UNIWAM: UNIFIED MOBILE MANIPULATION VIA MIXED-STREAM WORLD-ACTION MODELING AND MANIPULATION ANCHOR POSE SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mobile manipulation requires precise navigation to a manipulation-ready pose followed by reliable object interaction. These two stages differ in action spaces and visual requirements, which complicates unified policy learning. In addition, collecting diverse real-world navigation data with explicit manipulation-ready pose supervision remains costly and difficult to scale. We introduce UniWAM, a unified mixed-stream world-action model with separate action encoders and output heads for navigation and manipulation, sharing a common backbone. This design supports joint representation learning on independently sampled navigation and manipulation data. UniWAM supports independent inference for either stream and batch-parallel inference for both. We further introduce Manipulation Anchor Pose (MAP) supervision for where to stop and how to orient for manipulation. An automated pipeline constructs MAP-Data from large-scale 3D scenes, yielding over 1.5 million episodes and 7,500 hours. MAP-Data provides per-frame target-object bounding boxes and image-plane MAP coordinates as auxiliary navigation supervision. Together with projected end-effector trajectories for manipulation, these prediction targets provide stream-specific image-plane supervision for action learning from egocentric observations. With large-scale MAP-Data, UniWAM outperforms the strongest external baselines on our MAP-Bench by 30.1% in position error and 44.0% in heading error. Across 24 real-robot tasks, UniWAM achieves leading results in MAP navigation and mobile manipulation, with competitive manipulation performance. We will release code, data, and benchmark.

1 INTRODUCTION

Mobile manipulation requires a robot to move through an environment and interact precisely with objects after arrival. Compared with navigation or fixed-base manipulation alone, mobile manipulation supports a much broader range of tasks, such as fetching an item from another room (Yenamandra et al., 2023; Fu et al., 2024). However, navigation and manipulation differ in both what they need to observe and how they act. Navigation needs a forward view of the scene ahead, including distant targets and free space, whereas manipulation needs detailed views of the local workspace, often from downward-facing and wrist cameras (Yang et al., 2026a; Zhu et al., 2026; Chen et al., 2026a). Their action signals also differ in frequency: base motion varies slowly and smoothly over long distances, whereas arm and gripper motion changes rapidly near contact (Zhu et al., 2026).

Research on mobile manipulation mainly follows two routes: modular systems and unified models. Modular systems train navigation and manipulation policies separately and connect them at execution time (Wu et al., 2025b). This handoff adds an interface and can propagate errors between modules (Zhu et al., 2026; Wu et al., 2025b). Unified models instead learn the complete mobile-manipulation process end-to-end within a single policy. Recent unified approaches build on vision-language-action (VLA) models (Physical Intelligence et al., 2026; Bjorck et al., 2025) and world-action models (WAMs) (Yuan et al., 2026b; Fan et al., 2026) to predict navigation and manipulation actions within a shared model. However, unified mobile manipulation faces two linked challenges (Fig. 1). The first is architectural. A shared action token for navigation and manipulation (Black et al., 2025a; Bjorck et al., 2025) ignores their heterogeneous visual observations, action spaces, and control rates. Separate action experts (Chen et al., 2026c) decouple the action spaces,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

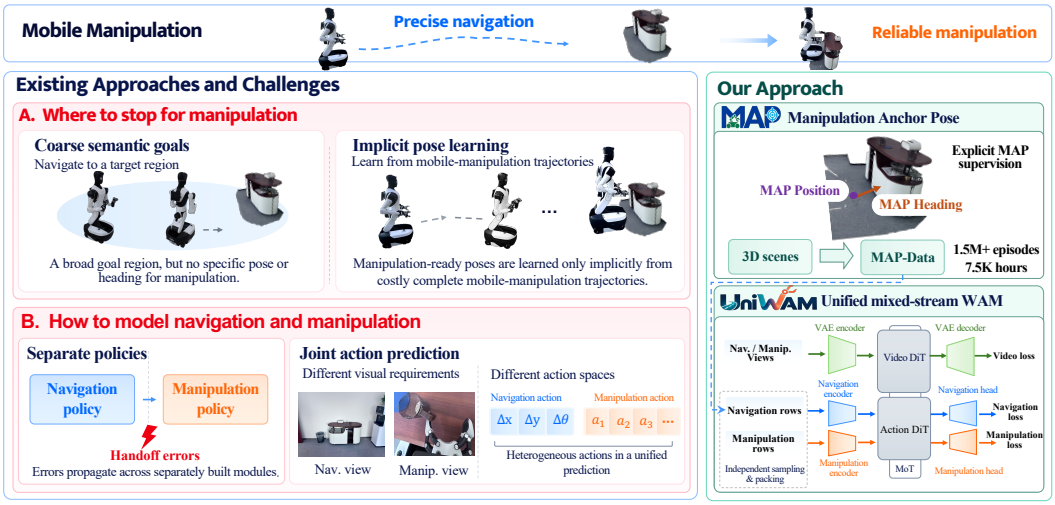

Figure 1: Challenges of mobile manipulation and our approach. UniWAM models navigation and manipulation as separate streams, and MAP-Data supervises the manipulation-ready pose at scale.

yet both designs still jointly supervise both action targets within the same training sample. Because mobile manipulation is largely sequential, the arm is mostly idle during navigation and the base is mostly idle during manipulation. Joint supervision thus spends much of the training on stationary commands, ties the sampling of the two behaviors to complete trajectories, and prevents direct use of navigation-only or manipulation-only data. The second challenge, a data problem, follows from the first. These models learn the manipulation-ready terminal pose only implicitly from complete trajectories, so they require large-scale mobile-manipulation data, which are costly to collect (Xu et al., 2026; Zhu et al., 2025). Without them, a policy may reach the target but stop at an unsuitable position or heading (Yang et al., 2025). Semantic navigation data scale better but specify only target proximity, not the terminal pose required for manipulation (InternNav Team, 2025).

To address the first challenge, we propose **UniWAM**, a mixed-stream world-action model with separate navigation and manipulation streams. Each stream has its own visual inputs, action encoder, and output head over shared video and action DiTs. The two streams are sampled independently and packed as separate batch rows without cross-row attention, so each sample supervises only its own stream. Navigation-only, manipulation-only, and mobile-manipulation data can all be used directly, and either stream runs alone or both run in one batch at deployment. To address the second challenge, we introduce the **Manipulation Anchor Pose (MAP)**, the base position and heading at which navigation should end for manipulation. Unlike floor affordances, candidate base placements, or pose preferences learned from rollouts (Zhang et al., 2025b; Zhong et al., 2026; Chen et al., 2026b; Chai et al., 2025), MAP is an explicit target that can be generated automatically and supervises the entire approach trajectory. We build **MAP-Data** from large-scale indoor 3D scenes with an automated pipeline, yielding over 1.5 million navigation episodes (7,500 hours) annotated with per-frame target bounding boxes and image-plane MAP coordinates, and hold out a subset of its targets as **MAP-Bench** for evaluating MAP navigation. Because the streams are sampled independently, UniWAM trains its navigation stream directly on MAP-Data, which supervises where to stop for mobile manipulation explicitly and at scale. Together with projected end-effector trajectories for manipulation, the bounding boxes and MAP coordinates serve as image-plane auxiliary targets for the two streams (Tu et al., 2026; Yang et al., 2026b). Our contributions are:

- We propose **UniWAM**, a mixed-stream world-action model that learns navigation and manipulation as two independently sampled streams over a shared backbone.
- We define the **Manipulation Anchor Pose (MAP)** and generate **MAP-Data**, over 1.5M episodes labeled with MAP and target boxes. To our knowledge, it is the largest dataset for manipulation-oriented navigation. We hold out **MAP-Bench**, a leakage-free benchmark for MAP navigation.
- We evaluate UniWAM on MAP-Bench and 24 real-robot tasks. Through mixed training on MAP-Data and diverse manipulation data, UniWAM outperforms pretrained baselines, reducing position and heading errors by 30.1% and 44.0% and raising mobile-manipulation success by 17.5 points.

2 RELATED WORK

Unified mobile-manipulation models. Most unified policies predict base, arm, and gripper commands through a shared action representation. RT-1 adds base motion to arm control and selects the controlled component with a discrete mode variable (Brohan et al., 2023). Mobile ALOHA concatenates bimanual joint positions with base linear and angular velocities into one action vector (Fu et al., 2024). VLA models such as π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), GR00T (Bjorck et al., 2025), and Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026) follow the same interface, predicting all degrees of freedom in one action chunk and zero-padding or masking the dimensions absent from a data source. This design learns coordinated behavior, but one action token must cover navigation and manipulation despite their different visual inputs, action spaces, and control rates, and each sample from a complete trajectory supervises both behaviors, including the idle one. World-action models (WAMs) build on video generation models to jointly predict future visual states and actions (Ye et al., 2026; Kim et al., 2026; Pai et al., 2025; Bi et al., 2025; Li et al., 2026; Yuan et al., 2026b). Most WAMs target fixed-base manipulation. ABot-M0.5 (Chen et al., 2026c) and MobileWAM (Fan et al., 2026) extend them to mobile manipulation. ABot-M0.5 assigns base and manipulation actions to separate sub-towers within a dual-level mixture-of-transformers. MobileWAM attaches an action expert to a pretrained video diffusion transformer, routes its feed-forward layers across shared, locomotion, and manipulation experts, and supervises future latents with Chain-of-Foresight. Both methods separate navigation and manipulation only within the action module: the two behaviors still share visual inputs, and each sample still supervises both. UniWAM instead separates them at the stream level: each stream has its own visual inputs, action encoder, and output head and is sampled independently, while both streams share the video and action DiTs. Appendix A compares these models with UniWAM in views, sampling, and supervision.

Manipulation-oriented navigation. Unlike semantic navigation, which only requires reaching the vicinity of a target (Zhang et al., 2025a; InternNav Team, 2025), manipulation-oriented navigation must determine a base position and heading that support the subsequent interaction. Existing methods fall into three groups. The first searches for feasible base placements from arm reachability (Vahrenkamp et al., 2013). The second learns manipulation-ready locations from labeled or executed data, including floor affordance maps (Zhang et al., 2025b; Zhong et al., 2026), grasp-aware base placement (Chen et al., 2026b), and pose preferences learned from manipulation rollouts (Chai et al., 2025). The third infers the base pose at test time by optimizing compatibility with a manipulation policy (Yang et al., 2025) or by reasoning with vision-language models (Wu et al., 2025a; Zhang et al., 2026). These methods output candidate poses, affordance regions, or placement scores for a specific robot, policy, or scene. As a result, navigation data that explicitly supervise the manipulation-ready terminal pose along entire approach trajectories remain scarce, although such supervision is essential for learning where to stop. We define this terminal pose as the MAP and generate MAP-Data from large-scale indoor 3D scenes, to our knowledge the largest dataset of its kind, and compare it with these methods in output, label source, and scale in Appendix A.

3 METHOD

3.1 MAP: MANIPULATION ANCHOR POSE

Existing representations of manipulation-ready base placement are specific to a robot, policy, or scene and do not supervise the approach trajectory (Zhang et al., 2025b; Chai et al., 2025). We instead make the terminal base pose an explicit target and generate approach trajectories at scale.

MAP definition. For a manipulation target o_i, its *Manipulation Anchor Pose* (MAP) is $\mathbf{m}_i = (\mathbf{p}_i^{\mathrm{MAP}}, \theta_i^{\mathrm{MAP}})$, where $\mathbf{p}_i^{\mathrm{MAP}} \in \mathbb{R}^2$ and $\theta_i^{\mathrm{MAP}} \in \mathbb{S}^1$ denote the base position and heading. The MAP explicitly specifies where the robot should stop and how it should orient for manipulation.

MAP-Data construction. MAP-Data is built from HM3D/HM3D-SEM (Ramakrishnan et al., 2021; Yadav et al., 2022), InteriorGS (Miao et al., 2026), and a large in-house scene corpus. As illustrated in Fig. 2, the pipeline has three stages. First, S1 curates semantically valid, geometrically complete, and visually reliable objects. S2 constructs MAP targets and feasible observation poses: a vision-language model identifies each target's manipulation-facing front edge, a ray cast from its midpoint along the front normal determines the MAP position at the boundary of the clearance-aware navigable region, and the heading points back toward the target surface. Finally, S3 increases the diversity

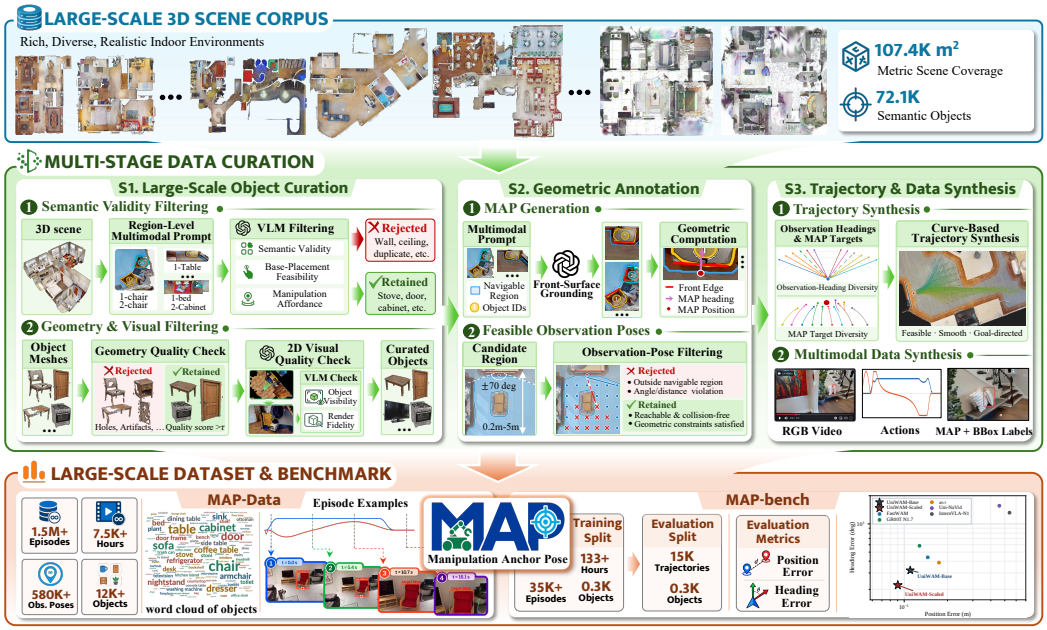

Figure 2: Overview of MAP-Data and MAP-Bench. From large-scale indoor 3D scenes, S1 curates valid target objects, S2 annotates each target with a MAP and feasible observation poses, and S3 synthesizes approach trajectories with RGB video, actions, and MAP and box labels. MAP-Bench evaluates position and heading errors from held-out initial poses.

of MAP targets and initial observation poses, then pairs them to generate diverse navigation trajectories with synchronized observations, actions, and annotations. Each trajectory contains RGB observations, navigation actions, projected MAP coordinates, and target bounding boxes, together with five target-specific instructions. MAP-Data contains over **1.5M** episodes and **7.5K** hours, providing navigation trajectories together with explicit MAP and image-plane geometric supervision for UniWAM. Further details of the MAP-Data construction process are provided in Appendix B.

MAP-Bench. MAP-Bench contains 300 targets in 109 scenes, with 35K training and 15K evaluation trajectories. To prevent leakage between the splits, we cluster the initial poses in $SE(2)$ with a distance that normalizes position and heading differences, and assign each cluster entirely to one split. Hence, the two splits share targets but never share an initial pose or a near-duplicate initial view, so the evaluation measures generalization to unseen starting configurations rather than memorized trajectories. All MAP-Bench targets are further excluded from the rest of MAP-Data, so the additional data used by UniWAM-Scaled contain no benchmark target. Appendix C.1 details the protocol. We evaluate MAP-Bench with the MAP Position Error and the MAP Heading Error. For N evaluation trajectories, they are defined as

$$E_{\text{pos}} = \frac{1}{N}\sum_{r=1}^{N}\left\|\widehat{\mathbf{p}}_r - \mathbf{p}_r^{\text{MAP}}\right\|_2, \qquad E_{\text{head}} = \frac{1}{N}\sum_{r=1}^{N}\left|\text{wrap}\left(\widehat{\theta}_r - \theta_r^{\text{MAP}}\right)\right|, \tag{1}$$

where $\widehat{\mathbf{p}}_r$, $\widehat{\theta}_r$ are the reached position and heading, and $\mathbf{p}_r^{\text{MAP}}$, θ_r^{MAP} those of the reference MAP. Eq. 1 is applied per distance range, and Overall averages the three ranges.

3.2 UniWAM: Unified Mixed-Stream World-Action Model

UniWAM separates navigation and manipulation at the level of control streams rather than within the action module (Fig. 3). Each stream has its own visual inputs, action encoder, output head, and learned tokens, while both streams share a video DiT, initialized from Wan2.2-TI2V-5B (Wan Team et al., 2025), and an action DiT, coupled through mixture-of-transformers attention (Liang et al., 2024). This design addresses the two problems identified in the introduction: stream-specific interfaces accommodate the heterogeneous observations and action spaces, and independent sampling lets each training sample supervise only one stream. The shared DiTs keep a single model

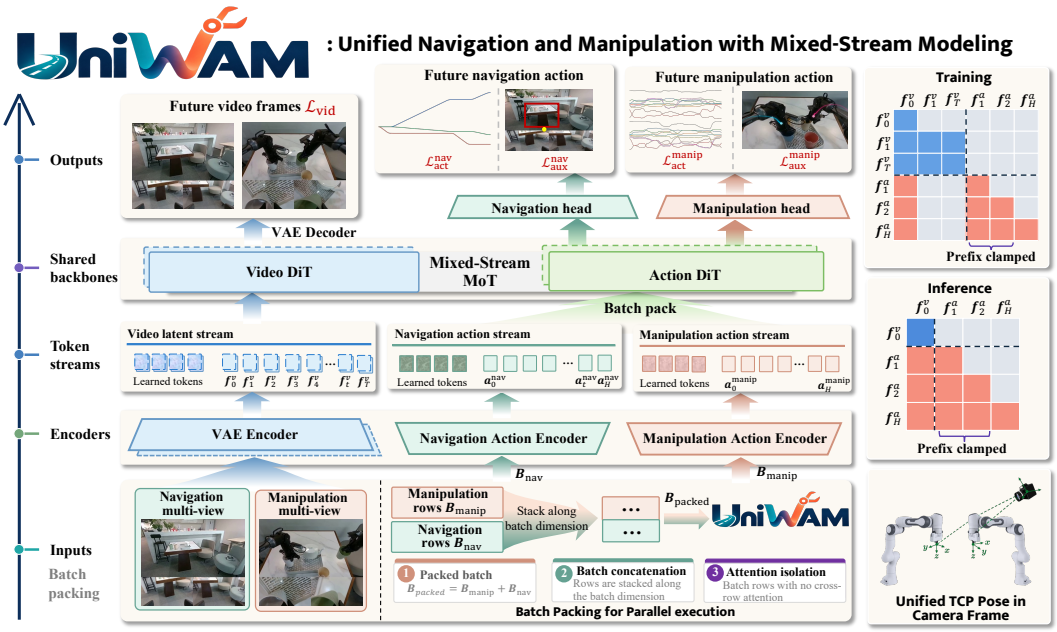

Figure 3: Overview of UniWAM. Navigation and manipulation samples are drawn independently, encoded by stream-specific action encoders, and packed as separate batch rows through the shared video and action DiTs. Stream-specific heads predict actions and image-plane auxiliary targets, while the video branch predicts future frames. Right: attention masks with clamped action prefixes for training and inference, and the camera-frame end-effector pose shared across embodiments.

and a shared representation for both streams. We describe below the stream interfaces, batch packing, independent sampling, auxiliary supervision, and training objective. Appendix D details the auxiliary-label construction, full attention mask, and masked flow-matching loss, and Appendix D.1 gives the network configuration, optimization settings, and training and inference acceleration.

Stream interfaces. Let $s \in \{\text{nav}, \text{manip}\}$. At time t, stream s receives camera views o_t^s, the instruction ℓ, and the proprioceptive state q_t, and predicts an action chunk $A_t^s = (a_{t,1}^s, \ldots, a_{t,H}^s)$. Navigation actions are planar pose targets $(\Delta x, \Delta y, \Delta\theta)$ relative to the current base pose, which specify where the base should go rather than how fast it should move. Manipulation actions are dual-arm end-effector poses in the current camera frame, each arm given by a 3D position, a continuous 6D rotation (Zhou et al., 2019), and a gripper state, which makes visually similar motions numerically similar across embodiments (Yuan et al., 2026a). Appendix D.2 details both action spaces. A stream-specific linear encoder maps each action to the shared hidden space, and a stream-specific head decodes it. Because the DiTs are shared, learned tokens for the video and actions of each stream mark which view and action space each sequence belongs to.

Mixed-stream batch packing. Examples of both streams are stacked along the batch dimension,

$$X^{\text{pack}} = \text{Concat}_B \left(X^{\text{manip}}, X^{\text{nav}} \right) \in \mathbb{R}^{(B_{\text{manip}}+B_{\text{nav}})\times L\times D}, \tag{2}$$

where B_{manip} and B_{nav} need not be equal. Visual tokens and conditioning are packed in the same order, and attention never crosses batch rows. A navigation row conditions only on navigation views and actions, and a manipulation row only on manipulation ones, while both update the same DiT parameters. Within a row, each action token attends to the current visual context and to earlier actions only. Under prefix-conditioned asynchronous inference, this causal mask keeps the clean committed prefix from attending to later noisy actions, so each new chunk continues the executed motion consistently. At deployment, both streams run continuously in one batch without switching model parameters or invoking a separate policy. Our optimized training and inference infrastructure reduces dual-stream inference latency on one NVIDIA H200 from about 220 ms to below 60 ms. Appendix D.1 details these training and inference optimizations.

Independent stream sampling. In a complete mobile-manipulation trajectory, the arm is mostly idle during navigation and the base is mostly idle during manipulation. Consequently, sampling

complete trajectories lets phase durations decide how often each stream sees active and idle actions. We instead index navigation and manipulation windows as separate examples: a mobile-manipulation trajectory contributes to both streams, whereas navigation-only data such as MAP-Data and manipulation-only data contribute only to their own stream. Examples are drawn from

$$q(d,i) = \frac{\alpha_d}{\sum_{d'} \alpha_{d'}} \frac{w_{d,i}}{\sum_j w_{d,j}}, \tag{3}$$

where α_d is the weight of source d and $w_{d,i}$ the weight of its i-th window. Normalizing within each source decouples its sampling mass from its size, and the window weights set the proportions of navigation and manipulation windows and of active and idle windows. In this way, idle windows are retained at a controlled proportion instead of dominating training.

Image-plane auxiliary supervision. Both streams predict spatial actions from images. To ground these actions in the image, we append image-plane targets to each action vector, $a^s_{t,i} = [a^{s,\mathrm{act}}_{t,i}; a^{s,\mathrm{aux}}_{t,i}]$, and denoise them jointly with the actions, as shown in the output row of Fig. 3. For navigation, the targets are the target bounding box and the projected MAP position, which indicate where the target is and where the base should stop at every step of the approach. For manipulation, the targets are the two end-effector trajectories projected into the main camera view, together with visibility flags. Predicting these tracks jointly with the camera-frame end-effector poses explicitly links the camera-frame actions to their image-plane locations in the main view. The projected MAP and end-effector positions are image-plane counterparts of the navigation goal and the manipulation action, keeping the auxiliary targets geometrically consistent with the actions. Appendices B.4 and D.6 describe how the navigation and manipulation labels are constructed.

Training objective. All targets are trained with the flow-matching loss described in Appendix D:

$$\mathcal{L} = \lambda_{\mathrm{vid}}\mathcal{L}_{\mathrm{vid}} + \sum_{s\in\{\mathrm{nav},\mathrm{manip}\}} \left(\lambda^s_{\mathrm{act}}\mathcal{L}^s_{\mathrm{act}} + \lambda^s_{\mathrm{aux}}\mathcal{L}^s_{\mathrm{aux}}\right), \tag{4}$$

where $\mathcal{L}_{\mathrm{vid}}$ supervises future video latents of both streams, $\mathcal{L}^s_{\mathrm{act}}$ the actions, and $\mathcal{L}^s_{\mathrm{aux}}$ the image-plane targets. Each term is averaged only over valid positions and features, as defined in Eq. 10, so a sample contributes only to the losses of its own stream and never to those of the other.

4 Experiments

4.1 Experimental Setup

Tasks and platforms. MAP-Bench evaluates MAP navigation from held-out initial poses. Its episodes are grouped by initial distance into a short range below 1.5 m, a mid range from 1.5 to 3.5 m, and a long range of at least 3.5 m. The real-robot evaluation covers 13 MAP Navigation, 4 Mobile Manipulation, and 7 Manipulation tasks on the AgileX Cobot Magic, AgiBot G2, dual-arm Franka Research 3, and AgileX single-arm platforms. On the mobile platform, the navigation stream observes a wide-view head camera and the manipulation stream a downward-facing head camera and the two wrist cameras. In Mobile Manipulation, the navigation stream also observes the wrist cameras. Fig. 5 in the appendix shows this setup. For MAP Navigation and Mobile Manipulation, the robot starts 1–5 m from the recorded reference MAP with a heading offset within $\pm 45°$. Manipulation starts from a fixed working pose. MAP Navigation tasks run on both the Cobot Magic and the AgiBot G2 with errors averaged over the two, and each task has 40 trials per embodiment.

UniWAM variants. Both variants start from Wan2.2-TI2V-5B without embodied pretraining. **UniWAM-Base** uses only the data available to baselines and is the matched comparison: the 35K-trajectory training split on MAP-Bench and the demonstrations of the 24 evaluation tasks on real robots. **UniWAM-Scaled** keeps the architecture and adds data through independent stream sampling, drawing large-scale and task data at a 7:3 ratio. On MAP-Bench, it adds MAP-Data, which excludes all benchmark targets, under the same budget, so its gain isolates the value of MAP-Data. On real robots, it adds MAP-Data to the navigation stream and 402 private manipulation tasks with 30 demonstrations each, disjoint from the evaluation tasks, to the manipulation stream. Appendix E.2 gives the training details of both variants. Each variant is a single checkpoint for all embodiments.

Baselines. On MAP-Bench, we compare with the VLA models $\pi_{0.5}$ (Black et al., 2025a) and GR00T N1.7 (Bjorck et al., 2025), the VLN models Uni-NaVid (Zhang et al., 2025a) and InternVLA-N1 (In-

Table 1: MAP-Bench results over the evaluation split. Rows below the double rule are UniWAM-Base ablations. Bold and underline mark the best and second-best non-ablation results.

Model	E_{pos} (cm) ↓ Short	Mid	Long	Overall	E_{head} (°) ↓ Short	Mid	Long	Overall
Vision-language-action models (VLA)								
GR00T N1.7 (Bjorck et al., 2025)	9.48	10.8	22.0	14.1	5.66	5.42	7.46	6.18
$\pi_{0.5}$ (Black et al., 2025a)	7.98	9.85	50.8	22.9	3.26	3.51	5.55	4.11
Vision-language navigation models (VLN)								
Uni-NaVid (Zhang et al., 2025a)	17.3	40.0	87.2	48.2	7.98	15.0	23.4	15.5
InternVLA-N1 (InternNav Team, 2025)	17.3	39.7	130.0	62.3	3.72	11.0	27.6	14.1
World-action models (WAM)								
FastWAM (Yuan et al., 2026b)	9.24	11.0	31.1	17.1	3.44	3.95	6.66	4.68
UniWAM-Base	<u>7.71</u>	<u>8.85</u>	<u>21.0</u>	<u>12.5</u>	<u>2.37</u>	<u>3.06</u>	<u>4.29</u>	<u>3.24</u>
UniWAM-Scaled	**7.48**	**7.86**	**14.2**	**9.85**	**1.82**	**2.08**	**3.01**	**2.30**
Ablations of UniWAM-Base								
Velocity actions (v, ω)	8.30	9.57	30.9	16.3	3.03	3.29	6.19	4.17
w/o causal mask	8.83	10.1	29.0	16.0	3.21	3.50	5.49	4.07
w/o target box	7.74	9.29	29.5	15.5	2.43	3.02	5.14	3.53
w/o image-plane MAP	8.05	8.40	22.5	13.0	2.63	2.82	4.72	3.39
w/o both auxiliary targets	9.31	11.6	29.1	16.7	4.45	4.91	6.76	5.37

Table 2: Real-robot results averaged over the tasks in each category. Rows below the double rule are UniWAM-Base ablations. Bold and underline mark the best and second-best non-ablation results.

Model	MAP Navigation E_{pos} (cm) ↓	E_{head} (°) ↓	Mobile Manipulation E_{pos} (cm) ↓	E_{head} (°) ↓	SR (%) ↑	Manipulation SR (%) ↑
$\pi_{0.5}$ (Black et al., 2025a)	23.3	5.15	26.2	5.72	62.5	**71.4**
GR00T N1.7 (Bjorck et al., 2025)	18.1	6.55	20.4	7.20	39.4	36.8
UniWAM-Base	<u>15.1</u>	<u>4.91</u>	<u>17.1</u>	<u>5.30</u>	<u>63.1</u>	44.3
UniWAM-Scaled	**11.7**	**4.25**	**13.2**	**4.56**	**80.0**	<u>68.9</u>
Ablations of UniWAM-Base						
w/o navigation auxiliary supervision	17.6	6.84	19.9	7.21	56.9	43.6
w/o manipulation auxiliary supervision	15.4	5.18	17.8	5.61	58.8	39.3
Joint whole-body sampling	16.8	5.62	18.6	5.87	60.0	41.8

ternNav Team, 2025), and the world-action model FastWAM (Yuan et al., 2026b). All models, including UniWAM, are trained on the same split for 40,000 steps on 16 NVIDIA H200 GPUs, and the baselines otherwise follow their official recipes. On real robots, we compare with $\pi_{0.5}$ and GR00T N1.7, trained on the same demonstrations as UniWAM-Base, each in the training configuration and checkpoint that worked best for it. For fairness, all methods share the data, cameras, initial configurations, and evaluation protocol. Appendices C.2, E.2, and E.3 and Table 6 detail these controls.

Metrics. MAP Navigation is measured by E_{pos} and E_{head} as defined in Eq. 1. On real robots, the reached pose is obtained from the onboard state estimator. Mobile Manipulation reports the same errors when the base stops, together with the success rate (SR) of the subsequent manipulation. Manipulation reports SR. Real-robot results are unweighted means over the tasks in each category.

4.2 MAP-Bench Results

Table 1 summarizes MAP-Bench performance across models and initial distances. **(i) Overall comparison.** UniWAM-Scaled achieves the lowest errors in every distance range, reaching 9.85 cm and 2.30° overall, which are 30.1% and 44.0% lower than the strongest external baselines, GR00T N1.7 at 14.1 cm and $\pi_{0.5}$ at 4.11°, respectively. Without MAP-Data, UniWAM-Base already reduces these errors by 11.3% and 21.2%. **(ii) Semantic versus manipulation-ready navigation.** Although trained on the same split, the VLN models retain large terminal errors, with Overall position errors of 48.2 and 62.3 cm and long-range errors of 87.2–130.0 cm. Both are designed for semantic navigation, and their official recipes keep the planning module of InternVLA-N1 frozen and restrict Uni-NaVid to discrete motion primitives. VLA and WAM baselines are more accurate but degrade with distance. For example, $\pi_{0.5}$ increases from 7.98 cm at short range to 50.8 cm at long range. The trajectories in Appendix G show the same pattern: the VLN baselines often head to a wrong region, whereas UniWAM follows the reference path and stops at the MAP. **(iii) Long-range dock-**

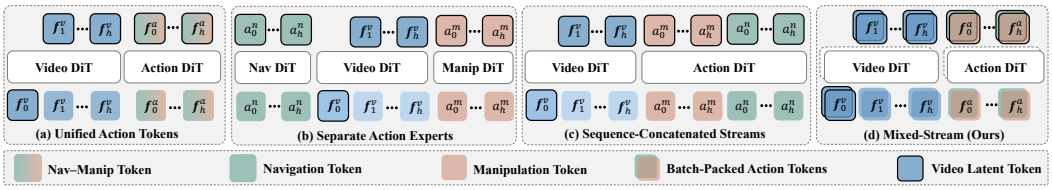

Figure 4: Unified mobile-manipulation architectures: (a) shared action tokens, (b) separate navigation and manipulation action DiTs, (c) streams concatenated along the sequence axis, and (d) Mixed-Stream (ours), with streams in separate batch rows of shared video and action DiTs.

Table 3: Architecture and sampling ablations on the four Mobile Manipulation tasks. (a)–(c) use joint sampling. (d) uses joint or independent sampling, the latter being UniWAM-Base.

	Alternative architectures			**(d) Mixed-stream**	
Metric	(a) Unified tokens	(b) Sep. experts	(c) Seq. concat.	Joint	**Indep.**
E_{pos} (cm) ↓	22.0	19.7	22.1	18.6	**17.1**
E_{head} (°) ↓	7.49	6.06	6.78	5.87	**5.30**
SR (%) ↑	46.9	54.4	60.0	60.0	**63.1**

ing. Relative to the best baseline in each range, UniWAM-Scaled reduces position error by 6.3% at short range and 35.5% at long range. Its position error grows by a factor of 1.9 from short to long range, compared with 2.3 for GR00T N1.7, 3.4 for FastWAM, and 6.4 for $\pi_{0.5}$. Its heading error is 40.7–45.8% lower than the best baseline across ranges. **(iv) MAP-Data.** UniWAM-Base and UniWAM-Scaled share the architecture and training budget, while UniWAM-Scaled mixes MAP-Data. This reduces Overall position and heading errors by 21.2% and 29.0%, and the position gain increases from 3.0% at short range to 32.4% at long range. As shown in Appendix F.2, the errors fall monotonically with more MAP-Data, whereas $\pi_{0.5}$ gains only 5.7% from the same data.

4.3 Real-Robot Results

Table 2 summarizes 24 real-robot tasks, and Appendix G shows representative executions. Among the evaluated models, UniWAM-Scaled achieves the lowest position and heading errors in MAP Navigation and Mobile Manipulation, the highest Mobile Manipulation success rate, and the second-highest Manipulation success rate. **(i) MAP navigation.** The two baselines trade off position and heading accuracy: GR00T N1.7 has the lower position error, 18.1 cm compared with 23.3 cm for $\pi_{0.5}$, whereas $\pi_{0.5}$ has the lower heading error, $5.15°$ compared with $6.55°$. Trained only on task data, UniWAM-Base outperforms both on both metrics, reaching 15.1 cm and $4.91°$. UniWAM-Scaled further reduces the errors to 11.7 cm and $4.25°$, 35.4% and 17.5% below the strongest baseline, respectively. It achieves the lowest position error on all 13 tasks and the lowest heading error on 11. **(ii) Mobile manipulation.** UniWAM-Scaled's docking accuracy is reflected in end-to-end performance: it stops 13.2 cm and $4.56°$ from the MAP, 35.3% and 20.3% below the best baseline, respectively. Its success rate reaches 80.0%, 17.5 percentage points above $\pi_{0.5}$ at 62.5%, with non-overlapping 95% intervals as shown in Appendix F.3, and is highest on all four tasks. UniWAM-Base reaches a success rate similar to that of $\pi_{0.5}$, 63.1% compared with 62.5%, while docking more accurately. **(iii) Manipulation.** $\pi_{0.5}$ achieves the highest average success at 71.4%, followed by UniWAM-Scaled at 68.9%. UniWAM-Scaled leads on the three sorting and placement tasks, while $\pi_{0.5}$ leads by 2.5–12.5 percentage points on six-cup stacking, the two tube-insertion tasks, and long-horizon coffee preparation. $\pi_{0.5}$ is pretrained on large-scale robot data, whereas UniWAM starts from a video model and relies on mixed-data training, so the remaining gap is largest on these dexterous or long-horizon tasks. GR00T N1.7 has the lowest success at 36.8%. **(iv) Scaling.** Relative to UniWAM-Base, UniWAM-Scaled reduces MAP Navigation errors by 22.5% and 13.4% and raises success by 16.9 and 24.6 percentage points on Mobile Manipulation and Manipulation. These gains reflect both more data and more compute. Sec. 4.2 isolates the effect of MAP-Data. As shown in Appendix F.1, UniWAM trained only on MAP-Data reaches a zero-shot position error of 18.7 cm on real robots, below the 23.3 cm of $\pi_{0.5}$ trained on real demonstrations.

4.4 Comparison of Unified Architectures

Figure 4 and Table 3 compare the designs on the four Mobile Manipulation tasks with the UniWAM-Base data, training budget, and evaluation protocol. The three alternative architectures require both action targets in each sample, so they use joint sampling. We also evaluate the mixed-stream design with joint sampling to separate architectural effects from sampling effects. **(i) Architecture.** Under joint sampling, the mixed-stream design obtains the lowest docking errors, 18.6 cm and $5.87°$, and ties Sequence-Concatenated Streams in success at 60.0%. Compared with Unified Action Tokens, it reduces position and heading errors by 15.5% and 21.6% and improves success by 13.1 percentage points. Compared with Separate Action Experts, it lowers position and heading errors by 5.6% and 3.1% and raises success by 5.6 percentage points, despite a single shared action DiT. Relative to Sequence-Concatenated Streams, it reduces position and heading errors by 15.8% and 13.4% at the same success rate. **(ii) Sampling.** In the mixed-stream design, independent sampling further reduces errors from 18.6 cm and $5.87°$ to 17.1 cm and $5.30°$ and increases success from 60.0% to 63.1%. Both design choices reduce docking errors and together give the highest success.

4.5 Ablation Studies

Each ablation changes one factor with the data, training budget, and evaluation protocol fixed. Results are in the bottom blocks of Tables 1 and 2. **(i) Pose targets.** Replacing the relative pose target $(\Delta x, \Delta y, \Delta\theta)$, motivated in Appendix D.2, with velocity commands (v, ω) increases the Overall errors from 12.5 cm and $3.24°$ to 16.3 cm and $4.17°$. The position error rises only from 7.71 to 8.30 cm at short range but from 21.0 to 30.9 cm at long range. **(ii) Causal mask.** Removing the causal mask increases the Overall errors to 16.0 cm and $4.07°$ and the long-range position error from 21.0 to 29.0 cm. The mask lets later actions condition on the clean committed prefix without leaking information from later noisy actions backward. **(iii) Auxiliary supervision.** The two navigation targets have different distance-dependent effects. Removing the target box barely changes the short-range errors but raises the long-range errors from 21.0 cm and $4.29°$ to 29.5 cm and $5.14°$, suggesting that the box helps localize distant, small targets. Removing the projected MAP, which marks where the base should stop, instead raises the short-range errors more than removing the box, from 7.71 cm and $2.37°$ to 8.05 cm and $2.63°$. Removing both raises the Overall errors to 16.7 cm and $5.37°$, above those of GR00T N1.7 and $\pi_{0.5}$, suggesting complementary effects. On real robots, removing navigation auxiliary supervision changes MAP Navigation from 15.1 cm and $4.91°$ to 17.6 cm and $6.84°$ and lowers Mobile Manipulation success from 63.1% to 56.9%. Removing manipulation auxiliary supervision leaves navigation close to the full model but lowers Manipulation success from 44.3% to 39.3% and Mobile Manipulation success from 63.1% to 58.8%. Each auxiliary target has its largest effect on its own stream. Appendix G visualizes these predictions. **(iv) Independent sampling.** Joint whole-body sampling worsens all six real-robot metrics, raising the MAP Navigation errors to 16.8 cm and $5.62°$ and lowering Mobile Manipulation success to 60.0%. Independent sampling thus improves both streams without extra data. **(v) Additional choices.** Appendix F further ablates the number of learned tokens per bank and the manipulation coordinate frame.

5 Conclusion

We presented UniWAM, a mixed-stream world-action model that treats navigation and manipulation as two independently sampled control streams over shared video and action DiTs, and the Manipulation Anchor Pose (MAP), an explicit target for manipulation-ready navigation supported by over 1.5 million automatically generated MAP-Data episodes. Independent sampling lets UniWAM learn directly from navigation-only and manipulation-only data, and image-plane auxiliary supervision grounds the actions of both streams in the egocentric image. On MAP-Bench, UniWAM trained with MAP-Data reduces position and heading errors by 30.1% and 44.0% relative to the strongest baselines. On 24 real-robot tasks across four embodiments, a single checkpoint trained with additional data achieves the most accurate docking and the highest mobile-manipulation success while remaining competitive in manipulation. Future work will study pretraining UniWAM on larger egocentric-video and robot-interaction datasets together with MAP-Data.

AI USE STATEMENT

Generative AI tools were used for language polishing and proofreading. The authors reviewed and verified all outputs and take full responsibility for the final manuscript.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023.

Kaixin Chai, Hyunjun Lee, and Joseph J. Lim. N2M: Bridging navigation and manipulation by learning pose preference from rollout. *arXiv preprint arXiv:2509.18671*, 2025.

Guangyu Chen, Qiwei Liang, Shaolong Zhu, Tianxing Chen, Zikuan Xiao, Yifan Xie, Lingfeng Zhang, Ping Luo, Renjing Xu, and Wenbo Ding. MoPA: Coordinated mobile manipulation via subsystem-specific perception alignment. *arXiv preprint arXiv:2609.12081*, 2026a.

Jizhuo Chen, Diwen Liu, Jiaming Wang, and Harold Soh. GBPP: Grasp-aware base placement prediction for robots via two-stage learning. *arXiv preprint arXiv:2509.11594*, 2026b.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, et al. ABot-M0.5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026c.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, 2024.

Zehua Fan, Junjie He, Wenxuan Song, Xi Wang, Wenqi Lyu, Linge Zhao, Fuhao Li, Zihan You, et al. MobileWAM: Bridging world action models to mobile manipulation with chain-of-foresight. *arXiv preprint arXiv:2608.04657*, 2026.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning*, 2024.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

InternNav Team. InternVLA-N1: An open dual-system navigation foundation model with learned latent plans. Technical report, 2025. URL `https://internrobotics.github.io/internvla-n1.github.io/`.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Bingchen Miao, Rong Wei, Zhiqi Ge, Xiaoquan Sun, Shiqi Gao, Jingzhe Zhu, Renhan Wang, Siliang Tang, Jun Xiao, Rui Tang, and Juncheng Li. Towards physically executable 3D gaussian for embodied navigation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M. Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, et al. Habitat-matterport 3d dataset (HM3D): 1000 large-scale 3d environments for embodied AI. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Ruisen Tu, Arth Shukla, Sohyun Yoo, Xuanlin Li, Junxi Li, Jianwen Xie, Hao Su, and Zhuowen Tu. SG-VLA: Learning spatially-grounded vision-language-action models for mobile manipulation. *arXiv preprint arXiv:2603.22760*, 2026.

Nikolaus Vahrenkamp, Tamim Asfour, and Rüdiger Dillmann. Robot placement based on reachability inversion. In *IEEE International Conference on Robotics and Automation*, pp. 1970–1975, 2013.

Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *Advances in Neural Information Processing Systems*, 2024.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-language navigation. In *International Conference on Learning Representations*, 2026.

Zhenyu Wu, Angyuan Ma, Xiuwei Xu, Hang Yin, Yinan Liang, Ziwei Wang, Jiwen Lu, and Haibin Yan. MoTo: A zero-shot plug-in interaction-aware navigation for general mobile manipulation. *arXiv preprint arXiv:2509.01658*, 2025a.

Zhenyu Wu, Yuheng Zhou, Xiuwei Xu, Ziwei Wang, and Haibin Yan. MoManipVLA: Transferring vision-language-action models for general mobile manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, et al. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Xiaomeng Xu, Jisang Park, Han Zhang, Eric Cousineau, Aditya Bhat, Jose Barreiros, Dian Wang, Jeannette Bohg, and Shuran Song. HoMMI: Learning whole-body mobile manipulation from human demonstrations. In *Robotics: Science and Systems*, 2026.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel X. Chang, Dhruv Batra, Manolis Savva, et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022.

Donglin Yang, Haoran Chen, Xingyu Chen, Lixing Liu, Manyi Li, Changhe Tu, Ke Xu, Xiaojian Ma, and Si Liu. Learning panorama-aware VLA for mobile manipulation with whole-body teleoperation. *arXiv preprint arXiv:2608.02257*, 2026a.

Jingyun Yang, Isabella Huang, Brandon Vu, Max Bajracharya, Rika Antonova, and Jeannette Bohg. Mobi-π: Mobilizing your robot learning policy. *arXiv preprint arXiv:2505.23692*, 2025.

Zheng Yang, Wenjie Zhang, Xiangyu Chen, Wenxuan Song, Xianpeng Wang, Yihang Kang, Jiawen Wen, Wen Chen, et al. DreamTrajectory: Trajectory-guided action generation with world model alignment for mobile manipulation. *arXiv preprint arXiv:2608.01381*, 2026b.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. HomeRobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-RobotManip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-NaVid: A video-based vision-language-action model for unifying embodied navigation tasks. In *Robotics: Science and Systems*, 2025a.

Pingrui Zhang, Xianqiang Gao, Yuhan Wu, Kehui Liu, Dong Wang, Zhigang Wang, Bin Zhao, Yan Ding, and Xuelong Li. MoMa-Kitchen: A 100k+ benchmark for affordance-grounded last-mile navigation in mobile manipulation. *arXiv preprint arXiv:2503.11081*, 2025b.

Zhuofan Zhang, Tianxu Wang, Guoxi Zhang, Yixiong Lin, Xilin Wang, Hongming Xu, Qing Li, Song-Chun Zhu, and Lifeng Fan. UniLM-Nav: A unified framework for zero-shot last-mile navigation. *arXiv preprint arXiv:2607.06537*, 2026.

Ping Zhong, Manling Teng, Tao Wu, Bolei Chen, Jiazhi Xia, and Jianxin Wang. FloAff-Kitchen: Bridging navigation and manipulation via canonical and progressive floor affordance learning. *arXiv preprint arXiv:2607.24207*, 2026.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

Lawrence Y. Zhu, Pranav Kuppili, Ryan Punamiya, Patcharapong Aphiwetsa, Dhruv Patel, Simar Kareer, Sehoon Ha, and Danfei Xu. EMMA: Scaling mobile manipulation via egocentric human data. *arXiv preprint arXiv:2509.04443*, 2025.

Xiangyu Zhu, Renjun Wu, Luzhou Ge, Jinyan Liu, and Xuesong Li. GeoHAT: Geometry-adaptive hybrid action transformer for mobile manipulation. *arXiv preprint arXiv:2606.13394*, 2026.

A Comparison with Prior Designs

Table 4 compares UniWAM with unified mobile-manipulation models. Prior models either share one action representation across the two behaviors or separate the behaviors inside the action module. In both cases, the behaviors share visual inputs, and training on complete trajectories supervises both behaviors in every sample. The VLA policies $\pi_{0.5}$, GR00T N1.7, and Xiaomi-Robotics-1 can use navigation-only or manipulation-only data only by padding or masking the absent action dimensions. UniWAM separates navigation and manipulation at the stream level, supervises one stream per sample, and learns an explicit terminal pose. ABot-M0.5 and MobileWAM also build on a video world model but have none of the other three properties. The cross-embodiment policies CrossFormer and HPT use embodiment-specific tokenizers or stems and action heads over a shared trunk, so each sample supervises the action space of one robot. They separate different robots, however, not the navigation and manipulation phases of one mobile manipulator, and they have neither a video world model nor an explicit terminal pose. UniWAM applies this separation within one mobile manipulator and combines it with a video world model and explicit MAP supervision. Table 5 compares MAP-Data with methods for manipulation-ready base placement, which predict a pose, a floor affordance map, or pose scores for a given scene. MAP-Data instead supervises the entire approach trajectory with an explicit terminal pose and dense image-plane labels. None of the prior methods supervises the approach, and the largest prior dataset, MoMa-Kitchen with 127K episodes, is about one-twelfth the size of MAP-Data with its 1.5M episodes.

Table 4: Comparison with recent unified mobile-manipulation models and cross-embodiment policies. The first three are VLA policies, the next two are world-action models, and the last two are cross-embodiment policies, which separate different robots rather than the navigation and manipulation phases of one mobile manipulator. Separate views: navigation and manipulation receive different camera inputs. One behavior per sample: a training sample can supervise one behavior alone without padding or masking. Video world model: the model predicts future video. Explicit terminal pose: the terminal base pose for manipulation is supervised directly.

Model	Action generator	Separate views	One behavior per sample	Video world model	Explicit terminal pose
$\pi_{0.5}$ (Black et al., 2025a)	single action expert for all degrees of freedom	✗	✗	✗	✗
GR00T N1.7 (Bjorck et al., 2025)	single action head for all degrees of freedom	✗	✗	✗	✗
Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026)	single action head for all degrees of freedom	✗	✗	✗	✗
ABot-M0.5 (Chen et al., 2026c)	action stream with mobility and manipulation sub-towers	✗	✗	✓	✗
MobileWAM (Fan et al., 2026)	action expert with routed locomotion and manipulation experts	✗	✗	✓	✗
CrossFormer (Doshi et al., 2024)	shared transformer with embodiment-specific action heads	✓	✓	✗	✗
HPT (Wang et al., 2024)	shared trunk with embodiment-specific stems and heads	✓	✓	✗	✗
UniWAM	shared action DiT with separate stream interfaces	✓	✓	✓	✓

Table 5: Comparison of MAP-Data with methods for manipulation-ready base placement. Supervises approach: the labels cover the entire approach trajectory rather than only a terminal pose or a score for a given scene. Scale is the amount of labeled data reported in each paper, and – denotes methods that do not learn from a dataset, such as kinematic search or test-time optimization.

Method	Output	Label source	Supervises approach	Scale
Reachability inversion (Vahrenkamp et al., 2013)	feasible base poses	arm kinematics	✗	–
MoMa-Kitchen (Zhang et al., 2025b)	floor affordance map	simulation	✗	127K episodes
FloAff-Kitchen (Zhong et al., 2026)	floor affordance map	simulation	✗	24.8K observations
GBPP (Chen et al., 2026b)	scores of candidate base poses	rule labels and simulation trials	✗	180K rule and 12K simulation labels
N2M (Chai et al., 2025)	distribution over base poses	policy rollouts	✗	12–15 rollouts per task
Mobi-π (Yang et al., 2025)	base pose for a given policy	test-time optimization	✗	–
MAP-Data	terminal pose, per-frame box and MAP labels	scene geometry and a VLM	✓	1.5M episodes

B MAP-Data Construction

B.1 Source-Specific Preprocessing and Target Curation

MAP-Data combines heterogeneous indoor 3D sources. Mesh-based scenes undergo geometry and navigability checks, whereas Gaussian-splatting scenes are validated through occupancy maps and rendered observations. For sources without object-instance annotations, semantic instance segmentation supplies candidate target masks. We convert all sources into a common representation comprising target instances, navigable regions, and renderable observations. Candidate targets pass four

filters. Semantic filtering verifies that the instance matches the task condition and is uniquely identifiable. Geometry filtering rejects incomplete target surfaces and corrupted scene regions. Base-placement filtering requires collision-free navigable space near the manipulation-facing surface. Visibility and rendering filtering removes targets that cannot be observed reliably from feasible camera poses or that produce invalid rendered views, and the rest proceed to MAP generation.

B.2 MAP AND OBSERVATION-POSE GENERATION

Let $\mathbf{p}_f$ be the ground-plane midpoint of the manipulation-facing target boundary and let $\mathbf{n}_f$ be its unit normal pointing toward navigable space. Denoting the clearance-aware navigable region by $\mathcal{F}_{\mathrm{nav}}$, we construct the MAP position and heading as

$$\mathbf{p}^{\mathrm{MAP}} = \mathrm{Ray}(\mathbf{p}_f, \mathbf{n}_f) \cap \partial\mathcal{F}_{\mathrm{nav}}, \qquad \theta^{\mathrm{MAP}} = \mathrm{atan2}\big((\mathbf{p}_f - \mathbf{p}^{\mathrm{MAP}})_y, (\mathbf{p}_f - \mathbf{p}^{\mathrm{MAP}})_x\big). \tag{5}$$

The ray–boundary intersection determines where the base stops, and the heading makes it face the target surface. Observation poses are sampled independently across target distance, bearing, and camera heading. They are retained only when the base lies in navigable free space, satisfies collision and geometric constraints, and provides a valid view of the target. Sampling observation poses separately from the MAP diversifies initial states and leaves the terminal target fixed.

B.3 CONSTRAINED BÉZIER TRAJECTORY PLANNING

Each retained observation pose is connected to its MAP by a single- or multi-segment cubic Bézier trajectory. A segment with control points $\{\mathbf{c}_j\}_{j=0}^{3}$ is

$$\mathbf{q}(s) = (1-s)^3\mathbf{c}_0 + 3(1-s)^2 s\,\mathbf{c}_1 + 3(1-s)s^2\mathbf{c}_2 + s^3\mathbf{c}_3, \qquad s \in [0,1]. \tag{6}$$

The control points preserve the initial motion direction and terminate at the MAP with the prescribed heading. If a direct curve is obstructed, A* (Hart et al., 1968) provides a collision-free corridor that guides piecewise fitting. A trajectory is retained only if it remains within the clearance-aware navigation mask, approaches the target from the annotated side, keeps the manipulation-facing boundary within the horizontal field of view, and satisfies curvature and path-length constraints. Trajectories with excessive backtracking or an inaccurate terminal position or heading are rejected.

B.4 SYNCHRONIZED GEOMETRIC SUPERVISION

At each trajectory step, the renderer records the egocentric RGB observation and the planar navigation action. The MAP position is projected into the current camera as

$$\mathbf{u}_t^{\mathrm{MAP}} = \Pi\Big(T_{c_t \leftarrow w}\,[x_{\mathrm{MAP}} \quad y_{\mathrm{MAP}} \quad z_{\mathrm{MAP}} \quad 1]^\top\Big), \tag{7}$$

where $T_{c_t \leftarrow w}$ maps world coordinates to the camera frame and Π applies the camera intrinsics and perspective division. Target-instance masks are converted to image-plane bounding boxes. RGB frames, navigation actions, projected MAP coordinates, target boxes, scene and target identifiers, and quality flags are stored with the same timestamps for frame-level alignment.

C MAP-BENCH DETAILS

C.1 BENCHMARK PROTOCOL

MAP-Bench contains 300 targets in 109 scenes. Its training split contains 35,454 trajectories from 6,732 observation poses, with 7,184,098 RGB frames at 15 Hz. Its evaluation split contains 15K trajectories. Each target has five equivalent English instructions, and one of them is sampled for each episode with a fixed random seed. Each episode specifies a reference MAP and an initial pose, and episodes are grouped by their initial distance to the reference MAP into a short range below 1.5 m, a mid range from 1.5 to 3.5 m, and a long range of at least 3.5 m.

Split construction. The two splits share targets but not initial poses. To prevent nearly identical initial observations from appearing in both splits, we cluster feasible initial poses in $SE(2)$ using

the normalized distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\left(\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2}{r_p}\right)^2 + \left(\frac{\mathrm{wrap}(\theta_i - \theta_j)}{r_\theta}\right)^2}, \tag{8}$$

where r_p and r_θ set the spatial and angular clustering scales. Entire clusters are assigned to one split, so every evaluation episode starts from a held-out initial pose. All MAP-Bench targets are further excluded from the rest of MAP-Data, although MAP-Data may contain other targets in the benchmark scenes. Appendix F.1 evaluates unseen scenes and targets zero-shot on real robots.

Evaluation. All models are evaluated in closed loop from the same held-out initial poses with the same instructions. A predicted action chunk is regarded as near zero if at least 80% of its predicted relative poses, including the last one, satisfy $\|(x_t, y_t)\|_2 \leq 0.05\,\mathrm{m}$ and $|\theta_t| \leq 0.05\,\mathrm{rad}$. For models that predict velocities, the test is applied to the base displacement integrated over the chunk. An episode stops after three consecutive near-zero chunks, and the third chunk is not executed. Uni-NaVid stops when it predicts its stop token. An episode also ends after twice the steps of its reference trajectory. The reached terminal pose is then compared with the reference MAP using Eq. 1.

C.2 Training Protocol

Common protocol. All models are trained on the MAP-Bench training split for 40,000 steps on 16 NVIDIA H200 GPUs. All models observe only the forward RGB camera, and each keeps its native action interface, into which the navigation labels are converted. Only UniWAM receives the bounding-box and MAP auxiliary targets. $\pi_{0.5}$, GR00T N1.7, InternVLA-N1, and Uni-NaVid start from their public pretrained checkpoints, and FastWAM starts from Wan2.2-TI2V-5B (Wan Team et al., 2025) as in its official implementation. Apart from the shared data and budget, each baseline follows its official fine-tuning recipe, including its optimizer, learning rate, and schedule.

UniWAM variants. UniWAM-Base is trained only on the training split, so its comparison with the baselines isolates the model design. UniWAM-Scaled keeps the architecture and budget and samples MAP-Data and the training split at a 7:3 ratio, so its gain over UniWAM-Base measures how much the decoupled navigation stream benefits from large-scale, diverse MAP supervision.

Baselines. The VLA models $\pi_{0.5}$ (Black et al., 2025a) and GR00T N1.7 (Bjorck et al., 2025) are general-purpose policies that generate robot actions from images and language. Both are fine-tuned with all parameters and observe only the current frame. The VLN models Uni-NaVid (Zhang et al., 2025a) and InternVLA-N1 (InternNav Team, 2025) are designed for language-conditioned semantic navigation. InternVLA-N1 starts from the released DualVLN checkpoint (Wei et al., 2026) and, as in its official recipe, keeps System 2 frozen and trains only the System 1 trajectory generator, using the current frame and up to four history frames. Uni-NaVid is fine-tuned with all parameters from the Uni-NaVid-7B checkpoint, using the current observation and up to three earlier observations sampled every 0.5 s. Because its native outputs are discrete forward and turning primitives, its labels are the primitive sequences that best fit the future trajectory, found by dynamic programming. The world-action model FastWAM (Yuan et al., 2026b) is the baseline closest to UniWAM in formulation. It shares the data, visual inputs, action targets, horizon, optimizer, and schedule of UniWAM-Base, but follows its own official architecture and does not use the stream-specific interfaces, the causal action mask, or the image-plane auxiliary supervision.

D UniWAM Details

D.1 Network, Optimization, and Inference

Network and optimization. UniWAM uses the Wan2.2-TI2V-5B video backbone, video VAE, and text conditioning. The action DiT has 30 transformer layers and a hidden dimension of $D_a = 1024$, compared with $D_v = 3072$ for the video DiT. The action horizon is $H = 32$. Sampling every fourth image from a 33-frame observation window yields nine RGB frames, which are encoded into three temporal VAE latents. Camera views are arranged into a 384×320 canvas with a main view and wrist panels. Manipulation actions use the 20-dimensional camera-frame dual-arm representation of Appendix D.2, and for single-arm data the unused arm dimensions are zero-padded and masked from

the loss. We optimize with AdamW, an initial learning rate of 10^{-5}, and cosine decay. Inference uses a deterministic schedule with ten Euler denoising steps and no classifier-free guidance.

Infrastructure acceleration. Action inference is latency-bound rather than compute-bound. Each denoising step of the action DiT processes only a few action tokens, so in eager execution the GPU often finishes a small kernel before the CPU has issued the next one, and much of the step time is spent on Python dispatch and kernel launches instead of computation. To address this, we reduce the CPU work per step so that the GPU stays busy. First, `torch.compile` traces the VAE encoder, the visual-cache prefill, and the cached action-denoising path into static graphs. It fuses consecutive element-wise operations such as normalization, activation, and residual addition into fewer kernels, which reduces both the number of launches and the memory traffic between them. Second, CUDA graph replay records the complete kernel sequence of a denoising step once and replays it with a single CPU call, so the per-step launch cost no longer grows with the number of kernels. Third, the video tokens of the current observation are encoded once, and their keys and values are cached at every layer. The ten Euler steps then run only over the action tokens and read the cached visual context instead of recomputing it. Fourth, batch packing places the navigation and manipulation rows in the same forward pass, so both streams share one launch sequence and each kernel operates on a larger batch, which raises GPU utilization. During training, `torch.compile` likewise captures the mixed video-action transformer layers. Together, these optimizations reduce dual-stream inference latency on one NVIDIA H200 from about 220 ms to below 60 ms, and distributed training on 16 H200 GPUs exceeds 600 samples per second. Asynchronous execution overlaps the remaining inference with executing committed actions, so the robot does not pause during inference.

D.2 Action Representations

Navigation. Many unified policies represent base motion by linear and angular velocities (v, ω), for example Mobile ALOHA (Fu et al., 2024). UniWAM instead predicts planar pose increments $(\Delta x, \Delta y, \Delta\theta)$ relative to the base pose at the start of the action chunk, and a chunk is a sequence of such waypoints over the horizon. This choice addresses a property of base velocity labels. Mobile bases run under conservative velocity limits with brief acceleration phases, so recorded linear and angular velocities lie mostly either near zero or at the velocity limits. As a result, the velocity labels follow a strongly bimodal distribution, much like the binary open and close states of a gripper. Neural networks learn continuous, smoothly varying targets more easily than such bimodal ones, which are known to be hard to fit with a continuous action model (Chi et al., 2023). Converting base motion into pose increments $(\Delta x, \Delta y, \Delta\theta)$ turns the bimodal velocity labels into continuous values, because the increments accumulate the velocities over the chunk and vary smoothly along an approach trajectory. Pose targets have two further advantages. They state where the base should be, which matches the goal of navigation, whereas small velocity errors accumulate when integrated over long approaches. They also remain easy to execute, because a low-level controller converts them into target velocities for the base. The ablation in Table 1 confirms this design under otherwise identical training: replacing pose increments with velocity commands raises the Overall errors from 12.5 cm and 3.24° to 16.3 cm and 4.17°, and the long-range position error from 21.0 to 30.9 cm, whereas the short-range error changes only from 7.71 to 8.30 cm. The degradation grows with distance, consistent with velocity errors that accumulate over time.

Manipulation. Each arm is represented by 10 dimensions, namely a 3D end-effector position, a 6D rotation, and a 1D gripper state, so the dual-arm action has $(3 + 6 + 1) \times 2 = 20$ dimensions. The 6D rotation consists of the first two columns of the rotation matrix, and the full matrix is recovered by Gram–Schmidt orthogonalization (Zhou et al., 2019). Unlike Euler angles and quaternions, this representation is continuous, so nearby rotations have nearby targets. Positions and rotations are expressed in the current main camera frame, as analyzed in Appendix F.7. For single-arm embodiments, the dimensions of the absent arm are zero-padded and masked from the loss.

D.3 Attention Mask

The video DiT reads visual tokens, while language and robot state enter through cross-attention. Current-observation tokens cannot read future frames, future-frame tokens may read the full visual sequence, and action tokens cannot read future video tokens. Within an action row, each action token may read the current visual context, the learned tokens, itself, and earlier action tokens, but

not later actions. Learned tokens may read the current visual context and one another, but not action tokens. Attention never crosses batch rows, so the two streams interact only through shared weights.

D.4 Stream Sampling

Candidates are drawn according to Eq. 3. Duplicate or temporally adjacent windows are rejected within a batch, and windows from the same episode must have start frames separated by at least δ frames. The selected examples are grouped by stream and packed with Eq. 2. The stream-specific batch sizes may differ, and navigation and manipulation rows need not originate from the same trajectory. Under joint whole-body sampling, which Sec. 4.5 uses as an ablation, each sampled window instead supervises both streams, so the proportions of active and idle windows follow the phase durations of the complete trajectories rather than the chosen window weights.

D.5 Flow Matching and Masked Supervision

For a clean video-latent sequence or normalized action chunk $\mathbf{x}$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and noise level $\sigma \in [0, 1]$, flow matching uses

$$\mathbf{x}_\sigma = (1 - \sigma)\mathbf{x} + \sigma\boldsymbol{\epsilon}, \qquad \mathbf{v}^* = \boldsymbol{\epsilon} - \mathbf{x}. \tag{9}$$

Video and action noise levels are sampled independently for each example as $\sigma = \kappa u/[1+(\kappa-1)u]$ with $u \sim \mathcal{U}(0, 1)$ and $\kappa = 5$. The current observation remains clean, and only future video latents contribute to $\mathcal{L}_{\text{vid}}$. The action and auxiliary dimensions of one action vector share a noise level. For prefix-conditioned training, positions $1 \le t \le P$ are replaced by clean committed actions and excluded from supervision. With probability 0.7, P is sampled uniformly from $\{6, \dots, 12\}$, and otherwise $P = 0$. For supervised group g, let $\mathcal{F}^g_{b,t}$ contain the valid features at position t of example b, and let $\mathcal{I}^g_b$ contain the non-prefix, non-padded positions with at least one valid feature. The per-example loss is

$$\ell^g_b = \frac{\omega(\sigma_b)}{|\mathcal{I}^g_b|} \sum_{t \in \mathcal{I}^g_b} \frac{1}{|\mathcal{F}^g_{b,t}|} \sum_{j \in \mathcal{F}^g_{b,t}} \left(\widehat{v}_{b,t,j} - v^*_{b,t,j}\right)^2, \tag{10}$$

where $\omega(\sigma_b)$ is a loss weight that depends on the noise level, and each group loss is averaged over the examples with valid supervision, using counts aggregated across workers.

D.6 End-Effector Auxiliary Labels

The manipulation auxiliary targets are the image-plane tracks of both end effectors in the main view. They are generated from recorded robot states and camera calibration in three steps. Algorithm 1 projects each end effector into the main view, Algorithm 2 decides in which frames the projected point is visible, and Algorithm 3 runs both steps on every episode and validates the output.

Calibrated geometry. For each arm, Algorithm 1 decodes the end-effector pose in the native base frame of that arm, maps it into a common reference frame with the per-arm transform $M^i_{\text{ref}\leftarrow b_i}$, and then into the main camera frame with $E_{c\leftarrow\text{ref}}$. Two points are tracked per arm: the flange of the sixth joint, J6, with zero offset, and a point on the gripper with offset $\mathbf{o}_{\text{vis}}$, the part seen in the image. Both points are projected with the pinhole model π_K. A projected gripper point is geometrically valid if it lies in front of the camera, falls inside the image, and belongs to the manipulation interval given by the operation mask m^{op}. The calibration is checked before projection, and a record of its inputs is stored with the labels, so every label can be traced to its calibration.

Appearance-gated visibility. A geometrically valid point may still be hidden, for example by the other arm or by an object. Algorithm 2 thus checks the image content at the projected point. It encodes every k-th frame with DINOv2 (Oquab et al., 2023), takes the patch token at the projected gripper point, and scores it by its highest cosine similarity to positive end-effector prototypes minus its highest similarity to negative background prototypes. The scores are interpolated only within contiguous valid segments, thresholded with hysteresis so that the visibility does not flicker near the threshold, and cleaned by filling short gaps and removing short runs. The output keeps the coordinates only in visible frames and labels each frame as visible, valid but occluded, or invalid.

Algorithm 1 Calibrated main-view end-effector geometry

Require: states $S_{0:T-1}$; main-view video V; intrinsics K; reference-to-camera transform $E_{c\leftarrow\mathrm{ref}}$; per-arm base-to-reference transforms $\{M^i_{\mathrm{ref}\leftarrow b_i}\}_{i\in\mathcal{A}}$; operation mask m^{op}; point offsets $\mathbf{o}_{\mathrm{J6}} = \mathbf{0}$ and $\mathbf{o}_{\mathrm{vis}}$; arms $\mathcal{A} = \{\mathrm{left}, \mathrm{right}\}$

Ensure: flange and gripper image points $P^{\mathrm{J6}}_{0:T-1}$ and $P^{\mathrm{vis}}_{0:T-1}$; geometric validity mask $m^{\mathrm{geo}}_{0:T-1}$; provenance record $\mathcal{R}$

Notation: $h(\mathbf{y}) = [\mathbf{y}^\top, 1]^\top$; $\mathrm{xyz}(\cdot)$ drops the homogeneous coordinate; $\pi_K([X,Y,Z]^\top) = [f_xX/Z + c_x,\ f_yY/Z + c_y]^\top$

$\mathcal{R} \leftarrow$ ValidateCalibration$(K, E_{c\leftarrow\mathrm{ref}}, \{M^i_{\mathrm{ref}\leftarrow b_i}\})$
for $t = 0, \ldots, T-1$ and $i \in \mathcal{A}$ **do**
  $(\mathbf{x}^i_t, R^i_t) \leftarrow$ DecodeState(S_t, i) ▷ pose in the arm base frame
  for $\rho \in \{\mathrm{J6}, \mathrm{vis}\}$ **do**
    $\mathbf{c}^{i,\rho}_t \leftarrow \mathrm{xyz}\left(E_{c\leftarrow\mathrm{ref}}\, M^i_{\mathrm{ref}\leftarrow b_i}\, h(\mathbf{x}^i_t + R^i_t\mathbf{o}_\rho)\right)$ ▷ point in the camera frame
    $P^{i,\rho}_t \leftarrow \pi_K(\mathbf{c}^{i,\rho}_t)$
  end for
  $m^{\mathrm{geo},i}_t \leftarrow \mathbf{1}[(\mathbf{c}^{i,\mathrm{vis}}_t)_z > 0] \cdot \mathbf{1}[P^{i,\mathrm{vis}}_t \in [0, W_{\mathrm{img}}) \times [0, H_{\mathrm{img}})] \cdot m^{\mathrm{op}}_t$
end for
$\mathcal{R} \leftarrow \mathcal{R} \cup$ HashInputs$(S, V, K, E_{c\leftarrow\mathrm{ref}}, \{M^i_{\mathrm{ref}\leftarrow b_i}\})$
return $P^{\mathrm{J6}}, P^{\mathrm{vis}}, m^{\mathrm{geo}}, \mathcal{R}$

Algorithm 2 Appearance-gated visibility of the end-effector track

Require: main-view video V; gripper points P^{vis}; geometric validity mask m^{geo}; DINOv2 encoder Φ; positive and negative prototypes $\mathcal{Q}^+, \mathcal{Q}^-$; stride k; thresholds $\eta_{\mathrm{in}} > \eta_{\mathrm{out}}$; minimum gap and run lengths $n_{\mathrm{gap}}, n_{\mathrm{run}}$; arms $\mathcal{A}$

Ensure: visibility mask $z_{0:T-1}$; visible-only coordinates $\widetilde{P}_{0:T-1}$; quality labels $q_{0:T-1}$

$U^+ \leftarrow$ EncodePrototypes$(\Phi, \mathcal{Q}^+)$, $U^- \leftarrow$ EncodePrototypes$(\Phi, \mathcal{Q}^-)$
for $t \in \{0, k, 2k, \ldots\}$ with $t < T$ **do**
  $F_t \leftarrow \Phi(V_t)$
  for $i \in \mathcal{A}$ **do**
    if $m^{\mathrm{geo},i}_t = 1$ **then**
      $\mathbf{f} \leftarrow$ PatchToken$(F_t, P^{i,\mathrm{vis}}_t)$ ▷ token at the projected point
      $\mu^i_t \leftarrow \max_{\mathbf{u}\in U^+} \cos(\mathbf{f}, \mathbf{u}) - \max_{\mathbf{u}\in U^-} \cos(\mathbf{f}, \mathbf{u})$
    else
      $\mu^i_t \leftarrow \mathrm{NaN}$
    end if
  end for
end for
$\bar{\mu} \leftarrow$ InterpolateWithinSegments(μ, m^{geo})
$z \leftarrow$ Hysteresis$(\bar{\mu}, \eta_{\mathrm{in}}, \eta_{\mathrm{out}}) \cdot m^{\mathrm{geo}}$
$z \leftarrow$ Morphology$(z, m^{\mathrm{geo}}, n_{\mathrm{gap}}, n_{\mathrm{run}})$ ▷ fill short gaps, remove short runs
for $t = 0, \ldots, T-1$ and $i \in \mathcal{A}$ **do**
  $\widetilde{P}^i_t \leftarrow P^{i,\mathrm{vis}}_t$ if $z^i_t = 1$, else NaN
  $q^i_t \leftarrow 1$ if $z^i_t = 1$; 2 if $m^{\mathrm{geo},i}_t = 1$ and $z^i_t = 0$; 0 otherwise ▷ visible, occluded, invalid
end for
return $z, \widetilde{P}, q$

Label generation. Algorithm 3 runs the two steps on every episode in parallel. It resolves the calibration of each episode from a registry, writes the labels together with a per-episode report, and validates all outputs against the dataset before training. Coordinates are normalized to the annotation canvas, and only visible points enter the auxiliary loss, while the other points are masked.

Algorithm 3 End-effector label generation for a dataset

Require: episodes $\mathcal{D}$; calibration registry $\mathcal{C}$; point offsets $\mathbf{o}_{\mathrm{J6}}, \mathbf{o}_{\mathrm{vis}}$; visibility parameters $\Theta_{\mathrm{vis}} = (\Phi, \mathcal{Q}^+, \mathcal{Q}^-, k, \eta_{\mathrm{in}}, \eta_{\mathrm{out}}, n_{\mathrm{gap}}, n_{\mathrm{run}})$
Ensure: per-episode labels and reports; validated manifest $\mathcal{M}$

for all episodes $e \in \mathcal{D}$ in parallel **do**
  $(S^e, V^e) \leftarrow \textsc{ReadEpisode}(e)$
  $(K^e, E^e, \{M^{i,e}\}, m^{\mathrm{op},e}) \leftarrow \textsc{ResolveCalibration}(e, \mathcal{C})$
  $(P^{\mathrm{J6},e}, P^{\mathrm{vis},e}, m^{\mathrm{geo},e}, \mathcal{R}^e) \leftarrow$ Algorithm 1
  $(z^e, \widetilde{P}^e, q^e) \leftarrow$ Algorithm 2 with Θ_{vis}
  $\textsc{WriteLabels}(e, P^{\mathrm{J6},e}, P^{\mathrm{vis},e}, \widetilde{P}^e, m^{\mathrm{geo},e}, z^e, q^e, \mathcal{R}^e)$ ▷ labels and report
end for
$\mathcal{M} \leftarrow \textsc{ValidateAll}(\mathcal{D})$ ▷ every episode has complete labels
return $\mathcal{M}$

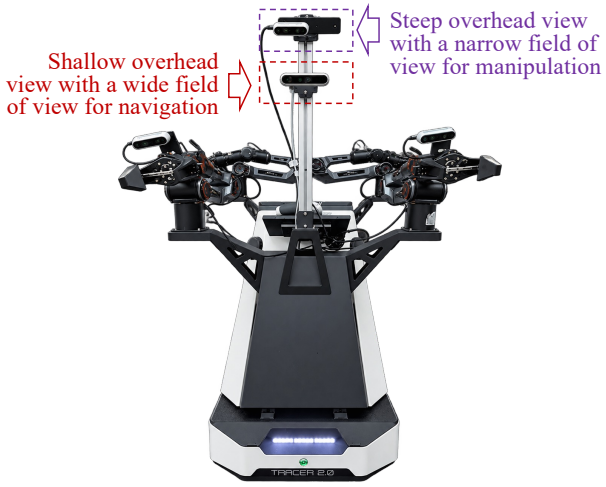

Figure 5: Mobile manipulation platform. A shallow-angle head camera with a wide field of view provides the navigation-stream input, and a steep-angle head camera with a narrow field of view, together with the wrist cameras, provides the manipulation-stream input. In Mobile Manipulation, the navigation stream also receives the wrist cameras. The two inputs form the navigation and manipulation rows of a mixed-stream batch, which UniWAM processes in one forward pass.

E Real-Robot Setup

E.1 Platforms and Stream-Specific Views

We evaluate on four embodiments: AgileX Cobot Magic, AgiBot G2, dual-arm Franka Research 3, and an AgileX single-arm platform. The 13 MAP Navigation tasks run on both the Cobot Magic and the AgiBot G2. The four Mobile Manipulation tasks and Manipulation task 5 run on the Cobot Magic, Manipulation tasks 1 to 3 and 7 on the dual Franka, and Manipulation tasks 4 and 6 on the AgileX single arm. Fig. 5 shows the mobile platform, whose two head cameras realize the stream-specific observations of UniWAM. The wide-view camera captures distant targets and free space for navigation, and the downward-facing camera captures the workspace for manipulation.

E.2 Training Protocol

Common protocol. All methods are trained on the same demonstrations of the 24 evaluation tasks with the same task instructions and are evaluated from the same recorded initial configurations. Each MAP Navigation task has 50 demonstrations on each of its two embodiments, and each Mobile Manipulation or Manipulation task has about 200, with 400 for selected challenging tasks such as six-cup stacking. All methods observe the same cameras. In MAP Navigation, every method observes only the wide-view navigation head camera. In Mobile Manipulation, all methods observe

four cameras: the navigation head camera, the downward-facing manipulation head camera, and the two wrist cameras. UniWAM assigns them to its streams: the navigation stream observes the navigation head camera and the two wrist cameras, and the manipulation stream observes the manipulation head camera and the two wrist cameras. $\pi_{0.5}$ and GR00T N1.7 receive all four views as a single multi-view observation. In Manipulation, all methods observe the same cameras of each embodiment. Table 6 summarizes the remaining controls. Apart from the shared data, each baseline follows its official fine-tuning recipe, including its optimizer, learning rate, and schedule.

UniWAM variants. UniWAM-Base uses only these demonstrations and serves as the matched comparison with the baselines. UniWAM-Scaled adds MAP-Data to the navigation stream, sampling 15 trajectories per navigation target, and adds 402 private manipulation tasks with 30 demonstrations each to the manipulation stream. These tasks are disjoint from the evaluation tasks. Large-scale and task data are sampled with probabilities 0.7 and 0.3, so UniWAM-Scaled measures how far the decoupled streams benefit from large-scale, diverse data. In both variants, windows of mobile-manipulation trajectories in which the arm is idle during base motion are sampled at 20% of their original probability, and windows in which the base is idle during manipulation at 15%. UniWAM updates all parameters except the video VAE and the text encoder, which stay frozen as in other world-action models built on Wan2.2 (Fan et al., 2026). The image-plane auxiliary targets are outputs of the UniWAM architecture and are computed automatically from the geometry of the same trajectories without additional annotation. All variants in the architecture comparison of Table 3 are trained with them, so that comparison isolates the effect of the architecture.

Baselines. $\pi_{0.5}$ (Black et al., 2025a) and GR00T N1.7 (Bjorck et al., 2025) start from their public checkpoints and keep their native action interfaces. For the four-camera Mobile Manipulation setting, GR00T N1.7 uses a bimanual embodiment configuration similar to that of ALOHA, with one video key per camera. Both baselines update all parameters. For MAP Navigation, each baseline is trained as one model on all 13 tasks for each of the Cobot Magic and the AgiBot G2, because these tasks share the action space and differ mainly in the target. For the other tasks, each baseline is fine-tuned separately for each embodiment on the Mobile Manipulation and Manipulation tasks of that embodiment. We also co-trained each baseline on all 24 tasks, as for UniWAM, but this performed worse than per-embodiment training for both $\pi_{0.5}$ and GR00T N1.7, which we attribute to their large-scale robot pretraining. Hence, we report their per-embodiment models. UniWAM has no robot pretraining and is designed for co-training, so each variant is one model for all tasks and embodiments. Every baseline model is trained for 50K steps on 8 GPUs, as listed in Table 6. For each embodiment, each baseline uses the checkpoint that performed best in our real-robot tests.

Table 6: Real-robot training settings. Steps and GPUs are given per model. Each baseline trains one MAP Navigation model per mobile robot and one model for the remaining tasks of each embodiment, whereas each UniWAM variant is one model for all tasks and embodiments.

	$\pi_{0.5}$	GR00T N1.7	UniWAM-Base	UniWAM-Scaled
Initialization	$\pi_{0.5}$ base checkpoint	GR00T N1.7-3B checkpoint	Wan2.2-TI2V-5B	Wan2.2-TI2V-5B
Robot pretraining	yes	yes	none	none
Frozen modules	none	none	VAE, text encoder	VAE, text encoder
Task demonstrations	24 tasks	24 tasks	24 tasks	24 tasks
Additional data	none	none	none	MAP-Data and 402 tasks
Image-plane auxiliary outputs	✗	✗	✓	✓
Steps per model	50K	50K	20K	30K
GPUs per model	8	8	32	64

E.3 Evaluation Protocol

MAP Navigation. Each trial starts at a random distance of 1–5 m from the reference MAP with a heading offset within $\pm 45^\circ$. The robot runs until it stops under the rule of Appendix C.1. The reference MAP of each task is the parking pose marked during data collection. In each trial, the robot starts at this pose and is moved to a random initial pose, and the onboard state estimator records both motions, so the reached pose is scored against the same reference.

Mobile Manipulation. Each trial consists of navigation followed by manipulation. When the base stops, E_{pos} and E_{head} are computed against the reference MAP, and the robot then executes the manipulation stage. UniWAM runs both streams in one batch throughout each trial: during navi-

gation the manipulation stream keeps the arms still, and during manipulation the navigation stream keeps the base still. During the manipulation stage, the policies still predict small but nonzero base motions, which can disturb precise placements. Therefore, for all methods, a predicted base chunk is not executed when its displacement stays below 0.05 m and 0.05 rad, analogous to the stopping rule of Appendix C.1. Each trial yields a position error, a heading error, and a binary success.

Manipulation. Trials start from a fixed working pose, and a task-specific predicate decides success.

Controls. Each task is evaluated over 40 trials on each embodiment it runs on, and MAP Navigation results are averaged over the Cobot Magic and the AgiBot G2. The initial configurations of these trials, including the initial robot pose and the object placements, are recorded once and reproduced for every method. All methods use the same task definitions, instructions, success criteria, and termination rules. Category results are unweighted means over tasks.

F Additional Results

F.1 Zero-Shot Transfer from MAP-Data to Real Robots

This experiment tests whether MAP-Data alone teaches real-robot docking. Each model is trained only on MAP-Data for 70,000 steps on 32 NVIDIA H200 GPUs, and is deployed directly on the Cobot Magic and the AgiBot G2. We refer to UniWAM trained in this way as UniWAM-Sim. The baselines use the same data and budget and otherwise follow their official recipes. Evaluation uses the 13 real MAP Navigation tasks and the protocol of Appendix E.3, with 40 trials per task on each embodiment, and the errors are averaged over the two embodiments as for the real-data models. No real-robot data, room, or target instance is used in training, so every real-robot test is unseen.

Table 7 shows that UniWAM-Sim stops 18.7 cm and 6.34° from the MAP on average. Without any real-robot data, its position error is lower than that of $\pi_{0.5}$ trained on the real demonstrations, although its heading error remains higher than the 5.15° of that model. The baselines transfer less well: the position errors of $\pi_{0.5}$ and GR00T N1.7 are 2.4 and 1.9 times those of their real-data versions in Table 2, and FastWAM, which shares the video backbone of UniWAM, remains 12.1 cm behind UniWAM-Sim. This shows that navigation learned from MAP-Data transfers to real robots, and the gap to FastWAM is consistent with the stream-specific design and image-plane supervision.

Table 7: Zero-shot real-robot MAP Navigation of models trained only on MAP-Data, averaged over the Cobot Magic and the AgiBot G2. The bottom rows are real-data models from Table 2.

Model	Training data	E_{pos} (cm)	E_{head} ($^\circ$)
$\pi_{0.5}$	MAP-Data only	55.1	11.4
GR00T N1.7	MAP-Data only	35.0	12.9
FastWAM	MAP-Data only	30.8	8.61
UniWAM-Sim	MAP-Data only	18.7	6.34
Reference: trained on real demonstrations			
$\pi_{0.5}$	real task data	23.3	5.15
UniWAM-Base	real task data	15.1	4.91

F.2 Scaling with MAP-Data

This experiment asks whether navigation keeps improving as more MAP-Data becomes available. We train UniWAM on MAP-Bench with a random subset of 25%, 50%, or 100% of the MAP-Data targets, sampling 15 trajectories per target as in UniWAM-Scaled to maximize diversity and excluding every benchmark target. All other settings follow UniWAM-Scaled: the same architecture, the 7:3 mixture of MAP-Data and the MAP-Bench training split, and 40,000 steps on 16 NVIDIA H200 GPUs. Because the mixture ratio and the number of steps are fixed, the 25%, 50%, and 100% runs see the same number of MAP-Data samples and differ only in how many distinct targets and scenes these samples come from. The 100% run is UniWAM-Scaled, and the 0% point is UniWAM-Base.

Table 8 shows that both errors decrease monotonically as more of MAP-Data is used. The gain concentrates at long range, where the position error falls from 21.0 to 14.2 cm, whereas the short-

range error changes by only 0.23 cm. The gain diminishes with more data but remains positive: the last doubling still reduces the Overall position error by 4.4%. To test whether other models benefit equally, we also train $\pi_{0.5}$ with the full MAP-Data under the same mixture and budget. Its Overall errors fall by only 5.7% and 1.0%, compared with 21.2% and 29.0% for UniWAM, and its long-range position error remains at 47.8 cm, compared with 14.2 cm for UniWAM-Scaled. $\pi_{0.5}$ gains mainly at short range, where it reaches 7.42 cm, on par with UniWAM-Scaled. Large-scale MAP supervision thus helps most when navigation is learned as its own stream with image-plane targets.

Table 8: MAP-Bench errors when training with a fraction of the MAP-Data targets under a fixed compute budget. The second column is the fraction of MAP-Data targets from which the MAP-Data samples are drawn. Whenever MAP-Data is used, it is mixed with the MAP-Bench training split at a fixed 7:3 ratio, so all runs with MAP-Data see the same number of MAP-Data samples. 0% denotes training on the MAP-Bench split alone. UniWAM at 0% and 100% are UniWAM-Base and UniWAM-Scaled, and $\pi_{0.5}$ uses the same data and budget. Errors are in centimeters and degrees.

Model	MAP-Data targets	E_{pos} Short	E_{pos} Mid	E_{pos} Long	E_{pos} Overall	E_{head} Overall
UniWAM-Base	0%	7.71	8.85	21.0	12.5	3.24
UniWAM	25%	7.63	8.52	18.1	11.4	2.81
UniWAM	50%	7.52	8.03	15.3	10.3	2.43
UniWAM-Scaled	100%	7.48	7.86	14.2	9.85	2.30
$\pi_{0.5}$	0%	7.98	9.85	50.8	22.9	4.11
$\pi_{0.5}$	100%	7.42	9.45	47.8	21.6	4.07

F.3 Statistical Reliability

Table 9 reports 95% Wilson intervals for the real-robot success rates. Each task has 40 trials, so the category means of Table 2 equal the success rates pooled over the 160 Mobile Manipulation trials and the 280 Manipulation trials of Table 10. On Mobile Manipulation, the intervals of UniWAM-Scaled and $\pi_{0.5}$ do not overlap. On Manipulation, the intervals of UniWAM-Scaled and $\pi_{0.5}$ overlap, as do those of UniWAM-Base and $\pi_{0.5}$ on Mobile Manipulation. Hence, this comparison shows no clear difference between these pairs. UniWAM-Base still docks more accurately than $\pi_{0.5}$. The intervals of UniWAM-Base and its real-robot ablations also overlap, so the ablation conclusions rest on the error metrics together with the success rates: every ablation worsens all six real-robot metrics. These intervals treat trials as independent and ignore the shared initial configurations.

Table 9: Real-robot success rates with 95% Wilson intervals, pooled over 160 Mobile Manipulation trials and 280 Manipulation trials per method, matching the category means of Table 2.

Model	Mobile Manipulation SR (%)	Manipulation SR (%)
$\pi_{0.5}$	62.5 [54.8, 69.6]	71.4 [65.9, 76.4]
GR00T N1.7	39.4 [32.1, 47.1]	36.8 [31.4, 42.6]
UniWAM-Base	63.1 [55.4, 70.2]	44.3 [38.6, 50.1]
UniWAM-Scaled	80.0 [73.1, 85.5]	68.9 [63.3, 74.1]
w/o navigation auxiliary supervision	56.9 [49.1, 64.3]	43.6 [37.9, 49.4]
w/o manipulation auxiliary supervision	58.8 [51.0, 66.1]	39.3 [33.7, 45.1]
Joint whole-body sampling	60.0 [52.3, 67.3]	41.8 [36.2, 47.6]

F.4 Per-Task Real-Robot Results

Table 10 reports the per-task results underlying Table 2. In MAP Navigation, UniWAM-Scaled has the lowest position error on all 13 tasks and the lowest heading error on 11. On the sink and the round side table, UniWAM-Base has the lowest heading error. In Mobile Manipulation, UniWAM-Scaled is best on all four tasks in all three metrics. UniWAM-Base exceeds $\pi_{0.5}$ in success on cube sorting and pouring and trails it by 2.5 points on the other two. In Manipulation, UniWAM-Scaled leads on tasks 4 to 6, and $\pi_{0.5}$ leads on tasks 1 to 3 and 7, with the largest gap of 12.5 points on coffee preparation, where GR00T N1.7 fails every trial and $\pi_{0.5}$ reaches 52.5%.

Table 10: Per-task real-robot results with 40 trials per task on each embodiment. MAP Navigation cells report position error and heading error averaged over the Cobot Magic and the AgiBot G2. Mobile Manipulation cells additionally report the success rate in percent, and Manipulation cells report only the success rate. Avg. is the unweighted mean over the tasks in each category. Bold and underline mark the best and second-best values for each metric.

ID	Task	$\pi_{0.5}$	GR00T N1.7	UniWAM-Base	UniWAM-Scaled
MAP Navigation	E_{pos} (cm) ↓ / E_{head} (°) ↓				
1	Water dispenser	18.9 / 4.83	19.7 / 5.47	13.9 / 4.23	**10.3 / 3.86**
2	Humanoid robot	24.7 / 5.17	18.4 / 6.57	15.3 / 4.84	**12.6 / 4.47**
3	TV cabinet	20.2 / 4.56	14.8 / 5.74	12.1 / 3.76	**8.94 / 3.58**
4	Tool cabinet	25.3 / 5.63	20.2 / 7.18	16.5 / 7.27	**11.7 / 4.06**
5	Cushion	21.8 / 4.69	15.6 / 5.93	13.2 / 4.34	**9.73 / 3.74**
6	Table	28.1 / 5.84	19.7 / 7.46	16.9 / 5.47	**13.7 / 4.86**
7	Sink	16.9 / 4.27	17.3 / 6.34	12.7 / **3.86**	**10.8** / 4.13
8	Potted plant	30.9 / 6.36	23.1 / 8.13	19.7 / 6.58	**14.8 / 5.24**
9	Wall-side cabinet	23.7 / 5.14	18.2 / 6.77	14.8 / 4.79	**11.9 / 4.31**
10	Workstation	24.9 / 5.46	18.8 / 6.86	15.9 / 5.24	**13.1 / 4.72**
11	Black side cabinet	26.4 / 5.91	21.1 / 7.57	17.1 / 5.96	**12.8 / 4.57**
12	Round side table	21.2 / 4.76	15.1 / 5.84	14.5 / **3.94**	**11.8** / 4.24
13	Countertop	19.5 / 4.35	13.9 / 5.24	13.4 / 3.53	**9.41 / 3.46**
	Avg.	23.3 / 5.15	18.1 / 6.55	15.1 / 4.91	**11.7 / 4.25**
Mobile Manipulation	E_{pos} (cm) ↓ / E_{head} (°) ↓ / SR (%) ↑				
1	Place items in storage box	25.6/5.67/65.0	19.2/6.91/42.5	16.7/5.16/62.5	**12.8/4.37/82.5**
2	Sort colored cubes into trays	23.8/5.29/55.0	21.3/7.46/35.0	18.2/5.73/60.0	**13.1/4.46/77.5**
3	Clear cups onto tray	29.1/6.18/70.0	18.7/6.58/47.5	15.9/4.87/67.5	**13.8/4.83/85.0**
4	Pour water into cup	26.3/5.74/60.0	22.4/7.83/32.5	17.6/5.42/62.5	**13.1/4.57/75.0**
	Avg.	26.2/5.72/62.5	20.4/7.20/39.4	17.1/5.30/63.1	**13.2/4.56/80.0**
Manipulation	SR (%) ↑				
1	Stack six-cup pyramid	**75.0**	35.0	45.0	72.5
2	Insert tubes into metal rack	**77.5**	35.0	42.5	67.5
3	Insert tubes into plastic rack	**65.0**	17.5	35.0	60.0
4	Sort medicine boxes and parts	62.5	47.5	42.5	**65.0**
5	Classify colored blocks	80.0	55.0	67.5	**87.5**
6	Place objects into tray	87.5	67.5	70.0	**90.0**
7	Prepare coffee, long-horizon	**52.5**	0.0	7.5	40.0
	Avg.	**71.4**	36.8	44.3	68.9

F.5 Unified Architecture Definitions

All variants in Table 3 share the video DiT initialization, training data, optimization budget, and evaluation protocol. They differ in how the two behaviors are represented, processed, and sampled.

Unified Action Tokens. Following $\pi_{0.5}$ (Black et al., 2025a), GR00T N1.7 (Bjorck et al., 2025), and FastWAM (Yuan et al., 2026b), each action token contains both the navigation and manipulation dimensions, and one action DiT predicts the combined command.

Separate Action Experts. Navigation and manipulation use separate action DiTs that share the video DiT, and each action DiT predicts the command of its own behavior. This variant represents behavior-specific action experts in general and is implemented within our framework, not as a reimplementation of a specific prior model. It has more parameters than our design.

Sequence-Concatenated Streams. This is a controlled variant, not a reimplementation of a prior method. Navigation and manipulation action sequences are concatenated into one token sequence and processed by one action DiT, so the two behaviors occupy different positions of one row.

Mixed-Stream. In the proposed design, the two behaviors keep separate visual inputs, action encoders, and output heads, share the video and action DiTs, and occupy separate batch rows.

The first three variants require navigation and manipulation targets in every sample and are all trained with joint whole-body sampling. The mixed-stream design is trained with both joint and independent sampling to separate the effect of the architecture from that of sampling.

F.6 Learned Tokens

Table 11 varies the number of learned tokens per token bank over $\{0, 4, 8\}$, using the same count for all four banks. Removing the tokens degrades all six metrics. The MAP Navigation errors increase from 15.1 cm and 4.91° to 16.2 cm and 5.38°, and Mobile Manipulation success drops from 63.1% to 61.9%. Explicit token identity helps the shared DiTs distinguish the two streams. Increasing the count to eight changes each success rate by at most one trial, from 63.1% to 63.8% and from 44.3% to 44.6%, and gives mixed changes in the errors. We use four tokens per bank.

Table 11: Real-robot results for the number of learned tokens per bank. † marks the default.

	MAP Navigation		**Mobile Manipulation**			**Manipulation**
Tokens per bank	E_{pos} **(cm)** ↓	E_{head} (°) ↓	E_{pos} **(cm)** ↓	E_{head} (°) ↓	**SR (%)** ↑	**SR (%)** ↑
0	16.2	5.38	17.8	5.48	61.9	43.6
4, ours†	15.1	4.91	17.1	5.30	63.1	44.3
8	14.8	5.02	17.3	5.24	63.8	44.6

F.7 Manipulation Coordinate Frame

Table 12 compares camera-frame and robot-base end-effector targets under multi-embodiment co-training. Camera-frame targets improve success on each reported embodiment and task group and raise the unweighted mean from 48.5% to 55.2%. Following the alignment principle of Qwen-RobotManip (Yuan et al., 2026a), expressing end-effector motion in the current camera frame makes visually similar motions numerically similar across embodiments. The same visual change in the egocentric image then corresponds to similar action targets, regardless of the base frame or arm geometry of each robot, which lets data from different embodiments reinforce one another. This comparison covers co-training on these embodiments and does not evaluate transfer to unseen robots.

Table 12: Manipulation target frame in UniWAM-Base under multi-embodiment co-training. Values are success rates in percent. Cobot Magic uses the Mobile Manipulation tasks, the dual Franka uses Manipulation tasks 1 to 3 and 7, and the AgileX single arm uses Manipulation task 6. Mean is the unweighted mean over the three reported embodiment and task groups.

Target frame	Cobot Magic	Dual Franka	AgileX single arm	Mean
Robot base	55.0	28.1	62.5	48.5
Camera, ours	**63.1**	**32.5**	**70.0**	**55.2**

G Qualitative Results

G.1 Auxiliary Predictions

Navigation. Fig. 6 overlays the target boxes and projected MAP positions that UniWAM predicts over consecutive frames of MAP-Bench episodes. The boxes tightly enclose the target across object categories, scenes, and viewing distances, from large targets that fill much of the view, such as the stove and the door, to smaller targets such as the chairs. The boxes from consecutive frames nearly coincide, so the predictions remain stable over time rather than jittering from frame to frame. The projected MAP lies on the floor directly in front of the manipulation-facing side of the target, at the point where the base should stop. These observations match the ablation in Sec. 4.5: the box keeps the target localized throughout the approach, and the MAP point marks the terminal pose.

Manipulation. Fig. 7 examines whether the predicted 2D end-effector tracks agree with the executed 3D actions. Each curve is the track predicted for one arm in the previous action chunk, and each circle marks the actual end-effector position after the robot executes that chunk. Across both arms, several tasks, and two embodiments, the circles lie on the predicted tracks at the grippers. In other words, the image-plane tracks agree with where the camera-frame actions actually move the end effectors, indicating that the manipulation stream has learned a consistent mapping from its 3D end-effector actions to their pixel locations in the main camera view of each embodiment.

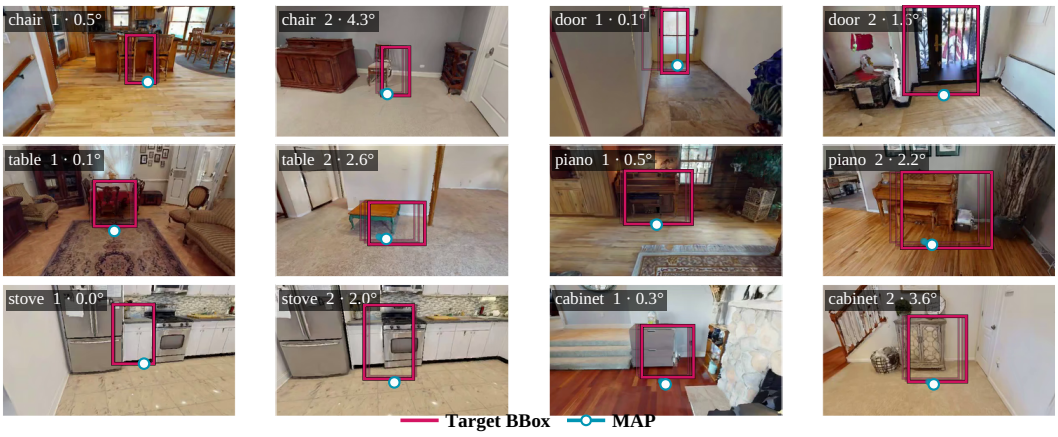

Figure 6: Navigation auxiliary predictions of UniWAM on MAP-Bench, overlaid over consecutive frames of each episode. Magenta boxes are the predicted target boxes, and cyan points are the predicted MAP positions projected into the image, both predicted jointly with the actions.

G.2 MAP-Bench Trajectories

Fig. 8 compares the trajectories of all methods on 24 randomly selected MAP-Bench scenes with initial distances from 1.6 to 4.2 m. The two VLN baselines show the typical failures of semantic navigation. InternVLA-N1 heads to a wrong region in many displayed scenes, for example in panels a to g. Uni-NaVid often reaches the vicinity of the target but stops beside it, as in panels b, f, l, and



Figure 7: Alignment between predicted 2D end-effector tracks and executed 3D actions on several tasks and two embodiments. Curves show the tracks of the two arms predicted in the previous action chunk, and circles mark the end-effector positions reached after executing that chunk.

n. The VLA and WAM baselines reach the target region in most displayed scenes but show visible path deviations or terminal offsets, such as FastWAM in panels j and u and $\pi_{0.5}$ in panel r. Both UniWAM variants follow the reference path closely and end at the MAP in most displayed scenes. The contrast is largest in the last two rows, where all initial distances are at least 3.5 m. There, the VLN baselines fail in most displayed scenes and the other baselines deviate more often, whereas UniWAM remains close to the reference path, in agreement with the long-range gains in Table 1.

G.3 Failure Cases

Grasping between adjacent objects. Fig. 9 shows a failure of UniWAM in the color-sorting task. A pink and a blue cube stand next to each other on the table, and either may be picked next. Instead of committing to one of them, the gripper descends toward the point between the two cubes and closes on the gap. The demonstrations contain grasps of either cube, so the action distribution has two nearby modes. When the modes are close, the deterministic ten-step sampler can produce an action between them rather than on one of them. We observed this failure when candidate objects were close together. Specifying the target object may reduce the ambiguity. **Residual base motion.** During the manipulation stage, the navigation stream predicts motions that are close to but not exactly zero, which can move the base slightly. The near-zero rule of Appendix E.3 suppresses it.

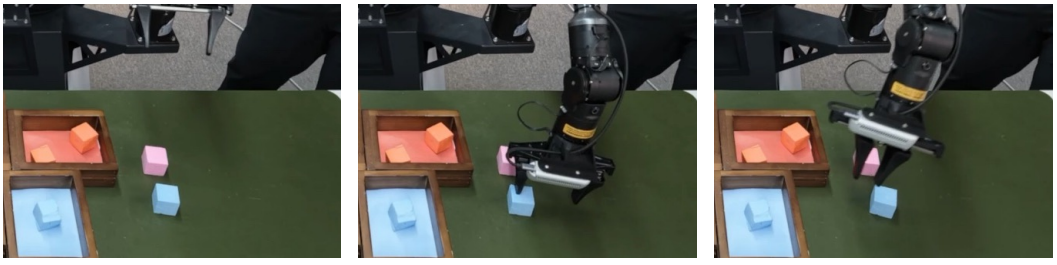

Figure 9: Failure case in the color-sorting task. From left to right: a pink and a blue cube stand next to each other. The gripper descends toward the point between them and closes on the gap.

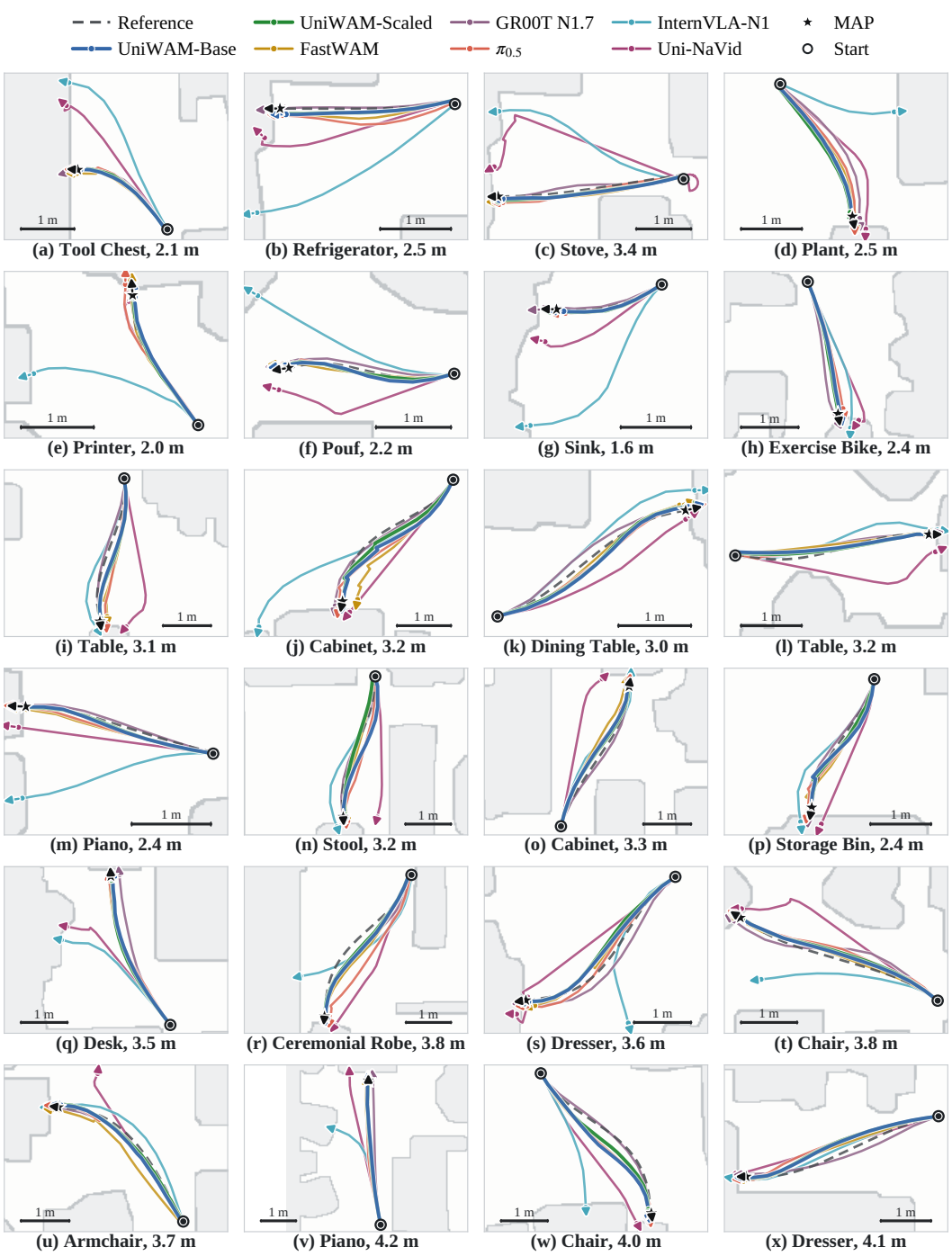

Figure 8: MAP-Bench trajectories of all evaluated methods on 24 randomly selected scenes. Subcaptions give the target category and initial distance. Stars mark the reference MAP, circles the start pose, and arrowheads the terminal heading. The last two rows start at least 3.5 m from the MAP.

G.4 Real-Robot Executions

Fig. 10 shows real-robot executions of UniWAM. The MAP Navigation examples dock at targets of different categories and sizes, the Mobile Manipulation examples show the complete transition from approach to interaction, and the Manipulation examples cover sorting, insertion, and stacking.

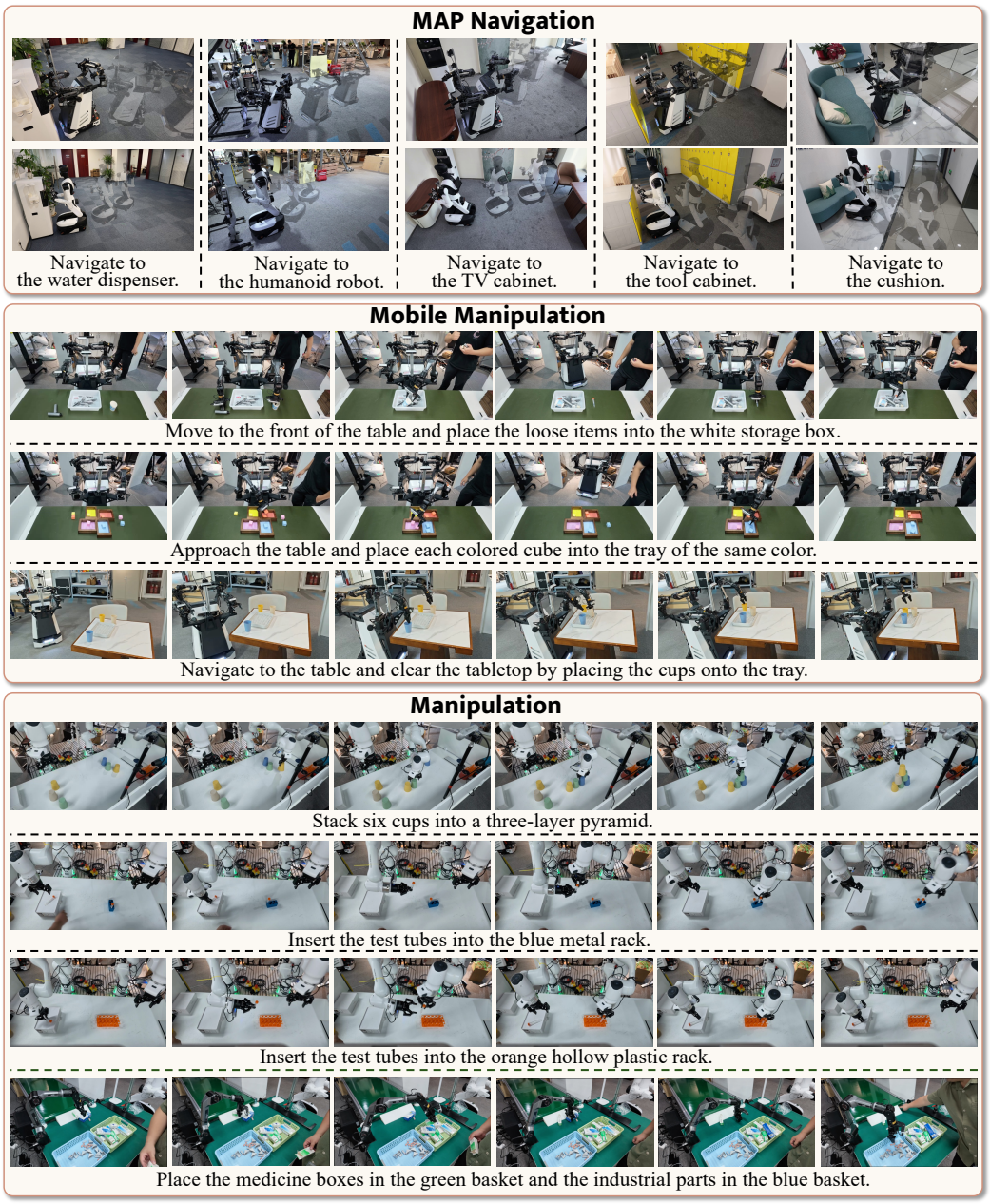

Figure 10: Real-robot executions of UniWAM. The MAP Navigation examples dock at targets of different categories and sizes, the Mobile Manipulation examples show the transition from approach to manipulation, and the Manipulation examples cover sorting, insertion, and stacking.