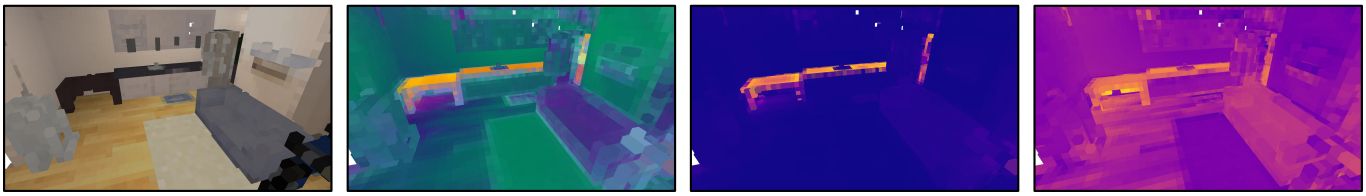
Seeing the Bigger Picture: 3D Latent Mapping for Mobile Manipulation Policy Learning

Sunghwan Kim[1], Woojeh Chung[1], Yulun Tian[1], Zhirui Dai[1], Arth Shukla[1,2], Hao Su[1,2] and Nikolay Atanasov[1]
[1]University of California, San Diego, [2]Hillbot Inc.
{suk063,w5chung,yut034,zhdai,arshukla,haosu,natanasov}@ucsd.edu



(a) ReplicaCAD Scene (b) 3D Latent Map (c) Text Similarity (d) Attention Score

Fig. 1: (a) RGB rendering of a ReplicaCAD scene. (b) 3D latent feature map constructed by our method, visualized with principal-component analysis (PCA). (c) Text similarity scores across the latent map relative to the text embedding of "table". (d) Attention weights on the latent map during policy execution highlight the regions most attended by the policy model.

***Abstract*—This paper investigates whether mobile manipulation policies utilizing a *3D latent map* achieve better spatial and temporal understanding compared to image-based reasoning. We introduce an end-to-end policy learning approach that operates directly on a 3D map of latent features, which (i) extends perception beyond the robot's current field of view and (ii) aggregates observations over time, resolving occlusions and suppressing noise. Our mapping approach incrementally fuses multiview observations into a grid of scene-specific latent features, while a shared pre-trained scene-agnostic decoder enables rapid online adaptation. Our policy design utilizes the feature map by receiving both *global* context obtained by tokenizing the scene-wide latent features and *local* perception information that injects nearby map features into observed visual embeddings. Experiments demonstrate that the map-conditioned policy reasons over the entire scene and successfully completes mobile manipulation tasks in novel layouts where target objects lie outside the robot's field of view.s**

I. Introduction

Recent advances in robot learning have led to remarkable progress in manipulation in semi-structured environments [1]–[3]. State-of-the-art systems harness pre-trained large vision–language models (VLMs), whose rich semantic priors and cross-modal reasoning translate natural language commands directly into low-level actions. The next frontier lies in extending these methods beyond table-top setups to room, building, and even neighborhood scales, to support long-term applications such as warehouse maintenance and last-mile delivery. However, existing learning methods rely on *image-based* designs that directly operates on raw video streams. While effective for short-term action prediction, the image-based approach inherently struggles with consistent 3D understanding and long-horizon reasoning—two critical capabilities for spatially or temporally extended tasks.

In this work, we advocate for an alternative, *3D-based* design that conditions robot policy learning on an explicit 3D representation of the environment. A growing body of recent works explores 3D scene representations for manipulation. Some methods lift 2D foundation-model features into 3D on a per-frame basis [4]–[7]. Another line of methods encode raw point-cloud observations directly with specialized 3D backbones [8]–[10]. While both families preserve metric geometry and enhance local scene understanding, reconstructing the scene from scratch at each timestep compromises temporal consistency and hinders long-horizon reasoning. Complementary efforts fuse multiview observations into feature fields offline [11]–[13]. Although these feature fields improve multiview consistency, they are confined to tabletop-scale setups where the entire workspace remains visible at every timestep and cannot adapt on the fly to novel views.

In this work, we aim to advance the state-of-the-art methods for robot mobile manipulation by conditioning the robot policy on a 3D latent map: a scene representation built incrementally from continuous observations and maintained across tasks. Persistent maps have long benefited navigation [14], [15], yet their potential to enhance manipulation remains underexplored. Such maps offer two key advantages for mobile manipulation: **(i)** *visibility* beyond the current field of view enables global reasoning about object locations and task objectives, and **(ii)** *temporal aggregation* resolves occlusions and suppresses noise from instantaneous observations, thereby enhancing generalization. Fig. 1b illustrates a latent map containing spatially grounded language features, generated by our method. A policy conditioned on this map attends to the entire latent map during task execution, demonstrating an ability to leverage spatially and temporally extended context.

Our mapping approach incrementally fuses multiview observations into a feature grid, capturing extended horizon context. We propose a modular design that decouples the *scene-specific* feature grid from a *scene-agnostic* decoder, pre-trained on

diverse environments. The feature grid represents the scene using compressed, multiview-aggregated latent features, while the decoder is pre-trained to reconstruct target embeddings (*e.g.*, CLIP [16]) from these latent features to support mobile manipulation tasks. During deployment, only the latent features need to be inferred online, while the pre-trained decoder is used directly, enabling rapid adaptation. To harness the main benefits of a 3D map, our map-conditioned policy utilizes **(i)** a *global scene token* that conveys scene-wide context, and **(ii)** *local feature fusion* that enriches observed visual embeddings with nearby map features.

Our contributions are summarized as follows.

- We propose a mapping approach that incrementally builds a 3D map of latent features with a modular design that decouples scene-specific feature optimization from scene-agnostic feature decoding to enable generalization across different environments.
- We design a policy model that augments visual observations with global and local latent map features, increasing the spatial and temporal reasoning context of the model.
- We demonstrate that the map-conditioned policy reasons effectively over the entire scene and completes mobile manipulation tasks under novel layouts where target objects lie outside the robot's field of view.

II. Problem Formulation

The first objective is to construct a dense 3D latent feature map of the robot's workspace. Once the map is available, we aim to design a mobile manipulation policy that treats the latent map as an explicit state variable to execute manipulation tasks specified in natural language.

A. Latent Feature Mapping

Let $\mathcal{X} \subseteq \mathbb{R}^3$ denote the robot's workspace, $\mathcal{F} \subseteq \mathbb{R}^d$ a latent feature space, and $\mathcal{Y} \subseteq \mathbb{R}^k$ a target embedding space (*e.g.*, of language features such as CLIP [16]). We represent a latent feature map as $\mathcal{M}{=}(F_\psi, D_\theta)$, where $F_\psi : \mathcal{X} \to \mathcal{F}$ is an encoder with parameters ψ that lifts workspace points $x \in \mathcal{X}$ to the latent space and $D_\theta : \mathcal{F} \to \mathcal{Y}$ is a decoder with parameters θ that projects a latent feature to the output space $\mathcal{Y}$. The intermediate feature space $\mathcal{F}$ enables the map to capture the geometric and semantic structure of the environment more effectively than a direct $\mathcal{X} \to \mathcal{Y}$ mapping [17], [18].

Problem 1. Given a dataset $\mathcal{D} = \{(x, y)\} \subset \mathcal{X} \times \mathcal{Y}$ of workspace points x with associated target labels y, learn encoder-decoder parameters (ψ, θ) for a latent map by optimizing the target label reconstruction:

$$\min_{\psi,\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\mathcal{L}\big(D_\theta(F_\psi(x)), y\big)\big], \tag{1}$$

where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a distance function on $\mathcal{Y}$.

To instantiate Problem 1 for learning language-grounded maps, we use dense visual features extracted from a VLM as target labels y. Given an RGB image I, a depth image Z, and camera pose (R, t), we compute per-patch embeddings $G \in \mathbb{R}^{k\times(H\times W)}$ by feeding I through the VLM's vision encoder. Here, following the ViT [19] convention, we partition the image into $H \times W$ non-overlapping patches, each producing a k-dimensional feature embedding at its corresponding spatial location. We back-project each patch p with valid depth Z_p into the 3D world frame using the camera intrinsics K and pose (R, t), as shown in Fig. 2:

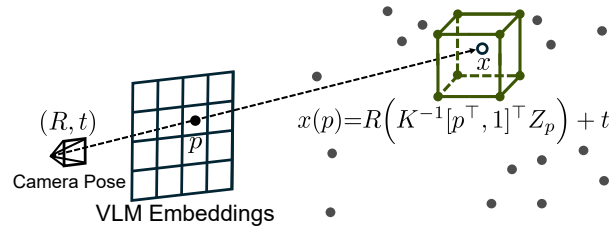


Fig. 2: Visualization of per-patch VLM embeddings back-projected into the 3D world frame using depth Z_p and camera pose (R, t).

$$x(p) = R\Big(K^{-1}[p^\top,\, 1]^\top\, Z_p\Big) + t. \tag{2}$$

Each 3D point $x(p)$ is then paired with its corresponding embedding $y(p) = G_p$, yielding a (x, y) pair. By aggregating these pairs across multiple viewpoints, we construct a training set $\mathcal{D}$. Minimizing (1) ensures that the latent map captures the semantics provided by the VLM and associates them with 3D spatial locations. As shown in Fig. 1c, the learned map localizes semantic concepts queried via text prompts, enabling spatial grounding of language.

B. Map-Conditioned Policy Learning

We consider a mobile manipulator robot performing object-picking tasks. We train a policy with behavior cloning (BC) [20], though other policy learning methods could also be used. The training data consist of expert demonstration episodes with varying target objects and environment configurations. For each episode, the target object label (*e.g.*, "bowl") is encoded by the VLM text encoder [21], producing a text embedding ℓ. At every time step the dataset logs robot configuration s (*e.g.*, from proprioception), action a (*e.g.*, mobile-base and arm-joint velocities), and onboard observation o (*e.g.*, RGB-D images). Given the robot's state and observations, we train the policy to imitate the expert's actions.

Problem 2. Let $\mathcal{T} = \{(o, s, a, \ell)\}$ be a set of expert demonstrations, where each tuple contains the robot observation, state, action, and a language embedding ℓ that specifies the target object in an object-picking mobile manipulation task. Using the learned map $\mathcal{M}$ from Problem 1, we train a policy π_ϕ, parameterized by ϕ, to imitate the expert by minimizing the negative log-likelihood of the demonstrated actions:

$$\min_{\phi} \mathbb{E}_{(o,s,a,\ell)\sim\mathcal{T}}\big[-\log \pi_\phi(a \mid \mathcal{M}, o, s, \ell)\big]. \tag{3}$$

III. Latent Feature Mapping

In this section, we present our latent map design, which is grounded in two principles. (1) *Incremental updates*: a

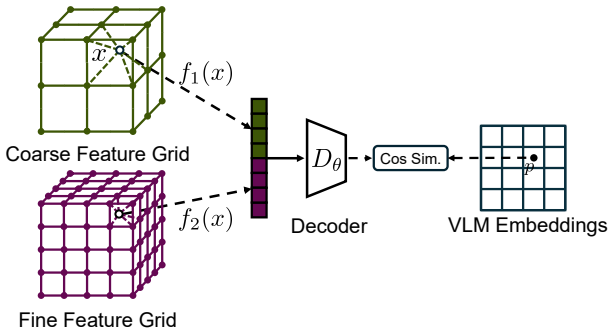


Fig. 3: Instantiation of the latent mapping approach for language grounding. For each 3D point x, we retrieve its coarse-level feature $f_1(x)$ and fine-level feature $f_2(x)$, concatenate them, and train the model to align the resulting vector with the target VLM embedding by maximizing cosine similarity.

3D feature grid continuously integrates new multiview observations, enabling the map to accumulate spatially and temporally extended context and act as a spatial memory. (2) *Modularity*: the encoder parameters ψ which associate scene-specific features with the workspace are separated from the decoder parameters θ, which are trained to reconstruct latent features into target embeddings. After pre-training the decoder on diverse environment configurations, adapting to a new environment requires tuning only the latent features, making our method efficiently generalizable to new environments.

A. Multiresolution Feature Grid

We represent the scene as learnable latent vectors anchored at the vertices of a regular 3D grid. These vectors act as a spatial memory that is updated incrementally as new observations arrive. Let $\mathcal{G} = \{(z_i, f_i)\}_{i=1}^{M}$, where each vertex position $z_i \in \mathcal{X}$ stores a latent vector $f_i \in \mathbb{R}^c$. For a query point $x \in \mathcal{X}$, its feature is retrieved by trilinear interpolation of the eight vertex features of the voxel containing x,

$$f(x) = \sum_{i \in \mathcal{N}(x)} w(x, z_i)\, f_i, \tag{4}$$

where $\mathcal{N}(x)$ indexes the neighboring vertices and $w(\cdot,\cdot)$ provides (trilinear) interpolation weights.

To capture information at multiple scales, we use a hierarchy of L grids $\{\mathcal{G}_l\}_{l=1}^{L}$, ranging from coarse ($l$=1) to fine ($l$=$L$) resolutions, based on the design proposed in [17], [18] (see Fig. 3). Let $f_{l,i} \in \mathbb{R}^c$ denote the latent vector at vertex $z_{l,i}$ of grid $\mathcal{G}_l = \{(z_{l,i}, f_{l,i})\}_{i=1}^{M_l}$, where M_l is the number of vertices at level l. The collection of all latent vectors $\psi = \{\, f_{l,i} \mid l = 1{:}L,\ i = 1{:}M_l\}$ constitutes the scene-specific map parameters. The interpolated feature at level l is $f_l(x)$. Concatenating the level-wise features yields the final feature given to the decoder:

$$F_\psi(x) \;=\; \bigoplus_{l=1}^{L} f_l(x) \;\in\; \mathcal{F} \subseteq \mathbb{R}^d, \quad d = L\,c. \tag{5}$$

B. Latent Feature Decoder

The decoder D_θ maps a latent feature $F_\psi(x)$ to the target space $\mathcal{Y}$. The predicted feature of any query point $x \in \mathcal{X}$ is

$$\hat{y}(x) = D_\theta\big(F_\psi(x)\big) \in \mathcal{Y} \subseteq \mathbb{R}^k. \tag{6}$$

We implement D_θ as a multilayer perceptron (MLP). The decoder is pre-trained on scenes from diverse environment configurations to learn a general mapping from the latent space $\mathcal{F}$ to the target space $\mathcal{Y}$. Intuitively, $F_\psi(x)$ serves as a multiview-aggregated, compressed representation of target feature embeddings, while D_θ is trained to reconstruct them back into the target space.

During deployment, we freeze the decoder; its pre-training procedure is detailed in Sec. V-A. Consequently, adaptation to a new environment only updates the grid parameters ψ, greatly accelerating optimization. The parameters ψ (and θ, if not pre-trained) are optimized by minimizing a loss $\mathcal{L}$ in (1). In practice, we align the predicted feature $\hat{y}(x)$ with the reference y using cosine similarity loss, which empirically outperforms alternatives such as L_2 loss. Given a dataset $\mathcal{D}$ of point–feature pairs (x, y), the objective is

$$\min_{\psi,\theta} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \big[1 - \cos\big(\hat{y}(x), y\big)\big]. \tag{7}$$

Fig. 3 summarizes the overall mapping approach, instantiated for language-grounding as described in Sec. II-A.

IV. Map-Conditioned Policy

This section details how our latent map $\mathcal{M}$ may be used to condition a BC policy π_ϕ. To harness the key benefits of the latent map, we enrich our policy network with both *global* context information obtained by tokenizing the scene-wide latent map, and *locally* fused perception that injects nearby map features into observed per-patch embeddings.

A. Global Scene Token

A latent map enables a robot to reason about task-relevant objects and goal positions even when they lie outside the robot's current field of view. We attend to the entire map and distill its features into a single global scene token using a 3D encoder adapted from Point Transformer [22]. To improve 3D understanding, we integrate 3D Rotary Positional Encoding (RoPE) [4], [23] into each attention layer, thereby conditioning attention weights on relative spatial offsets (see Sec. B). The encoder operates on the vertices of the feature grid. Before encoding, we re-weight each vertex feature by its cosine similarity to the target object's text embedding $\ell \in \mathcal{Y}$ (*e.g.*, "apple"), guiding the model to focus on task-relevant regions:

$$y_0(z) = S\big(\hat{y}(z),\, \ell\big), \quad z \in \mathcal{Z}_0,\ \hat{y}(z) \in \mathcal{Y}, \tag{8}$$

$$S(v, \ell) = \big(1 + \cos(v, \ell)\big)\, v, \quad v \in \mathcal{Y}. \tag{9}$$

Here, $y_0(\cdot)$ is the task-aware feature fed to the encoder, and $\mathcal{Z}_0 = \{z_{1,i}\}$ is the set of vertices of the coarse grid.

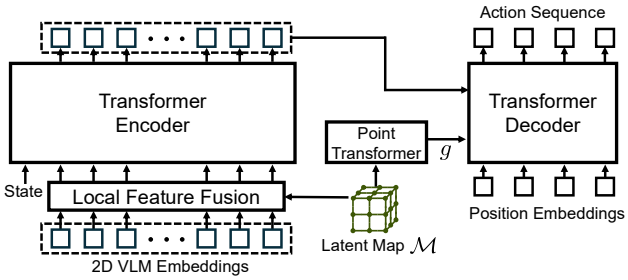


Fig. 4: Overview of map-conditioned policy architecture. Our model extends ACT [24] by conditioning on both global and local features from the latent map.

The encoder proceeds through N hierarchical stages. At stage $n \in \{1, \ldots, N\}$ it operates on the vertices $\mathcal{Z}_{n-1}$ with their corresponding features $y_{n-1}(\cdot)$. Each stage comprises

$$\begin{aligned}
&\text{Centroid sampling:}\ \ \mathcal{C}_n = \mathrm{FPS}\big(\mathcal{Z}_{n-1}, M_n\big),\\
&\text{Ball query:}\ \ \mathcal{N}_{n,j} = \mathcal{B}_{r_n}\big(c_{n,j}\big),\ c_{n,j} \in \mathcal{C}_n,\\
&\text{Attention:}\ \ \{h_{n,j}(b)\}_{b\in\mathcal{N}_{n,j}} = \mathrm{MHA}\big(\{y_{n-1}(b)\}_{b\in\mathcal{N}_{n,j}}\big),\\
&\text{Max-pooling:}\ \ y_n(c_{n,j}) = \max_{b\in\mathcal{N}_{n,j}} h_{n,j}(b). \qquad (10)
\end{aligned}$$

where $\mathrm{FPS}(\cdot)$ selects M_n farthest-point centroids, $\mathcal{B}_{r_n}(\cdot)$ is a ball query detailed in Sec. A, and $\mathrm{MHA}(\cdot)$ represents multi-head self-attention equipped with 3D RoPE. At each stage, the new vertex set is $\mathcal{Z}_n{=}\mathcal{C}_n$ with features $y_n(\cdot)$. After the final stage, we apply max-pooling to produce the scene token

$$g = \max_{j\in\mathcal{C}_N} y_N\big(c_{N,j}\big), \qquad (11)$$

which provides scene-wide, task-aware context to the policy.

B. Local Feature Fusion

Instantaneous observations can be noisy or occluded, whereas a temporally aggregated map supplies more robust long-term context. We therefore introduce a local feature-fusion module that enriches instantaneous per-patch embeddings with spatially aligned map features. This fusion operates on the vertices of the fine-level grid $\mathcal{G}_2$. For each patch p with VLM embedding G_p, we back-project to its 3D point $x(p)$ using (2) and retrieve its neighboring vertices $\mathcal{N}_p{=}\mathcal{B}_{r_2}\big(x(p)\big) \cap \mathcal{G}_2$. We then construct a gated token set

$$\mathcal{T}_p \;=\; \big(\underbrace{\mathcal{S}(G_p,\ell)}_{u_{p,0}},\ \underbrace{\mathcal{S}\big(\hat{y}(z_1),\ell\big)}_{u_{p,1}},\ \ldots,\ \underbrace{\mathcal{S}\big(\hat{y}(z_{|\mathcal{N}_p|}),\ell\big)}_{u_{p,|\mathcal{N}_p|}}\big), \qquad (12)$$

where each $z_i \in \mathcal{N}_p$. A self-attention layer equipped with 3D RoPE updates the token set:

$$\big(h_{p,0},\ldots,h_{p,|\mathcal{N}_p|}\big) = \mathrm{MHA}\Big(\{u\}_{u\in T_p}\Big), \quad \tilde{G}_p = h_{p,0}. \qquad (13)$$

The first output token $h_{p,0} \in \mathbb{R}^k$ corresponds to the original patch query modulated by local map context. We discard the remaining tokens $h_{p,m}$ $(m \geq 1)$ to keep the downstream token budget unchanged. Collecting $\tilde{G}_p$ over all patches yields the set of map-enriched per-patch embeddings $\tilde{G}$.

C. Policy Architecture

We implement our map-conditioned BC policy by adapting the Action Chunking Transformer (ACT) [24]. At each control step, ACT attends to the visual tokens $\tilde{G}$, the global scene token g, and the proprioceptive state s. We project s to the visual token dimension and append it to $\tilde{G}$. Spatial relations among the per-patch embeddings are encoded with 3D RoPE. A Transformer encoder processes the resulting tokens; its output, concatenated with g, provides keys and values for the ACT decoder. Fixed, learnable positional embeddings serve as decoder queries, whose MLP head then converts the resulting latent chunks into continuous control commands. Fig. 4 summarizes the full architecture.

V. Evaluation

This section evaluates whether our map-conditioned policy improves mobile manipulation performance. We show that it outperforms an image-based approach, particularly when the robot begins far from its target objects. After describing the implementation details and experiment setup, we compare our method against baselines using direct visual representations. All experiments are conducted with the ManiSkill [25] simulator using ReplicaCAD scenes from Habitat [26]. Our codebase extends ManiSkill-HAB [27].

A. Implementation Details

This subsection outlines our mapping pipeline; implementation details of policy architecture are provided in Sec. C. We process one ReplicaCAD [26] scene at a time. As each scene comprises multiple environment configurations with different object layouts, we assign a separate feature grid to every configuration. The decoder is jointly pre-trained across all configurations within a scene and then frozen. For efficiency, each configuration's feature grid is pre-trained offline and loaded on demand during policy learning. Maps are built directly from the robot's RGB-D observations, with dynamic regions masked out using depth. We use EVA-02-Large [21] as the VLM backbone. The map is represented by a two-level grid ($L{=}2$): a coarse-level with 0.4 m voxels ($l{=}1$) and a fine-level with 0.2 m voxels ($l{=}2$). Note that each map captures only the initial object arrangement.

B. Experiment Setup

Benchmark. We adapt the Pick subtasks of two home-rearrangement benchmarks, SetTable and PrepareGroceries [26], [27]. Unlike the original benchmarks, our training is conducted solely on demonstrations collected in a single scene (`sc1-13`). These demonstrations are generated using the Reinforcement Learning (RL) policy released by [27]. Subsequently, a BC policy is trained as described in Sec. IV.

Evaluation Metric. Performance is evaluated using Success Rate (SR) and Episode Reward (ER). ER is the time-accumulated dense reward, which is dominated by five main terms: reach shaping, a grasp bonus, post-grasp shaping, a success bonus, and a collision penalty. We evaluate performance on both the training scene (ID, `sc1-13`) and a novel,

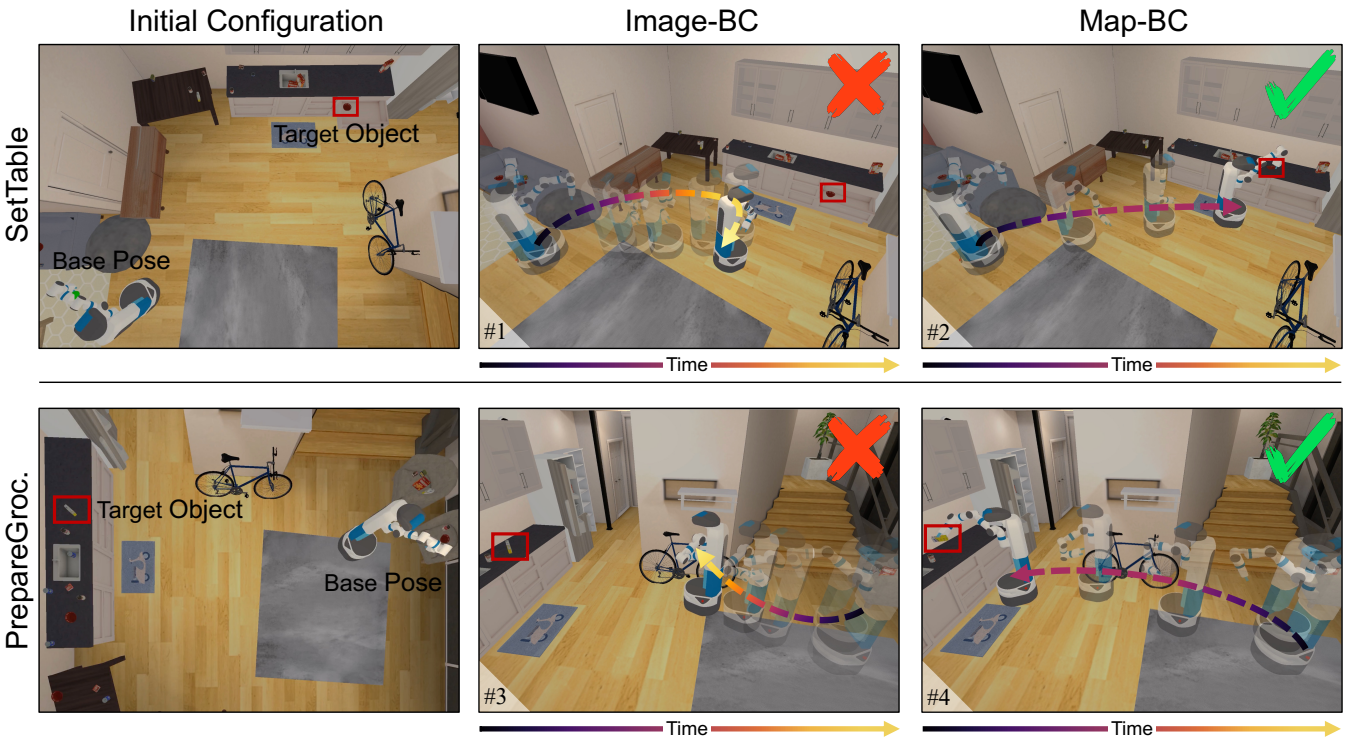


Fig. 5: Qualitative comparison on home-rearrangement tasks under out-of-distribution conditions. At the start of each episode, the robot is placed at a distant base pose with the target object completely outside of the robot's field of view. Image-BC (#1 and #3) fails to localize the object in these settings, producing inefficient trajectories that never reach the target. In contrast, Map-BC (#2 and #4) successfully navigates to and grasps the object, completing the task with direct efficient trajectories.

TABLE I: Performance of BC policies using different visual representations for the Pick subtasks of two home-rearrangement benchmarks. For each task, we report success rate (SR↑) and episode reward (ER↑) on both the training scene (ID) and a novel scene (OOD), averaged over three runs. Best and second-best results are highlighted in **bold** and underlined, respectively.

	SetTable-Pick				PrepareGroceries-Pick			
Method	SR (ID)	SR (OOD)	ER (ID)	ER (OOD)	SR (ID)	SR (OOD)	ER (ID)	ER (OOD)
Image-BC	0.38	0.31	0.51	0.47	0.21	0.17	0.4	0.38
Uplifted [5]	0.35	0.33	0.47	0.45	0.23	0.17	0.42	0.38
Point Cloud [8]	0.39	0.34	0.47	0.46	0.19	0.16	0.39	0.38
Map-BC (ours)	**0.54**	**0.44**	**0.59**	**0.53**	**0.28**	**0.25**	**0.47**	**0.42**

unseen scene (OOD, `sc1-10`). For the OOD scene, we assume access to a pre-generated latent map but no expert demonstrations, thereby assessing the generalization capacity. During evaluation, we sample 100 environment configurations per scene and report results averaged over three runs.

Baselines. We compare our map-conditioned policy (Map-BC) with three baseline policies that rely on alternative visual representations:

- Image-BC: Use raw 2D VLM embeddings directly.
- Uplifted: Use 2D VLM embeddings lifted to transient 3D tokens, following [5].
- Point Cloud: Process point-cloud observations processed using a 3D encoder, following [8].

To ensure a fair comparison, all baseline methods share the same policy architecture as our proposed approach. Additionally, 3D RoPE is applied to Uplifted and Point Cloud.

C. Results

Table I shows that Map-BC achieves the highest SR and ER in both ID and OOD scenes. The stronger scene understanding capability of our method allows the robot to localize and reach targets more efficiently, *e.g.*, a 10% ER improvement over Image-BC in the OOD setting. We attribute these gains to the latent map's capacity for global reasoning.

In Fig. 5 we test whether the map-conditioned policy can perform global reasoning using the latent map. To make the target difficult to localize, we initialize the robot at a pose deliberately perturbed in both translation and rotation, so that the target lies outside its field of view for an extended period. We evaluate five translational offsets paired with four rotational offsets. Map-BC outperforms Image-BC in every case; for clarity, the figure visualizes the two most illustrative runs. The robot's pose is plotted every 0.8 s until termination. The Image-BC baseline produces erratic, cluttered trajectories

and fails to reach the target, whereas Map-BC drives directly to the goal and completes the task. These qualitative results indicate that the latent map enables the policy to develop a global understanding of the scene.

VI. Conclusion and Future Work

While 3D environment maps have long been core components of navigation, they have been largely overlooked in learning manipulation policies. In this paper, we presented a 3D latent map formulation that offers key advantages for manipulation: (i) perception beyond the robot's current field of view and (ii) observation aggregation over long horizons. Building on these advantages, we proposed an end-to-end approach that couples a 3D latent map with a mobile manipulation policy, providing the robot with rich spatial and temporal context. Experiments on object-picking tasks demonstrate that the map-conditioned policy reasons over the entire scene and successfully completes tasks in novel layouts.

We identify three directions for future work. First, our current implementation uses pre-generated maps; extending it to online mapping would enable autonomous operation. Second, while our work considered object-picking tasks, it can be extended to more general long-horizon, multi-stage tasks to broaden its applicability. Finally, we plan to transfer the method from simulation to the real world, validating its robustness and effectiveness fro mobile manipulation in diverse environments.

Acknowledgments

We gratefully acknowledge support from NSF 2402689 (ExpandAI) and the Technology Innovation Program (20018112, Development of autonomous manipulation and gripping technology using imitation learning based on visual and tactile sensing) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea.

References

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "A vision-languageaction flow model for general robot control," *RSS*, 2025. 1

[2] 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," *arXiv preprint arXiv:2406.09246*, 2024. 1

[3] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang *et al.*, "PaLM-E: An embodied multimodal language model," *ICML*, 2023. 1

[4] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3D: 3D feature field transformers for multi-task robotic manipulation," in *CoRL*, 2023. 1, 3

[5] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3D Diffuser Actor: Policy diffusion with 3D scene representations," *CoRL*, 2024. 1, 5

[6] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang, "GNFactor: Multi-task real robot learning with generalizable neural feature fields," in *CoRL*, 2023. 1

[7] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li, "GenDP: 3D semantic fields for category-level generalizable diffusion policy," in *CoRL*, 2024. 1

[8] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D Diffusion Policy: Generalizable visuomotor policy learning via simple 3D representations," 2024. 1, 5

[9] Y. Ze, Z. Chen, W. Wang, T. Chen, X. He, Y. Yuan, X. B. Peng, and J. Wu, "Generalizable humanoid manipulation with improved 3D diffusion policies," *arXiv preprint arXiv:2410.10803*, 2024. 1

[10] S. Peri, I. Lee, C. Kim, L. Fuxin, T. Hermans, and S. Lee, "Point cloud models improve visual robustness in robotic learners," *ICRA*, 2024. 1

[11] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg, "Language embedded radiance fields for zero-shot task-oriented grasping," in *CoRL*, 2023. 1

[12] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola, "Distilled feature fields enable few-shot language-guided manipulation," *CoRL*, 2023. 1

[13] Y. Wang, M. Zhang, Z. Li, K. R. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li, "D^3 Fields: Dynamic 3D descriptor fields for zero-shot generalizable robotic manipulation," in *CoRL*, 2024. 1

[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *T-RO*, 2016. 1

[15] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *RA Magazine*, 2006. 1

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021. 2

[17] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *TOG*, 2022. 2, 3

[18] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3d shapes," in *CVPR*, 2021. 2, 3

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021. 2

[20] M. Bain and C. Sammut, "A Framework for Behavioural Cloning," in *Machine Intelligence 15: Intelligent Agents*, 1999. 2

[21] Y. Fang, Q. Sun, X. Wang, T. Huang, X. Wang, and Y. Cao, "EVA-02: A visual representation for neon genesis," *Image and Vision Computing*, 2024. 2, 4

[22] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point Transformer," in *ICCV*, 2021. 3

[23] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomputing*, 2024. 3

[24] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *RSS*, 2023. 4

[25] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao *et al.*, "ManiSkill2: A unified benchmark for generalizable manipulation skills," *ICLR*, 2023. 4

[26] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *NeurIPS*, 2021. 4

[27] A. Shukla, S. Tao, and H. Su, "ManiSkill-HAB: A benchmark for low-level manipulation in home rearrangement tasks," *ICLR*, 2025. 4

APPENDIX A
BALL QUERY

Given a query point $x \in \mathcal{X}$ and a radius $r > 0$, the ball-query operator returns

$$\mathcal{B}_r(x) = \{z \in \mathcal{X} \mid \|x - z\|_2 \le r\}, \tag{14}$$

where $\mathcal{X}$ is the set of map vertices. In practice, the K closest elements of $\mathcal{B}_r(x)$ are retained; if $|\mathcal{B}_r(x)| < K$, the set is padded with x.

APPENDIX B
3D ROTARY POSITIONAL ENCODING.

For a token located at $w = (w_x, w_y, w_z) \in \mathbb{R}^3$ and an embedding vector $f \in \mathbb{R}^d$ with $6 \mid d$, first reshape f into $\frac{d}{6}$ six-dimensional blocks

$$f = \left[f^{(1)}, f^{(2)}, \ldots, f^{(d/6)}\right], \quad \theta_k = \beta^{-\,k/(d/6)}, \tag{15}$$

where each $f^{(k)} \in \mathbb{R}^6$ and β (typically 10^4) sets the geometric progression of inverse wavelengths. Next, rotate every block about the three Cartesian axes,

$$M_k(w) = R\big(\theta_k w_x\big) \;\oplus\; R\big(\theta_k w_y\big) \;\oplus\; R\big(\theta_k w_z\big) \in \mathbb{R}^{6\times 6}, \tag{16}$$

where $R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$ is the planar rotation matrix and $\oplus$ denotes block-diagonal concatenation. Stack the block rotations once and apply them to f:

$$M(w) = \mathrm{diag}\big(M_1(w), \ldots, M_{d/6}(w)\big). \tag{17}$$

We define $\mathrm{RoPE}(w, f) = M(w)\, f$. For two tokens i, j at positions w_i, w_j with embeddings $f_i, f_j \in \mathbb{R}^d$,

$$\mathrm{RoPE}(w_i, f_i)^\top \mathrm{RoPE}(w_j, f_j) = f_i^\top M(w_j - w_i)\, f_j, \tag{18}$$

so the dot-product attention depends only on the relative displacement $w_j - w_i$.

APPENDIX C
MORE IMPLEMENTATION DETAILS

The global scene encoder is a four-stage hierarchical transformer ($N = 4$). At stage n we downsample the point set to $M_n = 0.25 M_{n-1}$ centroids, and we use ball query radii of $(1, 2, 4, 100)$ for the successive stages. For local feature fusion, the ball query radius is fixed to the voxel size of the fine-level grid, *i.e.* 0.2 m. In the policy network, the Transformer encoder contains four layers and the decoder consists of six layers, and we roll out action chunks over a time horizon of 16 steps.