4D Latent Mapping for Mobile Manipulation Policy Learning

Sunghwan Kim* Byeonghyun Pak* Kehan Long Yulun Tian Nikolay Atanasov

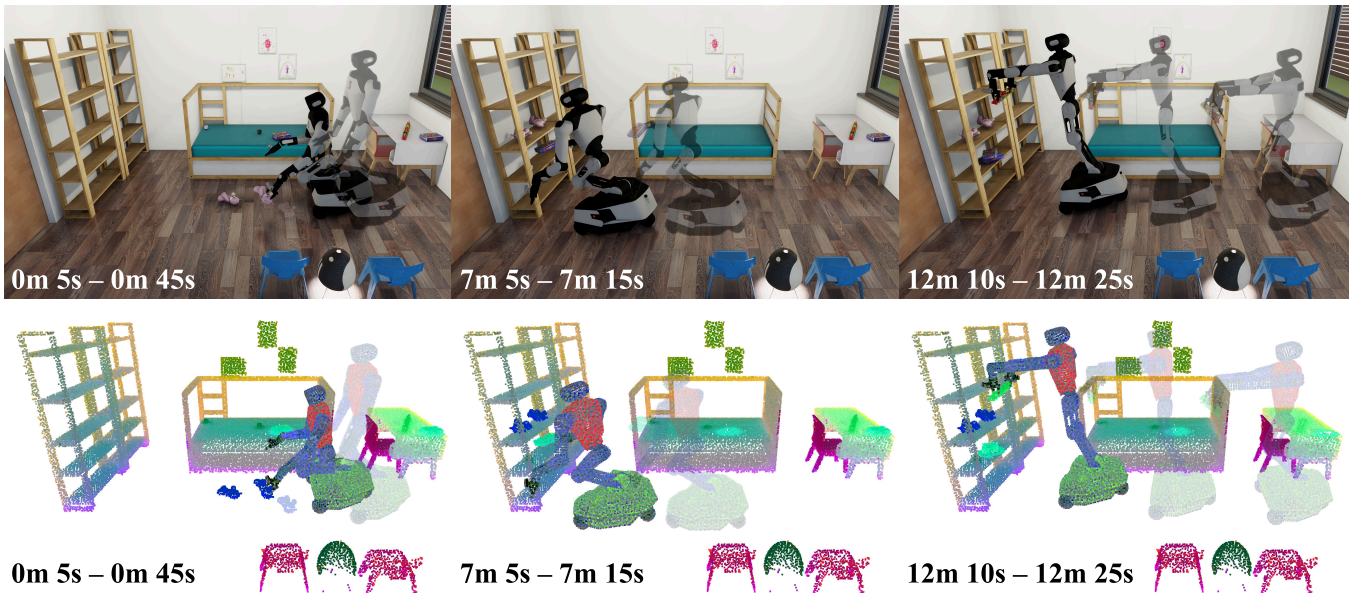

Fig. 1: Long-horizon robot mobile manipulation task utilizing a 4D latent map, with latent features visualized via PCA. The map represents the robot and the environment jointly in a shared latent space and is updated online from egocentric observations and proprioceptive state.

***Abstract*— Long-horizon mobile manipulation requires continual reasoning about self-localization, environmental changes, and task progress, all of which are challenging to infer from image observations alone. In this paper, we show that conditioning a mobile manipulation policy on a *4D latent map* improves spatiotemporal reasoning over long horizons. The map represents the environment and the articulated robot body as neural points in a shared latent space. At each step, we update the map online from egocentric observations and proprioceptive state. To track scene changes, we estimate per-object rigid transforms from 3D keypoint correspondences between consecutive frames and use them to reposition the corresponding environment points. Given a robot kinematic model, forward kinematics updates robot points from the proprioceptive state. To condition a VLA model on this representation, we extract map tokens across multiple coordinate frames and resolutions, providing the policy both local and global context. Experiments on long-horizon mobile manipulation tasks show that the map-conditioned policy outperforms image-only VLA baselines.**

*These authors contributed equally to this work.

We gratefully acknowledge support from NSF CCF-2402689 (ExpandAI), NSF 2120019 (CHASE-CI), and the Agency for Defense Development grant funded by the Korean Government (912A45701).

Sunghwan Kim and Nikolay Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA, e-mails: {suk063, natanasov}@ucsd.edu.

Byeonghyun Pak is with the Agency for Defense Development, Daejeon 34060, South Korea, e-mail: byeonghyun_pak@add.re.kr.

Kehan Long is with SceniX Inc, New York, NY 10011, USA, e-mail: kehan@scenix.ai.

Yulun Tian is with the Robotics Department, University of Michigan, Ann Arbor, MI 48109, USA, e-mail: yulunt@umich.edu.

I. Introduction

Motivation. Recent advances in robot learning have enabled impressive manipulation capabilities in short-horizon tabletop settings. Yet extending these successes to long-horizon mobile manipulation in large environments remains an open challenge. For example, consider a mobile manipulator tasked with tidying up a child's room (Fig. 1). It must pick up scattered toys, navigate around furniture while carrying them, and place them on the shelves of a bookcase. To achieve this, the robot must localize itself, remember which objects have been collected, track how the scene has changed, and decide which subgoal to pursue next. These requirements can be formulated as three questions for the robot: *Where am I? What has changed around me?* and *How far along am I in my task?* Answering them jointly requires *coherent spatiotemporal reasoning*: maintaining a unified model of the robot's motion and the evolving environment over long horizons.

Related Work. Vision-language-action (VLA) models have demonstrated strong potential for mobile manipulation by mapping visual observations and language instructions directly to robot actions [1], [2], [3], [4], [5], [6], [7]. However, most VLA policies encode spatial and temporal context only implicitly, without an explicit mechanism for maintaining information beyond the current observation or a short history. This limitation becomes critical in mobile manipulation, where action prediction depends on long-term context that evolves with the robot's interactions.

Recent work explores persistent memory to extend embodied agents beyond their current observations. One line builds modular mobile manipulation systems that integrate perception, navigation, planning, and manipulation [8], [9]. In this paradigm, persistent scene representations such as dynamic open-vocabulary 3D scene graphs [10], online spatio-semantic object memories [11], task-relevant scene-graph abstractions [12], and predictive world models [13] provide structured spatial grounding for long-horizon reasoning. These approaches provide explicit, interpretable scene memory, but they often abstract the scene into objects, symbols, or high-level planning states. Such abstractions support semantic reasoning, but may discard dense geometry and robot-environment interaction cues needed for continuous visuomotor control.

A second line of work embeds memory directly into policy architectures. Some methods condition policies on episodic memories, history tokens, retrieval buffers, or language-indexed long-term memories [14], [15], [16]. These mechanisms extend temporal context, but often leave spatial grounding implicit: the policy must infer which past observations are relevant to the current robot state and subgoal. More spatially grounded approaches condition policies on 3D feature maps distilled from vision foundation models, including semantic reconstruction-based feature maps [17], fused 3D latent maps [18], and 3D-aware world models that generate future observations as planning targets [19]. These maps provide spatial memory of object and goal locations. However, existing dense feature-map approaches typically represent only the environment with static spatial supports, such as voxel grids or point clouds, whose geometry does not update as the robot interacts with the scene. They also omit the robot body, requiring robot–environment relationships to be inferred indirectly from separate visual or proprioceptive inputs. These limitations motivate dense spatiotemporal memory that evolves with robot interaction, explicitly represents robot-scene spatial relationships, and provides structured context for neural policies.

Contribution. In this paper, we introduce a latent feature mapping formulation that captures scene evolution over long-horizon robot interaction (see Fig. 1). The map represents both the environment and the robot body in a shared latent space, providing a persistent, spatially grounded memory for policy learning. This unified representation supports two complementary forms of spatial reasoning: (I) *Allocentric reasoning*: the map serves as persistent spatial memory, allowing the policy to situate the robot within the global scene while maintaining object and goal locations over time. (II) *Egocentric reasoning*: by embedding the robot body in the same map, the policy can directly reason about robot–scene relationships such as distance, orientation, and reachability.

We instantiate the 4D latent map with *neural points* [20], [21]: 3D points whose learnable latent features are trained to reconstruct dense vision foundation model (VFM) embeddings (*e.g.*, DINOv3 [22]). To track scene changes, we construct 3D keypoint correspondences between consecutive observations, estimate an object-level $\mathrm{SE}(3)$ transform, and update the corresponding neural points. We extend neural points to the robot body by sampling surface points from a robot kinematic model and positioning them via forward kinematics at every time step. We train environment and robot neural points with a shared decoder, encouraging both sets to lie in a common latent space. Notably, we build and maintain the map using only egocentric observations and proprioceptive state (Sec. III). We introduce a VLA policy conditioned on the 4D latent map. It tokenizes the map across multiple coordinate frames and spatial resolutions and conditions a pretrained VLA model on the resulting tokens (Sec. IV). We evaluate our approach on long-horizon mobile manipulation tasks from the BEHAVIOR-1K benchmark [23]. The map-conditioned policy outperforms both image-only VLA baselines and a 3D map-based baseline in average task progress.

Our contributions are summarized as follows.

- We formulate a 4D latent map that represents dynamic scenes with neural points and embeds both the environment and the robot body in a shared latent space.
- We design a VLA policy that tokenizes the 4D map across coordinate frames and spatial resolutions, enabling reasoning over both local and global spatial context.
- We show that the map-conditioned policy improves spatiotemporal reasoning in long-horizon mobile manipulation, outperforming image-only VLA baselines.

II. Problem Formulation

We consider a mobile manipulator robot operating in a workspace $\mathcal{X} \subseteq \mathbb{R}^3$ and executing a task specified by a natural language command ℓ. Let e_ℓ denote the task embedding of ℓ. At each time step τ, the observation o_τ includes RGB-D images from the robot's head and wrist cameras. We additionally assume access to privileged per-view instance segmentation masks, with each pixel labeled by an integer object-instance ID. We encode RGB images into dense semantic embeddings in the target space $\mathcal{Y} \subseteq \mathbb{R}^k$ using a VFM (*e.g.*, DINOv3 [22]). The robot's proprioceptive state s_τ includes the robot base pose and joint configuration. Let $\mathcal{D}^*$ denote the set of expert demonstrations. Because the expert action at time τ may depend on the full observation-state history, we write the expert policy for a generic trajectory as $\pi^*(a_\tau \mid o_{1:\tau}, s_{1:\tau}, e_\ell)$.

We learn a time-indexed map $m_\tau \colon \mathcal{X} \to \mathcal{Y}$ that summarizes the observation-state history $(o_{1:\tau}, s_{1:\tau})$ into a persistent representation of the evolving 3D workspace. Instead of conditioning directly on the full history, we learn a policy $\pi_\phi(a_\tau \mid m_\tau, o_\tau, s_\tau, e_\ell)$ conditioned on the map, current observation, proprioceptive state, and task embedding. For map supervision, we construct a dataset $\mathcal{D}_\tau \subseteq \mathcal{X} \times \mathcal{Y}$ of coordinate–embedding pairs from the current observations and proprioceptive state (see Fig. 2); details are provided in Appendix I. We assume access to an initial-scene dataset $\mathcal{D}_0 \subseteq \mathcal{X} \times \mathcal{Y}$, constructed from observations collected before task execution.

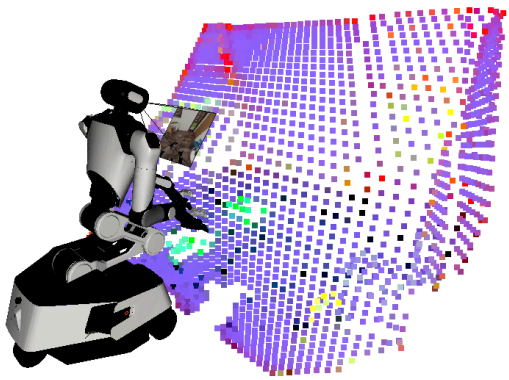

Fig. 2: **Lifted VFM embeddings.** Per-patch VFM embeddings from robot observations are lifted into 3D and visualized by PCA.

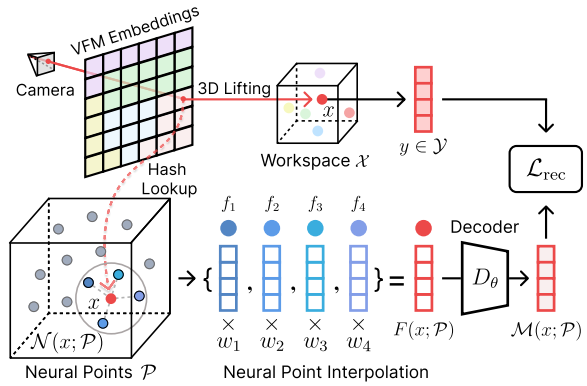

Fig. 3: **Latent feature learning.** Interpolated neural-point features are decoded to reconstruct target VFM patch embeddings.

Problem 1 (Spatiotemporal Feature Mapping)**.** Learn map parameters Θ by minimizing

$$\mathcal{J}_{\text{map}}(\Theta) = \mathbb{E}_{\tau,(x,y)\sim\mathcal{D}_\tau}\left[\mathcal{L}(m_\tau(x;\Theta), y)\right], \tag{1}$$

where τ is sampled from $\{0,\ldots,T\}$ and $\mathcal{L}\colon \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}_{\geq 0}$ is a distance function on the embedding space $\mathcal{Y}$.

Our approach to the mapping problem is to first learn Θ offline using $\mathcal{D}_0$. During execution, we keep Θ fixed and update the map state from $m_{\tau-1}$ to m_τ using o_τ and s_τ.

Problem 2 (Map-Conditioned Policy Learning)**.** Given a time-indexed map m_τ associated with each demonstration in $\mathcal{D}^*$, learn a behavior-cloning policy π_ϕ by minimizing

$$\mathcal{J}_{\text{bc}}(\phi) = -\mathbb{E}_{(o_\tau,s_\tau,a_\tau,\ell)\sim\mathcal{D}^*}\left[\log \pi_\phi(a_\tau \mid m_\tau, o_\tau, s_\tau, e_\ell)\right]. \tag{2}$$

The key difference from standard behavior cloning lies in learning a policy that uses the map as an explicit state input. In the rest of the paper, we formulate a representation and learning approach for spatiotemporal feature mapping (Sec. III), design a policy architecture that conditions action prediction on the map (Sec. IV), and evaluate the approach on long-horizon mobile manipulation tasks (Sec. V).

III. 4D Latent Feature Mapping

To parameterize the map m_τ, we use neural points, where each 3D point carries a learnable latent feature. We maintain separate neural-point sets for the environment and the robot body (Sec. III-A). We train the latent features and a shared decoder to reconstruct dense VFM embeddings using a cosine-similarity reconstruction objective. We further augment this objective with contrastive losses that align features from the same category or object part while separating features from different categories or parts, encouraging semantic structure in the latent space (Sec. III-B). To capture dynamic changes, we update the map by moving environment points with per-instance rigid transforms estimated from 3D keypoint correspondences and updating robot points with forward kinematics from the robot configuration (Sec. III-C).

A. Neural Point Map

We instantiate the latent map as a set of neural points [20]. Because each neural point has an explicit coordinate, it can be repositioned under rigid-body motion, making neural points a natural primitive for dynamic scenes. Formally, $\mathcal{P} = \{(p_i, f_i, c_i)\}_{i=1}^{N}$, where $p_i \in \mathbb{R}^3$ is a 3D position, $f_i \in \mathcal{F}$ is a latent feature vector, $\mathcal{F} \subseteq \mathbb{R}^c$ denotes the latent feature space, and c_i is the instance label associated with point i. At time step τ, the scene is represented in a common world frame by two neural point sets, the environment set $\mathcal{P}_\tau^e$ and the robot set $\mathcal{P}_\tau^r$. The union $\mathcal{P}_\tau = \mathcal{P}_\tau^e \cup \mathcal{P}_\tau^r$, together with the shared decoder D_θ, defines the latent map m_τ as a query function over the workspace.

Environment Points. We initialize the positions of the environment neural points from pre-sampled RGB-D observations of the prior scene and use instance segmentation to assign each point its corresponding instance label. We then voxelize the coordinates and insert one randomly initialized neural point per occupied voxel into a spatial hash table with linear probing [21]. We write $\mathcal{P}_\tau^e = \{(p_{i,\tau}^e, f_i^e, c_i^e)\}_{i=1}^{N_e}$, where $p_{i,\tau}^e$ is the world-frame position of environment neural point i, f_i^e is its latent feature, and c_i^e is its object instance label. During execution, the positions $p_{i,\tau}^e$ are updated to track object motion, while the latent features f_i^e remain fixed.

Robot Points. We also represent the articulated robot body with neural points. Given a robot kinematic model (*e.g.*, a URDF), we sample surface points from the robot link meshes and store them in their corresponding local link frames, yielding $\mathcal{P}^r = \{(u_j, f_j^r, c^r, l_j)\}_{j=1}^{N_r}$. Here, $u_j \in \mathbb{R}^3$ is a sampled surface point expressed in the local frame of link l_j, f_j^r is its latent feature, c^r is a shared robot-body label used to distinguish robot points from environment points, and l_j specifies the link to which the point belongs. Given the proprioceptive state s_τ, forward kinematics gives the world-frame robot point set $\mathcal{P}_\tau^r := \mathcal{P}^r(s_\tau) = \{(p_j(s_\tau), f_j^r, c^r)\}_{j=1}^{N_r}$, where $p_j(s_\tau) = R_{l_j}(s_\tau)u_j + t_{l_j}(s_\tau)$ and $(R_{l_j}(s_\tau), t_{l_j}(s_\tau)) \in \mathrm{SE}(3)$ is the world-frame transform of link l_j.

B. Latent Map Learning

During training, we query the environment and robot neural-point sets separately, while optimizing their features jointly

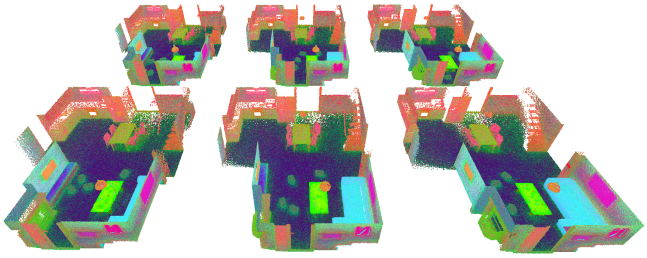

Fig. 4: **Inter-category contrastive learning.** PCA projections show that features from the same category cluster across scenes, whereas features from different categories separate in the latent space.

with the shared decoder D_θ so that both feature sets lie in a common latent space. To ground semantic meaning in the latent features, we use dense embeddings from a pretrained VFM (*e.g.*, DINOv3 [22]) as supervision targets. We obtain coordinate–embedding pairs (x, y) by back-projecting VFM patch embeddings into 3D (see Appendix I for details).

Latent Feature Query. Given a neural point set $\mathcal{P}$, we query the latent feature at an arbitrary location $x \in \mathcal{X}$ by gathering candidate points via a ball query, selecting the K nearest neighbors $\mathcal{N}(x; \mathcal{P})$, and interpolating their latent features:

$$F(x; \mathcal{P}) = \sum_{i \in \mathcal{N}(x;\mathcal{P})} w_i(x; \mathcal{P})\, f_i. \tag{3}$$

Here, $w_i(x; \mathcal{P}) = \mathrm{softmax}(-\|x - p_i\|/\sigma)$ is the interpolation weight, $\sigma > 0$ is a temperature parameter, and the softmax is taken over $\mathcal{N}(x; \mathcal{P})$. The neural point set $\mathcal{P}$ induces a latent map $m(\cdot; \mathcal{P})$ defined as $m(x; \mathcal{P}) = D_\theta(F(x; \mathcal{P}))$. The decoder $D_\theta \colon \mathcal{F} \to \mathcal{Y}$ is an MLP that maps interpolated latent features to the target VFM embedding space. Together, the decoder parameters θ and latent features $\{f_i\}$ constitute the learnable map parameters Θ in Problem 1.

Latent Map Optimization. For the environment, we optimize the latent features of the initial environment point set $\mathcal{P}_0^e$. We train its latent features with a cosine-similarity reconstruction loss, instantiating the primary supervision term in Problem 1 as $\mathcal{L}_{\text{rec}} = 1 - \mathrm{sim}(m(x; \mathcal{P}_0^e), y)$.

For the robot, we train the latent features of $\mathcal{P}^r$ so that each surface point maintains a consistent semantic identity across robot configurations. We construct a robot training dataset from multi-view observations of curated robot states by back-projecting the corresponding VFM embeddings into 3D (see Fig. 8). For each sampled state s, we apply the latent-feature query in (3) to the realized robot point set $\mathcal{P}^r(s)$, obtaining the interpolated robot feature $F^r(x; s) := F(x; \mathcal{P}^r(s))$ at query location x. We then decode this feature with the shared decoder D_θ. Using the robot coordinate–embedding samples, we supervise the decoded features with the same reconstruction objective, $\mathcal{L}_{\text{rec}} = 1 - \mathrm{sim}(D_\theta(F^r(x; s)), y)$. Fig. 3 illustrates the overall latent-mapping framework. More details are provided in Appendix II.

In addition to the reconstruction objective, we introduce contrastive losses to optimize the latent map: $\mathcal{L}_{\text{map}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{inter}}\mathcal{L}_{\text{inter}} + \lambda_{\text{intra}}\mathcal{L}_{\text{intra}}$. The inter-category objective $\mathcal{L}_{\text{inter}}$ pulls same-category features together and pushes different-category features apart across training scenes. Fig. 4 visualizes the resulting category-level clustering across scenes. We treat robot points as an additional category. The intra-instance objective $\mathcal{L}_{\text{intra}}$ [24] clusters features within the same SAM2 [25] segment and separates different segments of the same instance, thereby inducing part-level structure. Fig. 5 visualizes this part-level separation. Additional details are provided in Appendix III.

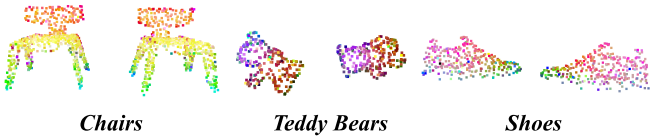

Fig. 5: **Intra-instance contrastive learning.** PCA projections show that features from the same SAM2 segment cluster together, whereas features from different segments of the same instance separate to capture part-level structure.

C. Dynamic Map Update

Environment Updates. For environment updates, we model each object instance as a rigid body and group environment neural points by their stored instance labels. Let $\mathcal{I} \subset \{1, \ldots, N_e\}$ denote the index set of neural points belonging to a given instance. For each instance and consecutive time-step pair $(\tau - 1, \tau)$, we track 2D keypoints [26], [27] within the instance region across the corresponding images and lift the resulting tracks to 3D. We then estimate a rigid transformation $(\hat{R}, \hat{t}) \in \mathrm{SE}(3)$ from these correspondences by solving:

$$\hat{R}, \hat{t} = \underset{R \in \mathrm{SO}(3),\, t \in \mathbb{R}^3}{\arg\min} \sum_{k=1}^{K_c} \left\| q_k^\tau - (R q_k^{\tau-1} + t) \right\|^2. \tag{4}$$

In practice, we obtain a coarse estimate with Fast Global Registration (FGR) [28] and refine it by aligning it to the observed point cloud at time τ with Iterative Closest Point (ICP) [29]. We then update all environment neural points of the instance as $p_{i,\tau}^e = \hat{R} p_{i,\tau-1}^e + \hat{t}$ for all $i \in \mathcal{I}$. The corresponding latent features $\{f_i^e\}_{i \in \mathcal{I}}$ and instance labels $\{c_i^e\}_{i \in \mathcal{I}}$ remain unchanged.

Robot Updates. After executing action $a_{\tau-1}$, we obtain the updated proprioceptive state s_τ. Because robot-body motion is specified by the kinematic model, the update is deterministic. We apply forward kinematics to the link-local robot points to realize $\mathcal{P}_\tau^r = \mathcal{P}^r(s_\tau)$ in the world frame. The latent features and shared robot-body label remain fixed. Together, the updated environment points and robot points form the full scene point set $\mathcal{P}_\tau = \mathcal{P}_\tau^e \cup \mathcal{P}_\tau^r$ used for downstream policy tokenization. Implementation details for map updates are provided in Appendix IV.

IV. Map-Conditioned VLA Policy

We condition a pretrained VLA model on tokens extracted from the latent map. To produce these tokens, we design a map tokenizer that operates across multiple coordinate frames and spatial resolutions, providing both local and global context. Fig. 6 illustrates the overall architecture.

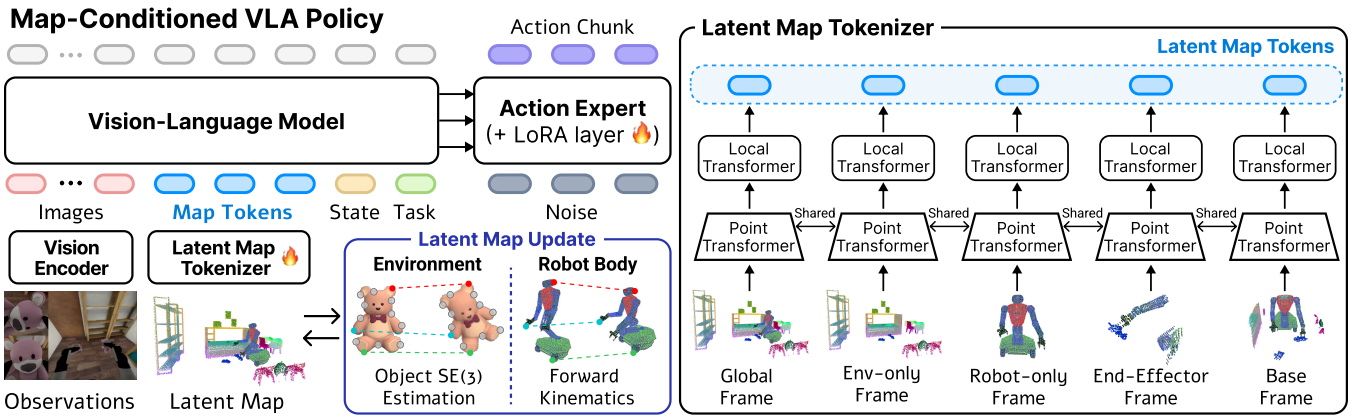

Fig. 6: **Overview of map-conditioned VLA policy.** A latent map tokenizer produces map tokens across multiple coordinate frames and spatial resolutions, including global, environment-only, robot-only, end-effector, and robot-centered tokens. We condition a pretrained VLA model on these map tokens, together with image observations, task embeddings, and proprioceptive state, to predict actions.

Latent Map Tokenizer. We directly tokenize the neural point set $\mathcal{P}_\tau$, since the contrastive objectives induce semantic structure in the latent feature space $\mathcal{F}$. Before tokenization, we use stored instance labels to discard background and structural points (*e.g.*, floors and walls) and retain only task-relevant object instances. A Point Transformer [30] backbone processes the filtered neural point set. The backbone then branches into eight parallel heads, each selecting a spatial subset via ball queries or mask-based selection to produce one map token: (I) three *robot-centered* tokens at increasing radii ($1\,\mathrm{m}$, $2\,\mathrm{m}$, $4\,\mathrm{m}$) around the robot base, capturing local context at multiple scales; (II) two *end-effector* tokens, extracted by $0.5\,\mathrm{m}$-radius ball queries around the left and right grippers, to support grasp reasoning; (III) a *robot-only* token that summarizes the robot body configuration; (IV) an *environment-only* token that summarizes the current environment state; and (V) a *global* token that aggregates all points to support scene-level reasoning. Each branch applies Point Transformer layers followed by attention pooling to produce a single map token. To prevent overfitting, we omit absolute-coordinate positional encoding from both the shared backbone and local branches.

Map-Conditioned Policy Learning. We condition a pretrained VLA model ($\pi_{0.5}$ [3]) on the map tokens. We project the map tokens into the VLA token space, forming the map-token sequence $e_m = [\tilde{z}_1, \ldots, \tilde{z}_8]$. A language encoder maps the instruction ℓ to the task embedding e_ℓ. We concatenate the map-token sequence with image features, proprioceptive state, and the task embedding to form a joint embedding $h_\tau = \mathrm{Concat}(E_I(o_\tau), s_\tau, e_\ell, e_m)$, where E_I is the image encoder. The action expert uses h_τ to predict a velocity field over a noisy action chunk, which is used during inference to produce $\hat{a}_{\tau:\tau+H}$. We instantiate the behavior-cloning objective in Problem 2 with flow matching [31] as

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{(h_\tau, a)\sim\mathcal{D}^*,\, t,\epsilon}\left[\left\|v_\phi(a^{t,\epsilon}, h_\tau, t) - (a-\epsilon)\right\|^2\right], \quad (5)$$

where $a = a_{\tau:\tau+H}$ denotes the expert action chunk, $a^{t,\epsilon} = ta + (1-t)\epsilon$ is the noised action sample along the interpolation path, $\epsilon \sim \mathcal{N}(0, I)$, $t \sim \mathrm{Uniform}(0, 1)$, and v_ϕ is the action expert, which predicts the target velocity field. To preserve pretrained knowledge, we freeze the VLM backbone and vision encoder, insert LoRA [32] modules into the action expert layers, and train only the map tokenizer, projection layer, and LoRA parameters. Additional VLA policy details are provided in Appendix V.

V. Evaluation

We evaluate our approach on long-horizon mobile manipulation tasks that require spatiotemporal reasoning. We use the BEHAVIOR-1K benchmark [23], where the robot navigates large workspaces, completes diverse subgoals, and manipulates multiple target objects. The results show that the map-conditioned policy outperforms image-only VLA baselines. We also demonstrate that our policy generalizes to scene-configuration shifts, including moved goals, added targets, and targets in unvisited regions.

A. Long-Horizon Mobile Manipulation

Benchmark. BEHAVIOR-1K [23] is a benchmark of household bimanual mobile manipulation tasks in OmniGibson [33]. We focus on three tasks: Task 21 (*Collecting Children's Toys*), Task 22 (*Putting Shoes On Rack*), and Task 26 (*Assembling Gift Baskets*). For each task, we report task progress (%) across 20 evaluation configurations, defined as the fraction of completed subgoals in the BDDL task specification. Additional details are provided in Appendix VI.

Baseline. We compare five policy variants that differ in their use of map inputs. (I) PI0.5 (pre) uses the $\pi_{0.5}$ checkpoint from Larchenko *et al.* [34], which was obtained by fine-tuning the original $\pi_{0.5}$ model on all 50 BEHAVIOR-1K tasks. All other variants are initialized from this checkpoint. (II) PI0.5 (ft) is the image-only VLA policy fine-tuned separately on each target task. (III) SBP augments the VLA policy with a prior 3D latent map [18]. (IV) Ours (env) uses our temporally updated environment map but excludes robot neural points. (V) Ours (full) denotes the full map-conditioned VLA policy. For a fair comparison, all variants share the same $\pi_{0.5}$ backbone and training setup; they differ only in the map

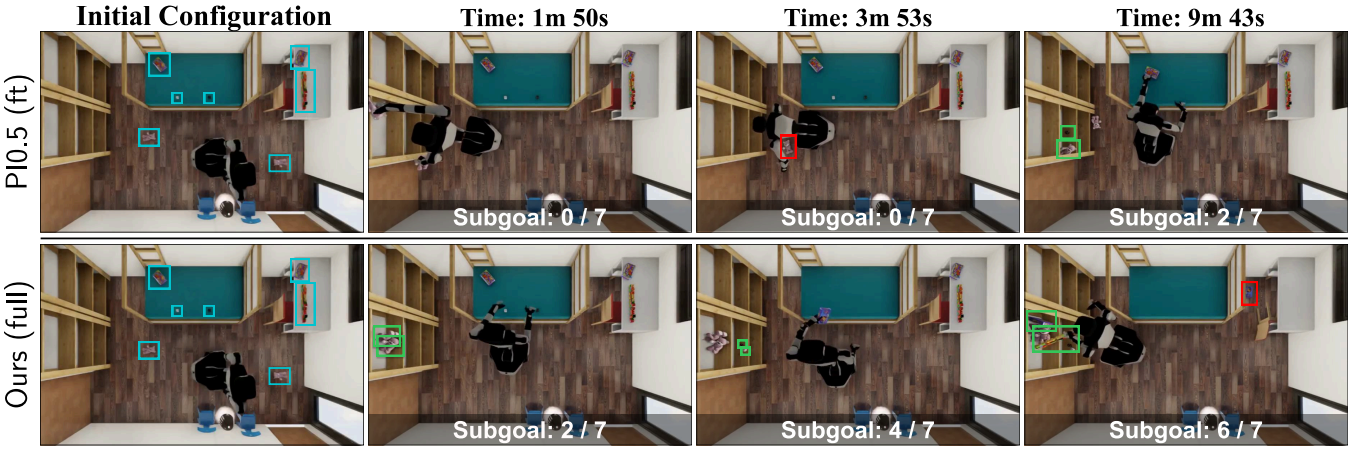

Fig. 7: **Qualitative comparison on long-horizon mobile manipulation.** Compared with PI0.5 (ft), Ours (full) takes more direct trajectories, reaches subgoals faster, and achieves higher task progress.

Method	Task 21	Task 22	Task 26
PI0.5 (pre) [34]	40.0±22.9	43.0±22.8	41.9±22.4
PI0.5 (ft) [34]	50.0±17.2	42.0±20.9	48.1±20.2
SBP [18]	52.9±15.7	49.0±27.0	52.5±13.5
Ours (env)	58.6± 7.7	53.0±19.5	51.9±15.1
Ours (full)	**68.6**±16.7	**61.0**±20.2	**58.8**±11.9

TABLE I: **Task progress (%) on BEHAVIOR-1K [23] tasks.** All methods use the same policy model ($\pi_{0.5}$ [3]) and training setup, differing only in the map tokens provided to the policy.

tokens, if any, provided to the policy. Ours (full) uses all eight map tokens, whereas SBP and Ours (env) use six tokens by excluding the robot-only and global tokens.

Experiment Results. Table I summarizes the quantitative evaluation results. Ours (full) achieves the highest mean task progress across the evaluated tasks. The map-conditioned policies SBP, Ours (env), and Ours (full) achieve higher mean task progress than the image-only baselines PI0.5 (pre) and PI0.5 (ft), supporting the benefit of spatial memory. Among the map variants, Ours (env) outperforms SBP, indicating the benefit of updating the scene representation over time. Including robot neural points further improves performance: Ours (full) outperforms Ours (env), suggesting that explicitly modeling robot–environment spatial relationships strengthens egocentric spatial reasoning. Fig. 7 compares representative rollouts from PI0.5 (ft) and Ours (full). Ours (full) follows more direct trajectories and reaches successive subgoals faster. By contrast, PI0.5 (ft) lacks explicit spatial memory and follows less efficient trajectories once it loses track of the scene state.

B. Scene-Configuration Generalization

To evaluate OOD generalization to scene-configuration changes, we consider three OOD variations applied only at test time: a moved goal location, additional target objects, and target objects placed in an unvisited navigation area. These settings test whether the policy can navigate to a relocated goal, handle additional targets, and search beyond demonstrated routes. Table II compares PI0.5 (ft) and Ours (full) under these OOD variations. Ours (full) achieves higher

Method	Task 21 Moved Goal	Task 21 Added Objects	Task 22 Unvisited Area
PI0.5 (ft) [34]	42.9	50.6	28.0
Ours (full)	**50.8**	**63.0**	**51.0**

TABLE II: **OOD task progress (%).** We evaluate generalization under out-of-distribution task variations. Our method consistently improves task progress over the fine-tuned $\pi_{0.5}$ [3] baseline.

task progress than PI0.5 (ft) across all three variations, suggesting that the 4D latent map improves robustness to OOD configuration changes. Additional details of the experimental setup are provided in Appendix VII.

VI. Conclusion

Long-horizon mobile manipulation requires coherent spatiotemporal reasoning about the robot, the evolving scene, and task progress. We introduced a 4D latent mapping formulation that represents dynamic scenes by embedding both the environment and the robot body. Building on this representation, we developed a VLA policy that uses structured map tokens for action prediction, enabling joint allocentric and egocentric spatial reasoning. Experiments on long-horizon mobile manipulation tasks show that the map-conditioned policy outperforms image-only baselines.

Limitations. Our work has several limitations that motivate future research. First, the current system uses privileged information from simulation (*e.g.*, instance segmentation) to build and update the latent map. In real-world settings, these signals could be estimated using segmentation foundation models such as SAM [25]. Second, real-world experiments are needed to assess transfer beyond simulation. Third, our dynamic environment updates assume per-instance rigid-body motion, limiting the map to scene changes that can be approximated by $\mathrm{SE}(3)$ motion. Extending the mapping formulation to articulated and deformable objects is an important direction for capturing richer real-world dynamics. Finally, the current method requires an offline prior map; a feed-forward encoder [35] could initialize neural point features from streaming observations and reduce this dependency.

REFERENCES

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han, "RT-2: Vision-Language-Action models transfer web knowledge to robotic control," in *Conference on Robot Learning (CoRL)*, 2023.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A Vision-Language-Action flow model for general robot control," in *Robotics: Science and Systems (RSS)*, 2025.

[3] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, "$\pi_{0.5}$: A Vision-Language-Action model with open-world generalization," in *Conference on Robot Learning (CoRL)*, 2025.

[4] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An open-source Vision-Language-Action model," in *Conference on Robot Learning (CoRL)*, 2024.

[5] Gemini Robotics Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. Gonzalez Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. Di Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. Hurwitz Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou, "Gemini robotics: Bringing AI into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[6] J. Bjorck, A. Prasad, K. Bonello, Y.-W. Chao, Y. Hu, D. Fox, M. Shoeybi, Y. Zhu, A. Anandkumar, B. Catanzaro, and F. Ramos, "GR00T N1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[7] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou, "$\pi^*_{0.6}$: a VLA that learns from experience," *arXiv preprint arXiv:2511.14759*, 2025.

[8] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "OK-Robot: What really matters in integrating open-knowledge models for robotics," in *Robotics: Science and Systems (RSS)*, 2024.

[9] J. Chen, H. Liang, L. Du, W. Wang, M. Hu, Y. Mu, W. Wang, J. Dai, P. Luo, W. Shao, and L. Shao, "OWMM-Agent: Open world mobile manipulation with multi-modal agentic data synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Z. Yan, S. Li, Z. Wang, L. Wu, H. Wang, J. Zhu, L. Chen, and J. Liu, "Dynamic open-vocabulary 3D scene graphs for long-term language-guided mobile manipulation," *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[11] P. Liu, Z. Guo, M. Warke, S. Chintala, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "DynaMem: Online dynamic spatio-semantic memory for open world mobile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[12] M. Mohammadi, D. Honerkamp, M. Büchner, M. Cassinelli, T. Welschehold, F. Despinoy, I. Gilitschenski, and A. Valada, "MORE: Mobile manipulation rearrangement through grounded language reasoning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[13] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun, "Navigation world models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[14] A. Sridhar, J. Pan, S. Sharma, and C. Finn, "MemER: Scaling up memory for robot control via experience retrieval," in *International Conference on Learning Representations (ICLR)*, 2026.

[15] M. Torne, K. Pertsch, H. Walke, K. Vedder, S. Nair, B. Ichter, A. Z. Ren, H. Wang, J. Tang, K. Stachowicz, K. Dhabalia, M. Equi, Q. Vuong, J. T. Springenberg, S. Levine, C. Finn, and D. Driess, "MEM: Multi-scale embodied memory for Vision-Language-Action models," *arXiv preprint arXiv:2603.03596*, 2026.

[16] M. Lin, X. Liang, B. Lin, J. Liu, Z. Jiao, K. Li, Y. Sun, W. Liufu, Y. Ma, Y. Liu, S. Zhao, Y. Zhuang, and X. Liang, "EchoVLA: Synergistic declarative memory for VLA-driven mobile manipulation," *arXiv preprint arXiv:2511.18112*, 2025.

[17] R. Steiner, A. Millane, D. Tingdahl, C. Volk, V. Ramasamy, X. Yao, P. Du, S. Pouya, and S. Sheng, "MindMap: Spatial Memory in Deep Feature Maps for 3D Action Policies," in *Conference on Robot Learning (CoRL)*, 2025.

[18] S. Kim, W. Chung, Z. Dai, D. Bhatt, A. Shukla, H. Su, Y. Tian, and N. Atanasov, "Seeing the Bigger Picture: 3D latent mapping for mobile manipulation policy learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[19] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan, "3D-VLA: 3D Vision-Language-Action Generative World Model," in *International Conference on Machine Learning (ICML)*, 2024.

[20] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky, "Neural Point-Based Graphics," in *European Conference on Computer Vision (ECCV)*, 2020.

[21] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: Lidar slam using a point-based implicit neural representation for achieving global map consistency," *IEEE Transactions on Robotics (T-RO)*, 2024.

[22] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski, "DINOv3," *arXiv preprint arXiv:2508.10104*, 2025.

[23] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei, "BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning (CoRL)*, 2024.

[24] C. M. Kim, M. Wu, J. Kerr, K. Goldberg, M. Tancik, and A. Kanazawa, "GARField: Group anything with radiance fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[25] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment anything in images and videos," in *International Conference on Learning Representations (ICLR)*, 2025.

[26] J. Shi and C. Tomasi, "Good features to track," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[27] N. Karaev, I. Makarov, J. Kolesov, N. Ravi, A. Vedaldi, and D. Novotny, "CoTracker3: Simpler and better point tracking by pseudo-labelling real videos," in *European Conference on Computer Vision (ECCV)*, 2024.

[28] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision (ECCV)*, 2016.

[29] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 1992.

[30] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point Transformer," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[31] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations (ICLR)*, 2023.

[32] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[33] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei, "OmniGibson 2.0: A platform for advancing embodied AI research in physically-grounded environments," in *Conference on Robot Learning (CoRL)*, 2024.

[34] I. Larchenko, G. Zarin, and A. Karnatak, "Task adaptation of Vision-Language-Action model: 1st place solution for the 2025 BEHAVIOR challenge," *arXiv preprint arXiv:2512.06951*, 2025.

[35] Y. Tian, H. Cao, S. Kim, and N. Atanasov, "MISO: Multiresolution submap optimization for efficient globally consistent neural implicit reconstruction," in *Robotics: Science and Systems (RSS)*, 2025.

[36] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

Appendix

I Mapping Dataset Generation 9
II Latent Map Details 9
III Contrastive Objectives 9
IV Map Update Details 10
V VLA Policy Details 10
VI Evaluation Benchmark Details 10
VII Scene-Configuration Experiments 11

Appendix I
Mapping Dataset Generation

Environment Mapping Dataset. We construct coordinate–embedding pairs for latent-map optimization by using dense VFM embeddings as supervision targets y (Sec. III-B). Given an RGB image o, depth image Z, and camera pose $(R_{\text{cam}}, t_{\text{cam}})$, the VFM encoder produces per-patch embeddings $G \in \mathcal{Y}^{H\times W}$, one for each of the $H \times W$ non-overlapping patches, where $\mathcal{Y} \subseteq \mathbb{R}^k$ denotes the embedding space. For each patch center v with valid depth $Z[v]$, we back-project v into the world frame using camera intrinsics Π and pose $(R_{\text{cam}}, t_{\text{cam}})$:

$$x(v) = R_{\text{cam}}\,\Pi^{-1}\begin{bmatrix} v \\ 1 \end{bmatrix} Z[v] \;+\; t_{\text{cam}}. \tag{6}$$

Each pair $(x(v), G[v])$ forms a coordinate–embedding sample; aggregating these samples across viewpoints yields the environment-map training dataset.

Robot Mapping Dataset. To train robot neural points, we construct an analogous dataset conditioned on the robot state s (Fig. 8). Given an RGB image o_s, a depth image Z_s, a robot mask M_s, and a camera pose $(R_{\text{cam}}, t_{\text{cam}})$, we extract per-patch embeddings $G_s \in \mathcal{Y}^{H\times W}$ and back-project each valid-depth patch center v that lies within the robot mask M_s:

$$x_s(v) = R_{\text{cam}}\,\Pi^{-1}\begin{bmatrix} v \\ 1 \end{bmatrix} Z_s[v] \;+\; t_{\text{cam}}. \tag{7}$$

Each robot training sample consists of the coordinate–embedding pair $(x_s(v), G_s[v])$ and the corresponding robot state s; aggregating samples across viewpoints and states yields the robot-neural-point training dataset.

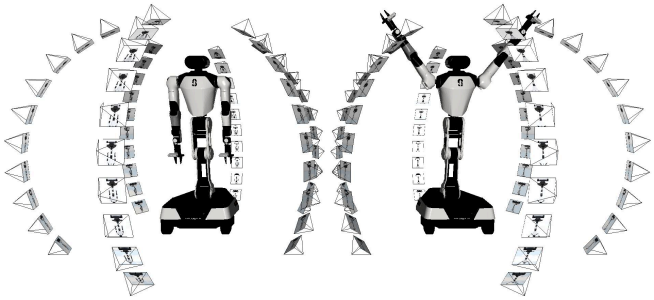

Fig. 8: **Robot mapping dataset generation.** Two curated robot states are rendered from multiple viewpoints to generate training data for robot neural points.

Appendix II
Latent Map Details

We use DINOv3 [22] to generate dense VFM embeddings that serve as latent-map training targets. Given a 480×480 RGB image, DINOv3 extracts 1280-dimensional patch embeddings on a 30×30 grid with patch size 16. We obtain part-level masks using the SAM2 base-plus model [25]. Environment neural points are registered with a voxel size of $0.02\,\text{m}$. Robot neural points are sampled from the robot mesh surface using the same resolution. Each neural point carries a 64-dimensional latent feature. The shared decoder D_θ is implemented as a two-layer MLP with residual blocks. For each spatial query, we retrieve $K{=}6$ nearest neural points and interpolate their latent features using a softmax temperature of 0.05, as in (3).

Appendix III
Contrastive Objectives

Inter-Category Contrastive Objective. The inter-category objective uses an InfoNCE loss [36] to align features from the same semantic category while separating features from different categories:

$$\mathcal{L}_{\text{inter}} = -\log \frac{\exp\big(\text{sim}(f_i, f_j^+)/\sigma_{\text{c}}\big)}{\sum_k \exp\big(\text{sim}(f_i, f_k)/\sigma_{\text{c}}\big)}, \tag{8}$$

where f_j^+ denotes a positive feature from the same category as f_i, the denominator sums over features from all categories, and σ_{c} denotes the contrastive temperature. To implement this objective across scenes, we construct a category-indexed feature bank from the active points. For the inter-category objective, we sample up to 16,384 features per iteration. We use category-balanced importance sampling to prevent categories with many neural points from dominating the contrastive batch. We treat robot features as an additional category in the feature bank. We use $\lambda_{\text{inter}} = 0.02$ and $\sigma_{\text{c}} = 0.1$ for this objective. Fig. 4 visualizes the resulting category-level clustering across scenes.

Intra-Instance Contrastive Objective. The intra-instance objective aligns features within the same SAM2 segment while separating different segments of the same object instance:

$$\mathcal{L}_{\text{intra}} = -\log \frac{\exp\big(\text{sim}(f_i, f_j^+)/\sigma_{\text{c}}\big)}{\sum_{k\in\text{parts}} \exp\big(\text{sim}(f_i, f_k)/\sigma_{\text{c}}\big)}, \tag{9}$$

where the positive pair (f_i, f_j^+) belongs to the same SAM2 segment within an instance, and the denominator sums over features from all segments within that instance. These part-level distinctions capture fine-grained geometric structure that is useful for downstream manipulation. For the intra-instance objective, we sample at most 16,384 valid features per iteration. We again use category-balanced importance sampling to form the sampled feature set. For robot batches, we apply the same part-level contrastive objective using instance segmentation labels rather than SAM2 masks. For the environment intra-instance objective, we use $\lambda_{\text{intra}} = 0.01$

and $\sigma_c = 0.1$. Fig. 5 visualizes the resulting part-level separation within object instances.

Appendix IV
Map Update Details

Offline and Online Updates. Policy training and online execution share the same map-update formulation. Robot points are updated by forward kinematics in both phases: during training, the robot state is read from the offline dataset; during execution, it is read from online state observations. During policy training, we precompute per-instance $\mathrm{SE}(3)$ trajectories from demonstration episodes and store them with the training dataset. At each training step, the data loader applies the precomputed transforms to the environment points, decoupling map updates from policy optimization. During online execution, the environment tracker estimates per-instance $\mathrm{SE}(3)$ transforms from egocentric observations before each policy step. The offline and online pipelines use the same tracking formulation but consume buffers differently. Offline preprocessing advances the recorded buffer by one sampled frame per update, whereas online execution accumulates observations in a rolling buffer and flushes the pending window when constructing a new policy input.

Tracking Details. We exclude background and structural categories (walls, floors, ceilings) from tracking. The tracker processes the head and wrist camera views, maintains a rolling buffer for each view, and updates at every simulation step. For each instance, we initialize sparse tracking points by combining Shi-Tomasi corners [26] with mask samples and track them across the buffer using CoTracker3 [27]. Objects with negligible centroid motion are treated as stationary and excluded from registration. When an object's observed centroid drifts substantially from its stored centroid (*e.g.*, after prolonged occlusion or rapid motion), we bypass FGR and initialize ICP from the centroid offset. We accept this rescue update only if the post-ICP centroid residual is small.

Appendix V
VLA Policy Details

Map Tokenizer. We sample $25{,}000$ neural points from the latent map at each policy query. Before sampling, we filter the map to retain task-relevant points. Each sampled point is represented by normalized 3D coordinates and a 64-dimensional latent feature. We process this point set with a local Point Transformer tokenizer that consists of a shared two-level backbone, branch-specific local Point Transformer layers, and attention pooling. The shared backbone uses channel widths $(128, 256)$, two Point Transformer blocks per level, a stride of 4 at each level, and 16 nearest neighbors for local neighborhoods. Each local branch uses a width of 256 with 16 neighbors and produces one 2048-dimensional map token.

VLA Model and Training. We use the BEHAVIOR-1K OpenPI implementation from Larchenko *et al.* [34] as the base VLA and initialize it with their checkpoint pretrained on 50 tasks. After tokenization, the map tokens are appended to the policy prefix together with visual observations and the proprioceptive state. The attention mask treats these map tokens as bidirectional prefix-context tokens, allowing the action expert to attend to the map during flow-matching denoising. We use an action horizon of 30 and an action dimension of 32. For each task, we fine-tune the map-conditioned policy for $20k$ steps with batch size 16 and 15 flow-matching samples per batch. We use per-time-step action normalization, delta-action targets, and correlated action noise with shrinkage coefficient 0.5. The learning rate follows a cosine schedule with $1k$ warmup steps, a peak learning rate of 2.5×10^{-6}, and a final learning rate of 1.0×10^{-6}.

VLA Inference. During evaluation, we recompute the map-conditioned policy input only at policy-query boundaries and use the same point-sampling and normalization pipeline as in training. The policy generates a 30-step action chunk using 20 Euler denoising steps. The execution wrapper applies chunked control by interpolating the first 26 predicted actions over 20 simulation steps using cubic interpolation. To smooth transitions across policy queries, we reuse the final four actions from the previous prediction as soft inpainting constraints for the next prediction. These constraints are applied only when the denoising time exceeds 0.3.

Task	Target objects	Workspace ($x\times y$)	Max steps
Task 21	2 dice, 2 teddy bears, 2 board games, 1 train set	4.4×3.0 m	38,372
Task 22	2 gym shoes, 2 sandals	5.5×9.0 m	15,384
Task 26	4 baskets, 4 candles, 4 butter cookies, 4 Swiss cheeses, 4 bows	7.6×8.8 m	52,120

TABLE III: **BEHAVIOR-1K evaluation task details.**

Appendix VI
Evaluation Benchmark Details

BEHAVIOR-1K [23] is a benchmark of long-horizon household manipulation tasks in OmniGibson [33], with task goals specified in BDDL. We evaluate three BEHAVIOR-1K mobile manipulation tasks involving multi-object rearrangement; each task uses 200 expert demonstrations and 20 evaluation configurations.

Task Summary. Table III summarizes each task in terms of target objects, task-relevant workspace extent, and maximum rollout length. The workspace extent denotes the x- and y-dimensions of the task-relevant region, and the maximum rollout length is set to twice the average demonstration length for each task.

BDDL Goal Conditions. In Task 21, all target toys must be placed in the same bookcase. In Task 22, each shoe must be placed on the hallstand without touching the floor, and shoes of the same category must be next to each other. In Task 26, each wicker basket must contain one candle, one butter cookie, one Swiss cheese, and one bow.

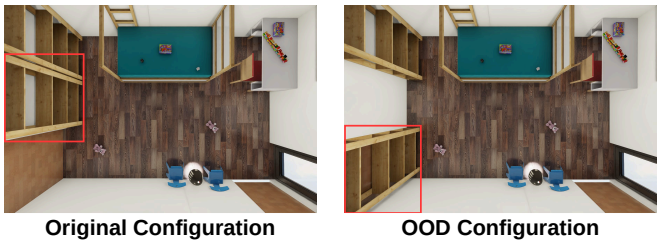

Fig. 9: **Moved-goal OOD setup.** The OOD configuration relocates the bookcase goal from its original position.

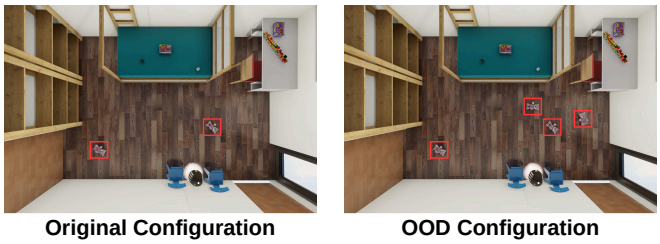

Fig. 10: **Additional-objects OOD setup.** The OOD configuration adds teddy bears to the original scene, yielding nine targets in total.

Appendix VII
Scene-Configuration Experiments

In all three OOD settings, policies are trained only on the original in-distribution scenes. Each OOD variation is applied only during evaluation, testing generalization to changes in navigation regions, goal locations, or object counts. We evaluate each OOD experiment over 10 configurations.

Moved Goal. We evaluate the *Moved Goal* setting on Task 21 (*Collecting Children's Toys*), where the bookcase serves as the placement goal for the collected toys. In this variation, we relocate the bookcase while leaving the target objects and task semantics unchanged. This tests whether the policy uses the updated scene map to navigate to the new goal rather than relying on memorized demonstration locations.

Additional Objects. We evaluate the *Additional Objects* setting on Task 21 (*Collecting Children's Toys*). The nominal task requires the robot to collect target toys and place them on the bookcase. We add extra teddy bears to the evaluation scene, increasing the number of teddy-bear targets from two to four and yielding nine target toys in total, while keeping the instruction and goal unchanged. This tests whether the policy can identify and transport more target objects than it observes in the training demonstrations.

Unvisited Area. We evaluate the *Unvisited Area* setting on Task 22 (*Putting Shoes On Rack*). Of the four target shoes and sandals, one target sandal and one target shoe are placed in a region of the same scene that the robot does not visit during training demonstrations. The goal receptacle remains unchanged, but success requires the robot to search and navigate beyond demonstrated routes. This tests whether the map-conditioned policy can use its broader scene representation to reach previously unvisited regions.

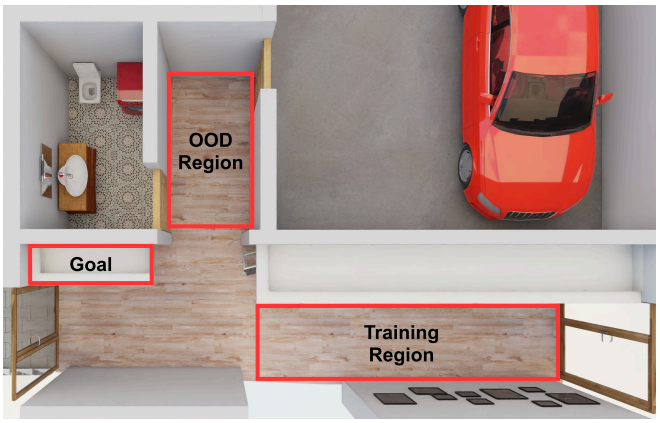

Fig. 11: **Unvisited Area OOD setup.** During evaluation, one sandal and one shoe from the four target objects are placed in a region where no target objects appear in the training demonstrations.