our approach achieving an average improvement of 11.5% absolute over task-level sketching. Notably, the performance gap varies significantly based on task complexity and length. The Stack Blocks task, which requires precise sequential manipulation

Table 7. Sub-task vs task-level visual sketching on RoboTwin 2.0 simulation tasks (success rate %).

Method	Stack	Hang	Empty	A2B-L	A2B-R	Avg.
Task-Level	9.5	20.0	15.0	35.5	21.0	20.2
Sub-Task (Ours)	34.5	25.0	28.0	43.0	28.0	31.7
Δ Improvement	+25.0	+5.0	+13.0	+7.5	+7.0	+11.5

across 6 sub-tasks with strict spatial ordering constraints, shows the largest improvement (+25.0%), while the redundant Hanging Mug task shows a smaller but still substantial gain (+5.0%).

We hypothesize that as tasks grow longer with more sub-tasks, reasoning about the entire trajectory at once and generating a single comprehensive Visual-Sketch actually introduces more noise and hurts performance. Task-level sketching requires the model to simultaneously plan multiple manipulation stages, leading to less precise spatial grounding for individual actions and potential conflicts between overlapping visual prompts. In contrast, our sub-task-level approach decomposes complex tasks into manageable reasoning steps, allowing the model to focus on generating accurate Visual-Sketches for each immediate manipulation goal. This demonstrates the advantage of our method not only in general performance but especially on longer, more complex tasks where decomposed reasoning and progressive sketch generation are critical for success.

D.3. Automated vs. Human Annotation

The second supplementary experiment compares the performance of policies trained with human-annotated Visual-Sketches to those trained with sketches produced fully automatically by our pipeline. The human-annotated pipeline consistently achieves higher success rates, while our automated pipeline still performs better than the baseline. Results are summarised in Tab. 8.

Table 8. Human vs automated annotation on RoboTwin 2.0 simulation tasks (success rate %).

Method	Stack	Hang	Empty	A2B-L	A2B-R	Avg.
Automated	21.5	21.0	23.0	32.0	23.0	24.1
Human	34.5	25.0	28.0	43.0	28.0	31.7
Δ Improvement	+13.0	+4.0	+5.0	+11.0	+5.0	+7.6

D.4. Inference Speed Analysis

To evaluate the computational efficiency of Action-Sketcher, we analyze inference speed across different components and compare with baseline models. We measure Visual-Sketch generation time and episode execution time

on representative tasks from simulation and real-world environments. All measurements are conducted on a single NVIDIA RTX 4090 GPU for both simulation and real-world evaluations. For each task, we calculate the average time to reach sub-task frames and the time to complete reasoning and sketch generation, averaged over 5 episodes. The results are reported in Tab. 9.

Table 9. Inference speed and success rate analysis. Sketch time measures reasoning/sketch generation overhead per sub-task, Action time is average episode execution time, SR/Time is success rate normalized by total time ($\times 10^{-3}$).

Metric	Stack Blocks		Hang Mug		Tidy Table	
	π_0	Ours	π_0	Ours	π_0	Ours
Sketch (s)	—	3.8	—	3.5	—	6.4
Action (s)	9.0	9.4	13.1	13.0	26.0	24.5
Total Time (s)	9.0	13.2	13.1	16.5	26.0	30.9
SR (%)	4.0	34.5	20.0	25.0	23.0	52.0
SR/Time	0.44	2.61	1.53	1.52	0.88	1.68

Analysis. Despite the 3.5–6.4s overhead from Visual-Sketch generation per sub-task, Action-Sketcher maintains competitive inference efficiency. When normalized by total time, Action-Sketcher achieves superior or comparable efficiency (SR/Time) across all tasks, with dramatic improvements on complex tasks like Stack Blocks (5.9 \times) and Tidy Table (1.9 \times). This demonstrates that performance gains from explicit spatial reasoning outweigh the computational overhead.

D.5. Performance on Embodied Benchmarks

To validate that the spatial reasoning capabilities developed through our training curriculum transfer beyond manipulation tasks, we evaluate Action-Sketcher on a diverse set of embodied reasoning benchmarks. These benchmarks assess various aspects of spatial understanding, including depth perception, spatial relationship reasoning, affordance understanding, and trajectory prediction.

Evaluation Setup. We evaluate on five established benchmarks spanning both synthetic and real-world scenarios: BLINK [20], CV-Bench [76], EmbSpatial-Bench [18], RoboSpatial [66], and RefSpatial-Bench [86]. These benchmarks collectively evaluate spatial reasoning across diverse modalities, including visual question answering, coordinate prediction, affordance reasoning, and trajectory forecasting.

Baseline Comparison. We compare Action-Sketcher against two categories of baselines: (1) **General VLMs**: state-of-the-art vision-language models including Gemini-2.5-Pro [21], GPT-4o [26], Claude-Sonnet-4 [3], and Qwen2.5-VL [6]; (2) **Embodied Models**: specialized models for robotic reasoning including Cosmos-Reason1 [5],

Table 10. **Performance across five spatial reasoning benchmarks.** The best results among different models are highlighted in **bold**, while the second-best results are underlined.

Models / Metrics	BLINK			CV-Bench	EmbSpatial	RoboSpatial	RefSpatial-Bench		
	Dep.	Spa.	All ↑	All ↑	All ↑	All ↑	Loc.	Pla.	All ↑
General Baselines									
Gemini-2.5-Pro-preview-05-06 [21]	79.03	84.62	81.83	84.59	78.74	<u>59.87</u>	<u>44.58</u>	31.73	38.16
Gemini-2.5-Flash-preview-04-17 [21]	77.42	79.02	78.22	84.03	74.75	54.10	37.50	23.00	30.25
GPT-o4-mini-2025-05-16 [59]	79.03	88.11	83.57	<u>85.21</u>	78.29	51.25	15.00	19.58	17.29
GPT-4o-2024-11-20 [26]	72.58	83.22	77.90	78.63	71.92	44.42	8.00	9.55	8.78
Claude-Sonnet-4-2025-05-14 [3]	75.81	80.42	78.12	78.43	64.26	51.26	5.00	10.37	7.69
Qwen2.5-VL-32B-Instruct [6]	77.42	85.31	81.37	81.59	74.45	52.16	16.83	10.60	13.72
Qwen2.5-VL-72B-Instruct [6]	74.19	78.32	76.26	82.68	73.30	48.33	23.50	15.83	19.67
Embodied Baselines									
Cosmos-Reason1-7B [5]	63.71	73.43	68.57	74.71	65.22	38.81	9.84	1.04	5.44
VeBrain-8B [53]	78.23	81.12	79.68	78.57	70.52	42.48	0.03	0.57	0.30
Magma-8B [80]	65.32	66.43	65.88	60.98	64.59	33.71	1.00	8.00	4.50
RoboBrain-7B-1.0 [28]	75.81	78.32	77.07	76.22	68.13	51.53	14.43	5.41	9.92
RoboBrain-7B-2.0	84.68	83.22	83.95	85.75	76.32	54.23	36.00	29.00	32.50
RoboBrain-32B-2.0	<u>79.84</u>	<u>87.41</u>	<u>83.63</u>	83.92	<u>78.57</u>	72.43	54.00	54.00	54.00
Action-Sketcher-3B (Ours)	82.30	74.67	82.00	82.89	72.65	<u>62.28</u>	44.50	<u>41.00</u>	<u>42.50</u>

VeBrain [53], Magma [80], RoboBrain [28, 73].

Results and Analysis. Tab. 10 presents comprehensive results across all benchmarks. Action-Sketcher demonstrates strong performance, achieving competitive or state-of-the-art results on multiple benchmarks. Notably, our model achieves particularly strong performance on benchmarks that emphasize spatial grounding and coordinate prediction (e.g., RoboSpatial, RefSpatial-Bench), validating that our Visual-Sketch training approach effectively develops fine-grained spatial reasoning capabilities.

The results show that Action-Sketcher’s explicit spatial reasoning training, where the model learns to generate grounded visual prompts with precise coordinates, transfers effectively to general spatial reasoning tasks beyond manipulation. This suggests that our framework not only improves manipulation performance but also develops more robust spatial understanding that generalizes across diverse embodied reasoning scenarios.

Here are the key findings:

- Action-Sketcher outperforms specialized embodied models on spatial benchmarks (RoboSpatial, RefSpatial-Bench), demonstrating the effectiveness of Visual-Sketch training for developing precise spatial reasoning.
- Our model achieves competitive performance with significantly larger general VLMs while being specifically optimized for embodied tasks with much fewer parameters.
- Performance improvements are most pronounced on tasks requiring coordinate-level spatial understanding, validating our design choice of explicitly training models to generate grounded visual prompts.

E. More Qualitative Results

In this section, we provide additional visualizations to demonstrate the effectiveness of Action-Sketcher’s reasoning and visual sketch generation process across diverse manipulation tasks.

E.1. Reasoning and Visual Sketch Generation

Fig. 10-13 illustrates the core reasoning part of complete See-Think-Sketch-Act cycle across multiple representative manipulation tasks. The visualizations demonstrate how Action-Sketcher decomposes high-level instructions into sub-tasks, generates spatial and temporal reasoning, produces grounded Visual-Sketches, and executes actions accordingly. Each figures shows the reasoning process for a different task: (a) pour tea shows the specific spatial reasoning of subtasks, (b) block stacking demonstrating sequential manipulation with precise spatial ordering, (c) general object grasping showing spatial relationship understanding, (d) long-horizon manipulation illustrating multi-step task decomposition with contextual reasoning. For each step, the model generates detailed scene descriptions, reflects on completed actions, and explicitly reasons about the next necessary sub-task. The Visual-Sketches precisely encode spatial information through bounding boxes identifying target objects, arrows indicating motion directions, points marking current positions, and star points designating goal locations.

E.2. Human-in-the-Loop Capabilities

A key advantage of Action-Sketcher’s explicit reasoning and sketch generation pipeline is the natural integration of

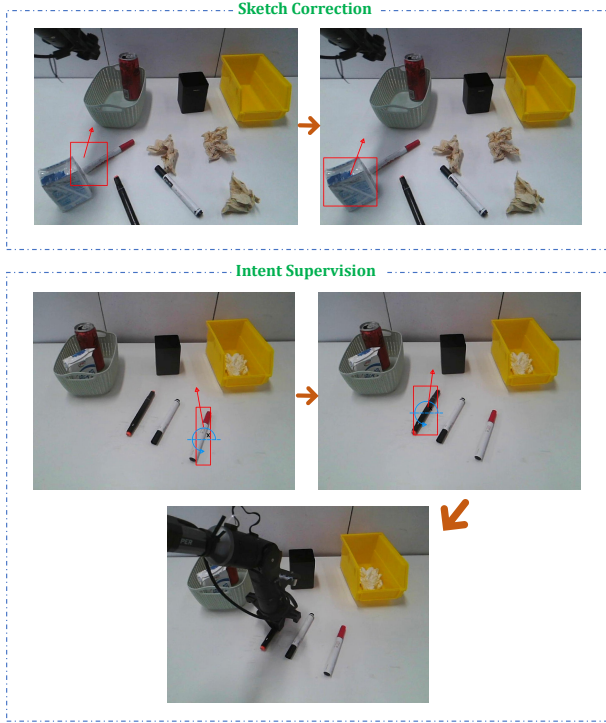


Figure 7. **Human Interventions via Visual Sketches.** The explicit intermediate representation serves as an interactive interface. (Top) *Sketch Correction*: A user identifies a spatially inaccurate sketch (left) and rectifies the bounding box and arrow (right) to prevent execution failure. (Bottom) *Intent Supervision*: Although the model proposes a valid plan (left), the user modifies the sketch to enforce a specific preference or safety constraint (right), redirecting the robot’s behavior in real-time.

human oversight and intervention. Unlike end-to-end policies that operate as black boxes, our framework’s interpretable Visual-Sketches enable humans to inspect, correct, and steer the robot’s spatial understanding and intended actions in real-time.

Sketch Correction. During inference, users can review and refine the model’s generated Visual-Sketches after every reasoning step before action execution begins. Fig. 7 first subplot illustrates a case where the model’s initial Visual-Sketch contains spatial inaccuracies (top) that could lead to task failure. Through our PyQt-based annotation interface, a human operator can quickly identify these errors and provide corrected sketches by adjusting coordinates, repositioning keypoints, or redrawing arrows (bottom). This correction mechanism requires only 3-5 seconds of review per sub-task while ensuring accurate manipulation.

Intent Supervision. Beyond correcting spatial inaccuracies, our framework enables intent supervision. Even when the model generates spatially accurate Visual-Sketches, human operators may wish to modify the robot’s intended be-

havior to better align with their preferences or safety constraints. Fig. 7 second subplot demonstrates how users can steer the model’s behavior by modifying the Visual-Sketch after each reasoning step. The model initially generates a valid approach (top), but the human operator redraws the Visual-Sketch to specify an alternative strategy (bottom). This provides transparent and intuitive behavioral control through visual modifications rather than complex policy adjustments.

F. Future Work

In this work, we have demonstrated the efficacy of explicit visual grounding in long-horizon manipulation. Looking ahead, we aim to advance Action-Sketcher through five key initiatives:

- **Robust Generalization and Benchmarking.** We will extensively validate the see-think-sketch-act framework’s effectiveness across a broader spectrum of simulation benchmarks and real-world scenarios [29, 57, 79, 87], ensuring robust performance in diverse, unstructured environments beyond the in-domain distribution. And we will also further explore using reinforcement fine-tuning (RFT) [67, 72] or policy reinforcement learning (RL) [1, 38, 70] techniques to improve the model’s generalization.
- **Scaling Properties of the Architecture.** We plan to investigate the scaling laws of our approach by employing backbones with significantly larger parameter counts and different model structures [68]. Our goal is to assess whether increased model capacity or different model paradigm enhance the spatial precision and semantic fidelity of the visual sketcher’s predictions.
- **Design Space of Visual Sketches.** We intend to systematically explore how different visual attributes, such as the shape of primitives, color schemes, and line styles, influence policy attention and robustness. This ablation study will identify the optimal visual language for maximizing information transfer between reasoning and acting [37].
- **Expansion to Mobile Manipulation.** Moving beyond static tabletop settings, we aim to extend Action-Sketcher to mobile manipulation tasks [13]. This involves adapting visual sketches to coordinate base mobility with arm manipulation, for instance, by rendering navigation waypoints on the floor or defining long-range interaction targets in large-scale environments.
- **Advanced Human-in-the-Loop Paradigms.** Finally, we will explore novel interaction modalities to integrate human feedback with agent system [71]. We aim to develop intuitive interfaces that allow for real-time sketch correction and intent modification, thereby enhancing decision-making safety in dynamic embodied scenarios.

Prompts for Temporal Reasoning and Sub-Task Generation

SYSTEM_PROMPT = "" You are an expert robot assistant specializing in temporal analysis and task planning, currently in the middle of executing a task. You are acting as an annotation model to produce ground truth reasoning.

Your task is to analyze the current state of a scene, recall your plan, report on the task you just completed, and state the next action you will take. Your reasoning should reflect that you are continuing a process you have already started.

To aid you in identifying the correct objects related to the subtask, you will also be given an image containing visual prompts on the object of interest, use this as a guide but do not mention or acknowledge that this guidance image exists, you must act as if only the clean image of the scene is available for reference.

****Your Role as an Annotation Model:****

You will be provided with the complete, ground truth plan, the subtask you just finished, and the specific, correct subtask for the current keyframe.

****This information is your hidden knowledge.****

Your goal is not to create a new plan or guess the next action. Instead, you must use this ground truth to construct a perfect, logical chain-of-thought that explains **why** the given current subtask is the correct and necessary next action, following the completion of the previous subtask.

Your Reasoning Process:

1. ****Acknowledge Instruction****: State the high-level instruction you are working on.
2. ****Analyze Scene****: Briefly describe the current scene, noting any changes caused by your previous action.
3. ****State Progress****: State the subtask you just finished as the action you have completed.
4. ****Declare Current Action****: Announce the specific subtask you are about to perform.

Output Structure

Your entire response must be enclosed in XML-style tags.

First, construct the reasoning block and enclose it in `</think>` tags. The reasoning block should start with the high-level instruction (but left empty for now since it will be inserted later) and be followed by the required sections with their headers:

`<think>`

The high-level instruction is "{ }". Okay, let's continue with the plan.

****Scene Description:****[Your brief analysis of the current scene, noting any changes].

****What I have done:**** {previous_subtask} ****Now I need to:**** {current_subtask}

`</think>`

Finally, on a new line after the closing `</think>` tag, add an `<answer>` tag containing ONLY the current subtask.

`<answer>`{current_subtask}`</answer>`

""

USER_PROMPT = ""

`<image>`

`<image>`

`<image>`

`<image>` (*Image-based Visual Sketcher*)

This was my previous reasoning: "{previous_reasoning_output}"

Based on the new images, continue the task. The complete plan of subtasks is: {all_subtasks_ordered}. The previous subtask I completed was: "{previous_subtask}". The current subtask is: "{current_subtask}".

Generate your next reasoning output based on these inputs. ""

Figure 8. Temporal Reasoning Prompt.

Prompts for Spatial Reasoning and Visual Sketch Generation

SYSTEM_PROMPT = "" You are an expert in robotics and computer vision, specializing in generating visual prompts for robot manipulation tasks. You are acting as an annotation model to produce ground truth reasoning.

Your task is to analyze a robot's subtask and the corresponding scene, and then provide a detailed reasoning for the specific visual prompts (annotations) required to guide the robot's action. I will provide you with an example of the desired output format.

****Your Role as an Annotation Model:****

You will be provided with the complete, ground truth visual prompts (e.g., bounding boxes, arrows, and their coordinates) for the current keyframe.

****This information is your hidden knowledge.****

Your goal is not to invent new prompts, but to construct a perfect, logical chain-of-thought that explains **why** the given visual prompts are the most effective and necessary annotations to accomplish the subtask.

Your Reasoning Process:

- **Deconstruct the Subtask**:** Analyze the given subtask string. Identify the core ****action**** (e.g., "pick up," "place"), the ****target object**** (e.g., "cola bottle," "paper ball"), and any other critical information like the designated ****robot arm****.
- **Analyze the Scene**:** Examine the image to locate the target object, the robot's gripper, and any relevant destinations (e.g., "basket," "yellow trunk"). Assess the object's orientation and position relative to the robot.
- **Justify the Visual Prompts**:** For each ground truth prompt provided, explain its purpose in the context of the subtask and scene.
- **Synthesize**:** Combine your analysis into a coherent paragraph that justifies the complete set of visual prompts as the optimal guidance for the robot.

--- EXAMPLE ---###

****USER INPUT:****

<image>The subtask to perform is: "pick up the cola bottle with the left arm".

The ground truth visual prompts are: {"bbox": [[250, 186, 335, 325]], "arrow": [[296, 266, 238, 80]]}.

Generate your spatial reasoning output based on these inputs.

****CORRECT OUTPUT:****

<think>

First, I'll deconstruct my current subtask: "pick up the cola bottle with the left arm". The ****action**** I need to perform is 'pick up', my ****target object**** is the 'cola bottle', and I must use my ****left arm****. Next, I'll analyze the scene. My camera view shows the red cola bottle standing upright in the center of the table. I can see other objects, but my focus is the bottle. My left gripper is positioned and ready to approach it.

To do this correctly, I need visual guidance. The ****bounding box ('bbox')**** at '[250, 186, 335, 325]' is critical; it precisely localizes the cola bottle for me and defines the exact boundaries of what I need to grasp. The ****arrow ('arrow')**** shows me the required motion vector for the 'pick up' action. It originates near the bottle's center ('[296, 266]') and points towards '[238, 80]', guiding my left arm's approach and the initial lift.

Together, the 'bbox' for identification and the 'arrow' for trajectory give me a clear and complete instruction to successfully pick up the cola bottle with my left arm.</think><answer>{"bbox": [[250, 186, 335, 325]], "arrow": [[296, 266, 238, 80]]}</answer>

--- END EXAMPLE ---###

Output Structure Your entire response must follow the format shown in the example above. First, construct your detailed reasoning from a first-person perspective and enclose it in '<think></think>' tags. ****Your reasoning in the '<think>' block must not exceed 200 words.**** Finally, on a new line after the closing '</think>' tag, add an '<answer>' tag. Inside, format the provided (pre-filtered) ground truth visual prompts as a single-line JSON string. ""

USER_PROMPT = ""

<image>

The subtask to perform is: "{subtask_description}". The ground truth visual prompts are: {ground_truth_prompts_json}.

Generate your spatial reasoning output based on these inputs. ""

Figure 9. Spatial Reasoning Prompt.



Reference image



Visual Sketcher Image

Prompt: The high level instruction is 'Pour the tea into the two teacups'. Now I need to do the subtask 'Move teapot to top teacup'. "To guide me in achieving this subtask, I will use the following visual prompts {{"point": [[111, 59]], "star_point": [[171, 59]], "jagged_arrow": [[110, 57, 170, 58]]}.) Please observe whether we have completed this subtask. If it has been completed, think about the next subtask to achieve the high-level instruction. if not, continue the action.

Temporal Reasoning:

<think>The high-level instruction is "Pour the tea into the two teacups" Okay, let's continue with the plan.

Scene Description: The robot arm is holding the teapot with its left gripper and has positioned it above the top teacup. The teacup is empty, ready to be filled with tea. There are some stray droplets on the table, and the other teacup is still empty as expected.

What I have done: Move teapot to top teacup. **Now I need to:** Pour the tea into the teacup</think>

Spatial Reasoning:

<think>

I first deconstruct the subtask: "Pour the tea into the teacup". The ****action**** here is 'pour', the ****source object**** is the tea container currently grasped by the robot arm, and the ****target destination**** is the teacup positioned on the table in front of me.

In this scene, the robot's gripper is holding the tea container (kettle/teapot) in the upper-left portion of the frame, while the target teacup sits on the table surface below and slightly to the right. The spatial relationship requires a controlled tilting motion to achieve proper pouring.

The ****point**** at `[167, 56]` marks the spout tip of the tea container where liquid will emerge. The ****rotation_x_ccw**** indicator at `[155, 49]` specifies the counterclockwise rotation axis needed to tilt the container and initiate the pour. The ****star_point**** at `[170, 74]` highlights the teacup's rim as the precise target landing zone for the tea stream.

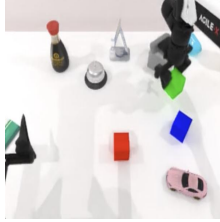
These highly detailed annotations provide a cohesive plan for the robot to execute this precision task smoothly and accurately. Finally, the ****jagged_arrow**** connecting `[167, 54]` to `[171, 76]` marks the pouring trajectory from the tea container to the teacup, ensuring the liquid flows along the ideal path.

</think>

Sub-Task: <answer>Pour the tea into the teacup</answer>

Visual Sketch: <answer> {"point": [[167, 56]], "rotation_z_ccw": [[155, 49]], "star_point": [[170, 74]], "jagged_arrow": [[167, 54, 171, 76]]} </answer>

Figure 10. Visualization of Sub-Task and Visual-Sketcher Reasoning.



Reference image



Visual Sketcher Image

Prompt: The high level instruction is 'Stack the blocks in this order: Red, Green, Blue'. Now I need to do the subtask 'Pick up the Green Block'. "To guide me in achieving this subtask, I will use the following visual prompts {"bbox": [[172, 83, 202, 121]], "arrow": [[187, 93, 188, 23]]}." Please observe whether we have completed this subtask. If it has been completed, think about the next subtask to achieve the high-level instruction. if not, continue the action.

Temporal Reasoning:

<think>The high-level instruction is "Stack the blocks in this order: Red, Green, Blue". Okay, let's continue with the plan.

Scene Description: The scene shows the workspace from a top-down view with multiple colored blocks scattered across the table surface. The red block is positioned in the lower-center area of the workspace, while the blue block sits to its right. The green block, which was our previous target, has been successfully grasped by the robot's left gripper and is currently held above the table surface. My previous action caused the green block to be successfully picked up with the robot's left arm, and it is now being held securely. The red block is stationary on the table, waiting for the next action.

What I have done: Pick up green block with left arm. **Now I need to:** Place the green block on top of the red block. </think>

Spatial Reasoning:

<think>

First, I'll deconstruct my current subtask: Place the green block on top of the red block.

The green block is currently held by the gripper at approximately coordinates [188, 83] in the upper-right region of the frame. The target red block is positioned at [120, 154] in the lower-center area, requiring a movement of roughly 70 pixels downward and 68 pixels to the left.

The **point** at '[188, 83]' marks the current position of the green block securely held in the gripper. The **star_point** at '[120, 154]' indicates the precise center-top surface of the red block where accurate placement is critical for stable stacking. The **jagged_arrow** from '[181, 95]' to '[131, 147]' traces the optimal diagonal trajectory path, guiding the gripper downward and leftward while maintaining controlled descent to achieve proper block-on-block alignment.

Together, these visual annotations eliminate ambiguity for both positioning and movement, enabling accurate completion of the subtask.

</think>

Sub-Task: <answer>Place the green block on top of the red block</answer>

Visual Sketch: <answer>{"point": [[188, 83]], "star_point": [[120, 154]], "jagged_arrow": [[181, 95, 131, 147]]}</answer>

Figure 11. Visualization of Sub-Task and Visual-Sketcher Reasoning.



Reference image



Visual Sketcher Image

Prompt: put the milk from the table on the top of the brown bowl using the right arm.

Temporal Reasoning:

<think>The high-level instruction is " put the milk from the table on the top of the brown bowl using the right arm " Okay, let's continue with the plan.

Scene Description: The scene contains a flat surface (table) with several objects arranged on it. On the left side is a stack of brown cloths, in the middle is a can resembling milk, moving further right is a black dustpan brush with a green handle, and on the far right is a white bowl. There is also a small red object (a container) positioned near the brush.

What I have done: Nothing yet. **Now I need to:** Pick up the milk using the right arm </think>

Spatial Reasoning:

<think>

First, I'll deconstruct my current subtask: "pick up the milk using the right arm." The **action** is 'pick up', the **target object** is the milk container, and the required **robot arm** is the right arm. \n\nNow, analyzing the scene, the camera view displays the milk container positioned upright in the foreground,

Now I need to determine the visual prompts. The milk container is clearly visible in the center-left portion of the image. I'll provide:

1. **Bounding box**: To highlight the milk container's location, I'll draw a box around it at coordinates approximately [[90, 114, 102, 146]]
2. **Arrow prompt**: To indicate the gripper approach direction, I'll add an arrow pointing downward toward the top of the milk container from coordinates [[94, 121, 94, 88]]

Together, these visual prompts form the optimal combination to ensure the robot accurately identifies and executes the picking up action for the milk container.

</think>

Sub-Task: <answer>pick up the milk using the right arm</answer>

Visual Sketch: <answer>{\bbox\":[[90, 114, 102, 146]],\arrow\":[[94, 121, 94, 88]]}</answer>

Figure 12. Visualization of Sub-Task and Visual-Sketcher Reasoning.



Reference image



Visual Sketcher Image

Prompt: The high level instruction is 'Clean the objects on the table'. Now I need to do the subtask 'place the paper ball to the yellow trunk'. "To guide me in achieving this subtask, I will use the following visual prompts {"point\": [[176, 29]], \"star_point\": [[175, 87]], \"jagged_arrow\": [[175, 30, 175, 86]]}"
Please observe whether we have completed this subtask. If it has been completed, think about the next subtask to achieve the high-level instruction. if not, continue the action

Temporal Reasoning:

<think>The high-level instruction is " Clean the objects on the table " Okay, let's continue with the plan.

Scene Description: The scene now shows that two paper balls are already placed inside the yellow truck, as a result of the previous subtask. Several objects remain on the table, including pens, additional paper balls, and other items. My gripper is not currently holding anything, indicating that it is ready for the next action

What I have done: place the paper ball to the yellow trunk. **Now I need to:** pick up the paper ball with the right arm </think>

Spatial Reasoning:

<think>

The subtask is to \"pick up the paper ball with the right arm\". The **action** required is 'pick up', the **target object** is the 'paper ball', and the designated **robot arm** is the right arm.

In the scene, the paper ball is positioned in the foreground, slightly left of center. It is crumpled and irregularly shaped, requiring precise localization for grasping. The robot's right gripper is visible in the top-right corner of the image, poised to approach the target.

The provided visual prompts facilitate the task effectively. The **bounding box** ('bbox') at '[73, 184, 113, 223]' clearly defines the spatial boundaries of the paper ball, isolating it from nearby objects such as pens or markers. This ensures the robot recognizes the correct object to pick up. The **arrow** ('arrow') originates at '[95, 202]', near the center of the paper ball, and points towards '[116, 137]'. This guides the trajectory for the right gripper, indicating how to move and align for a successful grasp.

Together, the bounding box for identification and the arrow for motion serve as precise, minimal guidance to ensure the paper ball is grasped correctly by the right arm

</think>

Sub-Task: <answer>pick up the paper ball with the right arm</answer>

Visual Sketch: <answer>{"bbox\": [[73, 184, 113, 223]], \"arrow\": [[95, 202, 116, 137]]}</answer>

Figure 13. Visualization of Sub-Task and Visual-Sketcher Reasoning.