Appendix

In the appendix, we provide additional details of this work. In Section A, we introduce the robot and teleoperation platform, along with more qualitative results. In Section B, we provide further details about our *MM-CoS* dataset, including data distribution and the automatic labeling pipeline. In Section C, we present additional quantitative results on efficiency under `Share Control`. In Section D, we describe the implementation details.

A. Real-World Experiments

A.1. Evaluation Platform

We evaluate the instruction-following performance of our model on a wheeled robot developed by Coco Robotics. As illustrated in Figure 8, the robot platform includes both the onboard robotic infrastructure and a teleoperation interface for monitoring and control. Notably, during testing, the inference computer is placed inside the robot's storage compartment. For the onboard robotic system, we primarily use the front fisheye camera for perception. The raw fisheye images are undistorted before being passed to the inference computer. The robot is connected to the inference computer through an Ethernet cable, and the inference computer outputs the desired linear and angular velocities generated by a PD controller, which are then sent to the robot executor. The robot performs localization using odometry based on IMU and GPS fusion. Communication with the teleoperation station is established through a 4G router. The teleoperation station consists of a computer equipped with a joystick, mouse, and keyboard. The mouse and keyboard are used for issuing instructions, while the joystick is reserved for full manual control.

A.2. More Visualization Results.

We present additional real-world results in Figure 18, showcasing three cases for each type of instruction. AURA demonstrates robust performance in real-world navigation tasks, including path selection, lane recovery, and obstacle avoidance. The captions under the images describe the robot's actions. The first three rows illustrate that the robot can accurately interpret geo-information under drafting guidance. Rows four to six show that the robot can follow arrowing instructions, effectively performing obstacle avoidance, lane recovery, and path selection. Finally, rows seven to nine demonstrate that our model can follow high-level textual instructions provided by a human.

B. Details about *MM-CoS*.

B.1. *MM-CoS*

Finally, we generate 29K annotations for our *MM-CoS* dataset from 50 hours of teleoperation data. Figure 9 illustrates the data distribution. The frequencies were calculated based on keyword occurrences, and the data are categorized by action mode and interaction behavior. Each major category contains specific subcategories, which may overlap; for example, a single scene could include both "Straight_Movement" and "Turning_Left" at the same time. The percentage values shown on each bar indicate the proportion of the dataset corresponding to that behavior. The weather distribution and time of day are illustrated in Figure 11. The *MM-CoS* dataset (50 hours) contains 3,040 videos, covering a wide range of weather and lighting conditions. Figure 10 illustrates the diversity of the *MM-CoS* dataset, covering variations in lighting, weather, and sidewalk scenes. Such data better reflect real-world scenarios and present significant challenges.

In addition, the *MM-CoS* dataset is further augmented with 18K annotations from open-source datasets [4, 29, 48].

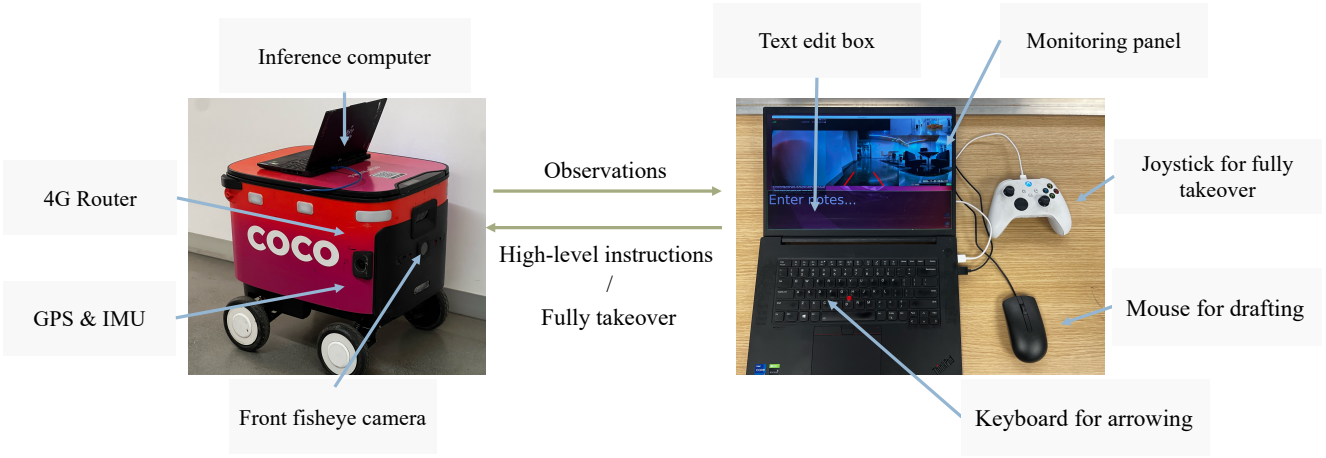
B.2. Auto labeling pipeline

As illustrated in Figure 12, we employ a two-stage selection strategy to identify informative frames for annotation. This approach combines VLM-based assessment with trajectory-based motion analysis, corresponding to the "behavior and state filtering" step shown in the figure.

First, we use a pre-trained VLM (InternVL3-8B [68]) to classify each video segment as either "interesting" or "boring" based on scene complexity, such as pedestrian interactions, obstacles, and terrain changes. This produces a temporal interestingness map $\mathbf{I} = [i_1, i_2, \ldots, i_n] \in 1, 2^n$ for the n segments of each clipped trajectory, where $i_j = 1$ denotes an interesting segment and $i_j = 2$ denotes a boring one. The prompts used for this classification are shown in Figure 13.

Second, we perform motion analysis on the robot trajectory using a sliding window approach. For each window, we compute acceleration and turning scores as weighted combinations of motion statistics: $S_{accel} = w_1\alpha_{avg} + w_2\alpha_{max} + w_3\sigma_\alpha^2$ and $S_{turn} = w_4\theta_{avg} + w_5\theta_{max} + w_6\sigma_\theta^2$, where α and θ denote acceleration and turning angle respectively. To prioritize semantically meaningful scenes, we apply priority-based weighting: $S' = w_p \cdot S$ where $w_p = w_{\text{int}} > 1$ for interesting segments, $w_p = w_{\text{bor}} < 1$ for boring segments, and $w_p = 1$ otherwise. We then rank windows by their weighted scores and select the top-k frames with high acceleration or turning, ensuring temporal diversity by filtering adjacent candidates.

Once keyframes are identified (Figure 12), we construct the VLM inputs by overlaying the robot's future trajectory and frame-wise speed cues on the front-view images. With the prompts in Figures 14 and 15, the VLM generates (i) a short command-style instruction and (ii) a detailed description of the underlying reasons. In parallel, we derive geometric/control supervision (drafting and arrowing) from the same trajectory and speed signals; formal definitions are provided in Section B.3. Notably, for all appendix prompts, the purple text explains the input signals and the pink text specifies rules to follow so that the output satisfies the requirements shown in blue.



(a) A wheeled robot developed by Coco Robotics

(a) Teleoperation station

Figure 8. **Overview of the real-world experiment setup.** Left: the robot hardware platform; Right: the teleoperation interface.

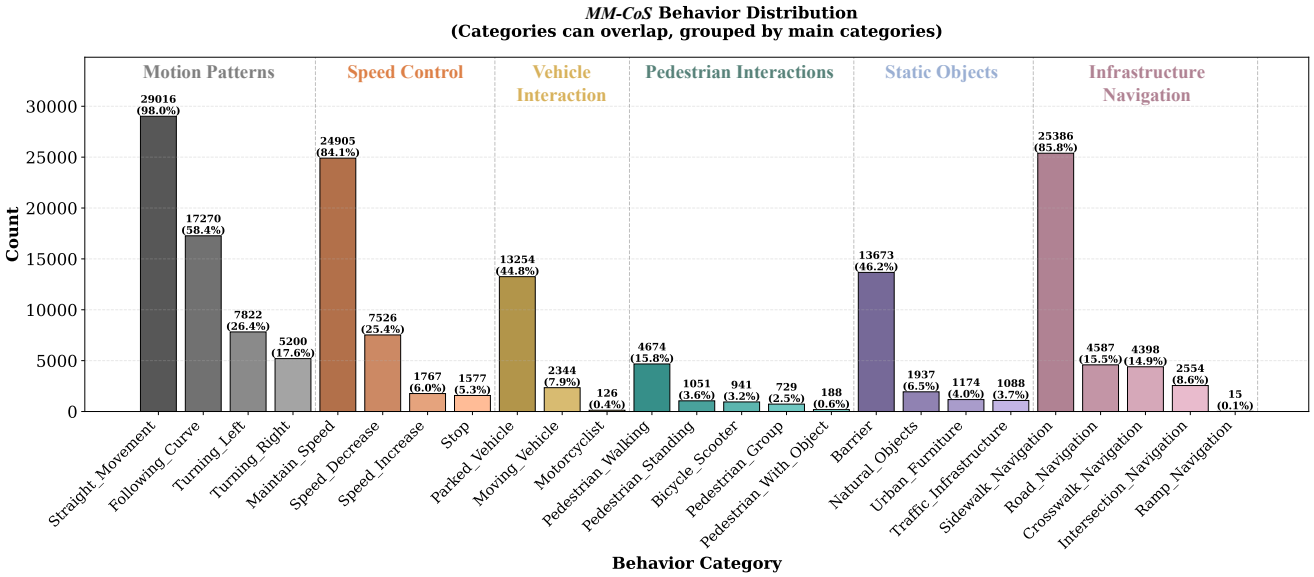


Figure 9. **Data distribution of our *MM-CoS* dataset (50h)**. Behaviors are divided into six subclasses based on their outcomes and causes. Bars of the same color represent different specific actions within each subclass. The top of each bar indicates the number of occurrences of the behavior and its proportion of the total annotations.

B.3. Multi-Modal Annotation Generation

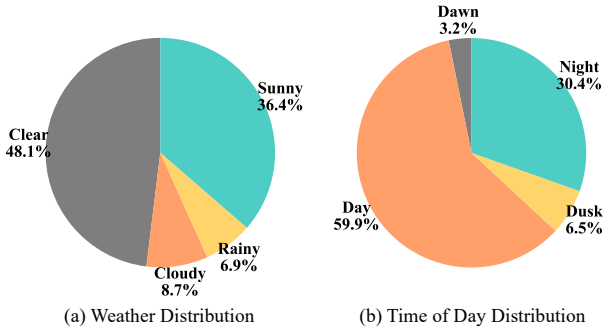
For each selected frame, we generate three complementary types of annotations to provide rich supervisory signals for navigation learning.

CMD `Drafting.` We project the robot's planned 4-second future trajectory from 3D world coordinates onto the 2D image plane using a pinhole camera model with intrinsic parameters $[f_x, f_y, c_x, c_y]$ and the corresponding 6-DoF camera poses. The projected waypoints are rendered as green trajectory lines overlaid on the RGB image $\boldsymbol{I}_d$, showing the intended path relative to the observed scene. Along this line, we uniformly sample n pixel points $\boldsymbol{p}_d$ to obtain the visual instruction $C_d = (\boldsymbol{I}_d, \boldsymbol{p}_d)$.

CMD `Arrowing.` We compute the average speed and directional control signals from the trajectory waypoints. Specifically, given the first N waypoints $\{\mathbf{p}_i\}_{i=1}^{N}$, the average velocity vector is calculated as $\mathbf{v}_{\text{avg}} = \frac{1}{N-1}\sum_{i=1}^{N-1}(\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \text{fps}$, and the corresponding heading angle as $\theta = \arctan 2(v_y, v_x)$. The velocity vector $\mathbf{v}_{\text{avg}}$ is then projected onto the front-view camera image and visualized as a fixed-length arrow indicating direction, with the absolute speed value overlaid as text, producing the image $\boldsymbol{I}_s$. The resulting arrowing instruction is defined as $C_s = (\boldsymbol{I}_s, \mathbf{v}_{\text{avg}})$.

CMD `Texting.` We employ a pre-trained vision-language model, Qwen2.5-VL [8], deployed with vLLM [32] for efficient infer-

Figure 10. **A thumbnail montage showing a subset of the *MM-CoS* videos collected from real world environments for shared autonomy.**



(a) Weather Distribution (b) Time of Day Distribution

Figure 11. **Data distribution across weather conditions and time of day.** The dataset contains 3,040 scenes in total, with weather conditions primarily consisting of clear (48.1%) and sunny (36.4%) conditions, while cloudy (8.7%) and rainy (6.9%) conditions are less represented. Time-of-day distribution shows a strong bias toward daytime scenes (59.9%), followed by nighttime scenes (30.4%), with transitional periods of dawn (3.2%) and dusk (6.5%) being relatively scarce.

ence, to generate two levels of language supervision: (i) a short command-style instruction (verb phrase) describing the maneuver (e.g., "go straight", "slow down", "speed up"), and (ii) a more detailed natural-language description used to supervise reason-

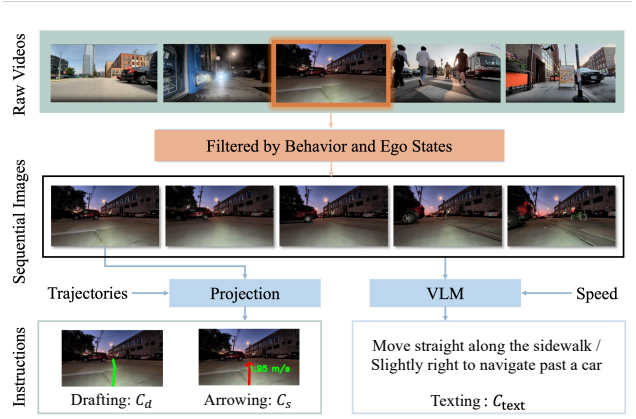


Figure 12. **Auto labeling pipeline.** We first select interesting clips based on behavioral cues and ego state information, focusing on frames that show rich interactions with objects or the environment. We then feed the front view images with visual prompts of the projected future trajectory and the speed of each frame into the VLM to generate textual instructions. Finally, we use the projected future trajectory on the front images to derive the drafting and arrowing instructions.

Behavior filtering prompt

```
You are an expert video analyst. Your task is to analyze a 10-second video segment from a delivery robot's first-person perspective in an urban environment.

The video shows the robot navigating through city streets. Please determine if this 10-second segment is "interesting" or "boring" based on the following criteria:

INTERESTING scenes include:
- Interactions with pedestrians (avoiding, following, or being near people)
- Interactions with objects (obstacles, traffic signs, vehicles, etc.)
- Complex terrain navigation (stairs, ramps, uneven surfaces)
- Traffic situations (crossing streets, waiting at lights, etc.)
- Environmental interactions (ground cracks, curb/sidewalk transitions, walkable area changes)
- Dynamic obstacles or unexpected situations

BORING scenes include:
- Simple straight-line walking on sidewalks
- Repetitive movement without interactions
- Just following a path without any challenges

Please respond with only "1" for interesting or "2" for boring.
```

Figure 13. **Prompt for behavior filter.** We use this prompt to let the VLM determine whether the robot in the clip is interacting with objects or the environment.

ing about the underlying scene and interactions. We sample future observation frames covering 4 seconds and overlay the corresponding 4-second trajectory visualizations on each frame. The VLM processes these temporally ordered frames together with frame-wise speed measurements to produce the instruction and the detailed description, covering directional changes, speed adjustments, obstacle avoidance, and interactions with the environment. These language annotations provide high-level semantic insights into navigation behaviors, complementing the geometric and control-level annotations.

TEXT annotation prompt

```
"The speeds of each frame are: {:2f} m/s, {:2f} m/s, {:2f} m/s, {:2f} m/s, {:2f} m/s.\n\n"
"Please describe the robot's actions as short, clear movement phrases. Follow these rules:\n\n"

"Rules:\n"
"- Use short phrases like 'go straight', 'turn left', 'stop at crosswalk'.\n"
"- Mention direction changes, stops, or speed changes only if significant.\n"
"- Focus on the path the robot is traveling on (road, sidewalk, crosswalk, bike lane).\n"
"- Do not include background scenery or unobserved actions.\n\n"

"Examples:\n"
"- 'go straight along the sidewalk.'\n"
"- 'turn right at the corner.'\n"
"- 'go through the intersection.'\n"
"- 'stop at the crosswalk.'\n"
"- 'speed up to pass the slow bicycle.'\n"
"- 'slow down.'\n"
"- '...'\n"
```

Figure 14. **Prompt for generating texting instructions.** We expect the VLM to produce short, simple instructions that guide the robot at a high level.

C. Details in Shared Control

C.1. The Judgment Module.

As shown in Figure 16, we employ both rule-based and VLM-based criteria to determine whether a human takeover is required.

Reason annotation prompt

```
"The green line shows the 4-second future trajectory the human wants the robot to follow."
"and the white transparent polygon around it indicates the ego vehicle's occupancy.\n\n"
"The speeds of each frame are: {:2f} m/s, {:2f} m/s, {:2f} m/s, {:2f} m/s, {:2f} m/s.\n\n"
"Please provide a detailed first-person description of the robot's actions and movements, following these rules:\n\n"

"**Action & Environment (REQUIRED):**\n"
"- Describe movements in temporal order: e.g., 'first moved straight, then turned left'.\n"
"- Only mention objects/obstacles that the robot DIRECTLY interacts with or avoids: "
"'steered around a parked car', 'navigated past a pedestrian', 'stopped for a red light'. "
"Do NOT describe background objects that don't affect the robot's path.\n"
"- Only include purposeful directional changes: 'turned left to follow the sidewalk', 'turned right to avoid a pedestrian'. Ignore minor adjustments.\n"
"- Describe movement patterns: straight, curved, stopped.\n"
"- Mention path transitions only if the robot actually moves between different surface types.\n"
"- For speed changes: Only describe significant changes with clear reasons, such as "
"'slowed down approaching pedestrians', 'stopped at the red traffic light'.\n"
"- Focus on the path surface the robot is actually traveling on (road, sidewalk, crosswalk, bike lane). "
"Only mention other environmental elements if they directly influence the robot's movement.\n\n"

"**Formatting Rules:**\n"
"- Output 1–2 continuous sentences, no bullet points.\n"
"- Describe only observable movements from the robot's first-person perspective.\n"

"**Important Notes:**\n"
"- Focus on concrete, observable actions rather than general movement terms\n"
"- Be as specific as possible about directions, distances, and spatial relationships\n"
"- Avoid making assumptions about speed changes unless clearly observable\n"
"- CRITICAL: Only describe path transitions (sidewalk/crosswalk/road changes) if you can clearly see the robot's trajectory actually crossing from one surface type to another. Do NOT assume transitions just because these elements are visible in the scene.\n"
"- CRITICAL: Only describe turns that have a clear purpose (following road geometry, avoiding obstacles, navigating intersections). Do NOT describe random minor directional adjustments.\n"
"- For sidewalk navigation: specify turn direction only when there's an actual turn ('turned right to continue along the sidewalk'), but 'continued along the sidewalk' is fine for straight movement\n"
"- Describe from FIRST-PERSON perspective - what the robot actually did, not what exists in the environment\n"
"- CRITICAL: When describing left/right turns, use the ego vehicle's perspective (robot's own left/right), NOT the left/right relative to other objects in the scene\n"
"- CRITICAL: Determine whether vehicles/objects are stationary or moving by observing their consecutive positions across the video frames, not just their appearance in a single frame\n"
"- Do NOT mention the green trajectory line or white polygon in your description - describe only the robot's actual movements and environment"
"- Some actions have no reason other than the human controller's intention. In such cases, simply describe the action without explaining a reason."·
```

Figure 15. **Prompt for generating reasoning.** We expect the VLM to generate a detailed description of the robot's behavior in the video, and to provide supervision signals (drafting and arrowing) for interpreting human instructions.

There are two primary situations that trigger a takeover. The first is collision risk. Specifically, we project the predicted trajectory together with the ego vehicle's occupancy polygon onto the front camera view to obtain a mask M_p. Using OpenSeed [66], we segment all pixels corresponding to impassable regions (such as obstacles or walls) as mask M_f. If the overlap between M_p and M_f exceeds 10% of M_p, the system flags the frame as requiring a takeover. The second is instruction compliance. To assess whether the predicted trajectory aligns with the human's original instructions, we input two rendered images, one with the predicted trajectory and one with the ground-truth trajectory, into QwenVL2.5-72B [8]. The model evaluates whether the prediction follows the intended goal, for example when the agent deviates from the cor-

Table 4. **Pseudo simulation evaluation on share control**.

	HO% ↓	TF ↓	OF ↓	HO_{1s}% ↓	TF_{1s} ↓	OF_{1s} ↓	HO_{2s}% ↓	TF_{2s} ↓	OF_{2s} ↓	HO_{3s}% ↓	TF_{3s} ↓	OF_{3s} ↓
MBRA [22]	15.1	0.757	0.757	30.5	0.305	1.523	41.6	0.211	2.082	48.1	0.164	2.407
CityWalker* [36]	19.6	0.978	0.978	40.3	0.404	2.016	54.5	0.276	2.725	60.7	0.207	3.037
AURA	6.0	0.298	0.298	14.9	**0.151**	0.744	22.4	**0.116**	1.122	28.0	**0.098**	1.400
AURA	7.2	0.359	0.359	9.0	0.195	0.450	12.5	0.170	0.624	15.0	0.152	0.751
AURA	5.0	0.248	0.248	6.4	0.169	0.320	9.1	0.152	0.456	11.2	0.144	0.562
AURA	**4.5**	**0.226**	**0.226**	**5.7**	0.163	**0.284**	**7.7**	0.150	**0.386**	**9.7**	0.142	**0.483**

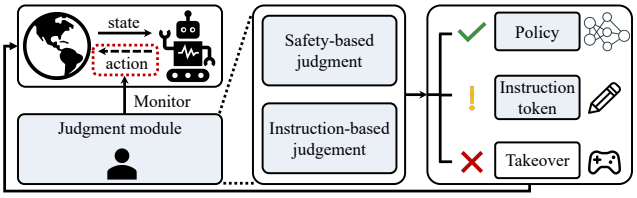


Figure 16. **Judgment module.** After the policy updates its state and receives the observation, it produces a candidate action through a forward pass of the model, which is temporarily cached before execution. We then apply a judgment module that evaluates the action using both safety-based criteria and instruction-based constraints. Based on this evaluation, the system decides whether to (i) execute the action, (ii) inject an instruction token to guide the policy toward human instruction, or (iii) trigger a full human takeover when the candidate action violates critical safety conditions.

rect path or remains stationary while the human intends to move.

C.2. More Results

These experiments evaluate how much human intervention time can be reduced during teleoperation. To ensure reproducibility, we conduct all evaluations in a pseudo-simulation environment. We select 29 scenarios from the test set in which every clip is annotated as *interesting*.

Additional results on Human Cost Evaluation in pseudo-simulation are shown in Table 4. Here, HO denotes the human operation ratio, defined as the percentage of time under human control. TF represents the takeover frequency, calculated as the total number of takeover events divided by the total duration. OF indicates the operation frequency, defined as the number of operation actions per unit time. The first three columns without footnotes assume that human takeover occurs instantaneously. In our actual testing setup, the system runs at 5 Hz, meaning that a full takeover requires 0.2 seconds of human control. The footnoted columns report results under the assumption that each full takeover lasts 1, 2, or 3 seconds.

The results indicate a clear trend: as humans intend to sustain longer takeover periods, the total amount of required intervention decreases. Notably, when the human remains in control for longer durations, more potential takeover events are absorbed within that interval and therefore do not trigger additional interventions.so the takeover frequency are reducing while the human takeover time increase.

Judgement module prompt

```
"You are a delivery robot safety judge. Based on the following images:\n\n"
"1. Predicted trajectory from the autonomous system (first image).\n"
"2. Reference trajectory from the human operator (second image).\n\n"
"Green line = future robot path.\n"
"White polygon = robot occupancy at each future pose.\n"
"Red points with timestamps = future positions.\n"
"If no green line/polygon, the robot is stopped (no collision risk).\n\n"

"Decide if the human operator should take over.\n\n"
"Rules:\n"
 "- Predicted trajectory should generally follow the reference direction and path, though length may differ.\n"
"- Predicted trajectory should generally follow the reference route, though timestamp and the end point may differ.\n"

"- A takeover is required if:\n"
" • White polygon overlaps obstacles, pedestrians, or vehicles.\n"
" • Trajectory goes into invalid areas (buildings, roads, restricted zones).\n"
" • Trajectory clearly deviates from the reference path (e.g., wrong turn, different branch).\n"
" • Predicted motion is unsafe (abrupt turns, unsafe proximity, excessive speed).\n"
"- Minor deviations in shape or length are acceptable if overall direction/path is consistent.\n"
"- Judge only on the given prediction. Do not imagine unseen events.\n\n"

"Answer with one word: 'YES' (takeover required) or 'NO' (not required)."
```

Figure 17. **Prompt for the VLM judge.** The judge module is required to output only yes or no. It checks whether the predicted trajectory matches the human-intended action. If the robot's intent diverges from the human intention, human intervention is needed. For example, the robot may take an incorrect path or make unnecessary stops.

D. Implementation Details

Our training pipeline consists of two stages. For stage 1, we train the SIE and the LoRA-adapted LLM to generate action captions conditioned on the drafting and the arrowing prompt. The modules are trained on eight A6000 GPUs using a cosine learning rate schedule with a base learning rate of 2×10^{-5}, weight decay of 0.05, and a warmup ratio of 0.03, for 5 epochs with a global batch size of 32. For stage 2, we freeze all other components and train only the action decoder in an end-to-end manner. The policy is trained on eight RTX 4090 GPUs using the AdamW optimizer with a learning rate of 1×10^{-4} for 100 epochs and a global batch size of 32. The AdamW hyperparameters are set to $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

Figure 18. **Qualitative results in the real world.** The first three rows are guided by drafting, rows 4 to 6 by arrowing, and the last three rows by texting. The first column shows human instructions, while columns 2 to 5 show the actions executed by the robot.