mindmap: Spatial Memory in Deep Feature Maps for 3D Action Policies

Remo Steiner[*], **Alex Millane**[*], **Clemens Volk**[*], **David Tingdahl**[*], **Vikram Ramasamy**[*], **Xinjie Yao**[*], **Peter Du**, **Soha Pouya**, **Shiwei Sheng**
NVIDIA, Zurich, Switzerland. Santa Clara, California.
{remos,amillane,cvolk,dtingdahl,vramasamy,xyao,peterd,spouya,shiweis}@nvidia.com

Abstract: End-to-end learning of robot control policies, structured as neural networks, has emerged as a promising approach to robotic manipulation. To complete many common tasks, relevant objects are required to pass in and out of a robot's field of view. In these settings, spatial memory - the ability to remember the spatial composition of the scene - is an important competency. However, building such mechanisms into robot learning systems remains an open research problem. We introduce *mindmap* (Spatial **M**emory **in D**eep Feature **M**aps for 3D **A**ction **P**olicies), a 3D diffusion policy that generates robot trajectories based on a semantic 3D reconstruction of the environment. We show in simulation experiments that our approach is effective at solving tasks where state-of-the-art approaches without memory mechanisms struggle. We release our reconstruction system[1], training code[2], and evaluation tasks[2] to spur research in this direction.

Keywords: Manipulation policy, Imitation learning, 3D reconstruction, Diffusion policies

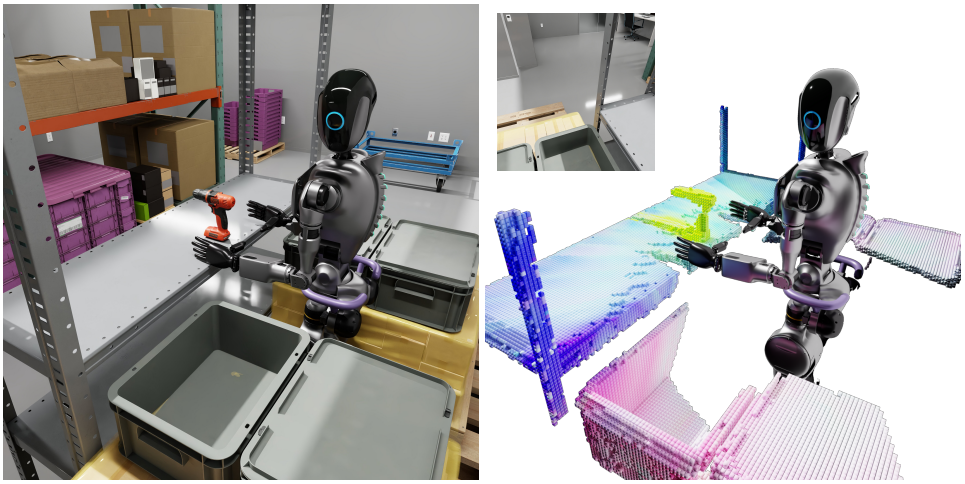

Figure 1: **Spatial Memory Task:** A humanoid in a simulated industrial space (left) and within a metric-semantic reconstruction built by *mindmap* (right) (colored by Principal Component Analysis). The robot's first-person view is shown inset. The task requires the robot to transfer the hand drill from the shelf to the open box. The drill and box positions must be discovered by the policy, and both objects cannot be captured in a single view. Therefore, successful task completion requires the policy to remember the spatial layout of the scene. By leveraging the reconstruction, *mindmap* generates trajectories that depend on parts of the scene that are outside the robot's current Field of View.

[1] github.com/nvidia-isaac/nvblox
[2] github.com/NVlabs/nvblox_mindmap

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

[*] Equal Contribution.

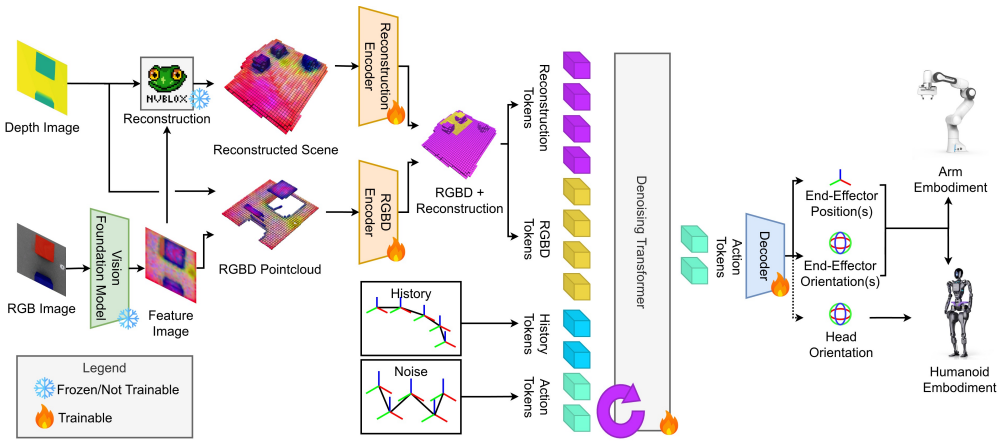

Figure 2: **Overview of *mindmap***. *mindmap* is a Denoising Diffusion Probabilistic Model that samples robot trajectories conditioned on sensor observations and a reconstruction of the environment. Images are first passed through a Vision Foundation Model and then back-projected, using the depth image, to a pointcloud (as in 3D Diffuser Actor [1]). In parallel, a reconstruction of the scene is built that accumulates metric-semantic information from past observations. The two 3D data sources, the instantaneous visual observation and the reconstruction, are passed to a transformer that iteratively denoises robot trajectories.

1 Introduction

Designing generalist robot manipulation policies remains a holy grail of robotics. Such policies would perform manipulation tasks with a high level of competence and be instructed to do so in natural language. Recent advances in deep learning, vision, and natural language processing have, for the first time, brought this goal within reach; however, significant challenges remain.

Existing approaches to developing learned manipulation policies generally aim to learn a mapping from sensor observations to robot control signals [2, 3, 4, 5]. These models typically employ transformer-based architectures to process image and proprioceptive inputs to generate control signals. Such methods have shown an impressive ability to complete language-guided manipulation tasks. One limitation of several leading approaches, however, is that the generation of output signals is conditioned on *current* visual observations only. Such approaches lack spatial memory - the ability to remember the spatial and semantic composition of the scene (see [6] for a taxonomy of robot memory). This leads to surprising limitations to their capabilities. Although some methods incorporate temporal information by maintaining a temporal window of past images, these approaches have drawbacks of their own (see Section 2).

In this work, we introduce *mindmap*, an approach that combines a diffusion policy with a metric-semantic 3D reconstruction of the scene. *mindmap* generates trajectories of 3D end-effector poses in the reconstructed space. This approach allows the policy to generate actions that depend on parts of the scene that are outside of the camera's current Field of View (FOV). Our experiments show that, on tasks requiring spatial memory, *mindmap* is effective in completing tasks on which several current approaches struggle.

Contributions: In this paper, we contribute tools for extending 3D manipulation policies with spatial memory. In particular, we release metric-semantic mapping[3] in *nvblox* [7], our GPU-accelerated reconstruction library[1], in addition to our training code[2], and simulation environments[2] for testing spatial memory. We demonstrate the efficacy of these tools by extending a state-of-the-art 3D diffusion policy [1]. We show that by making changes to the architecture and training, the policy's performance, on challenging tasks that require spatial memory, is significantly improved.

[3] `nvidia-isaac.github.io/nvblox/pages/torch_examples_deep_features`

2 Related Work

Learning robot control policies that map observations directly to robot actions has received considerable recent attention. Following the success of deep learning in other fields, structuring these policies as neural networks has emerged as a promising approach for building generally intelligent machines.

Vision-Language-Action Models: Recent robotics research has attempted to replicate the success of large-scale task-agnostic pre-training in other fields, such as language understanding. RT-1 [8] trained a transformer-based model to produce discrete action tokens on a dataset of 130k demonstrations. To improve generalization and reasoning abilities, several approaches have sought to incorporate Vision-Language Models (VLMs) into robotic models, the combination termed Vision-Language-Action (VLA) models. RT-2 [5] and OpenVLA [4] fine-tune VLMs with robot data, resulting in state-of-the-art zero-shot performance. These models faced limits in their dexterity due to action-space discretization and execution frequency. The π_0 [3] model addressed these limitations, using a diffusion-based action head [9] to represent continuous distributions over action-space, and to produce high-frequency output. GR00T N1 [2] suggests a flow-matching-based VLA trained on varied data sources. Many recent works have sought to improve VLA models through improved action tokenization [10], action-chunking [11, 12], and multi-step instruction following [13], among others.

3D Manipulation Models: In parallel, efforts have been made to train models that utilize 3D sensor data. Perceiver-Actor [14] voxelizes an RGB-D pointcloud and uses a transformer to produce language-conditioned goals. RVT [15] represents the 3D scene through several virtual views, leading to dramatically improved training times. 3D Diffuser Actor [1] represents the scene as a set of featurized 3D points, and processes them using 3D relative attention to produce continuous actions. At the time of writing, policies consuming 3D data have not typically undergone large-scale pre-training. FP3 [16] represents an early attempt to scale up a 3D policy, using the DRIOD [17] dataset.

Reconstruction for Manipulation: Several works have investigated the use of reconstructions in manipulation policies. LERF-TOGO [18] and SplatMover [19] build metric-semantic maps upon which grasp points are predicted, using NERFs and Gaussian splats respectively. In contrast, *mindmap* follows an end-to-end approach, diffusing robot trajectories directly from a reconstruction, without intermediate prediction of grasps. GNFactor [20] uses several external cameras to build a 3D voxel grid of Vision Foundation Model (VFM) features, which are then processed by a transformer to produce voxelized actions. The reconstruction, however, is built from views of the scene at a single timestep. In contrast, our results are generated using a single ego-centric camera that accumulates prior views of the scene to provide past information to the network.

Memory: One limitation of many VLAs and 3D models is that they produce actions based on the *current* observation. As we shall show, this is a significant limitation, even on seemingly trivial tasks. A recent work SAM2ACT [21], addresses the issue of spatial memory in manipulation policies. The authors propose adding a memory bank to RVT2 [22], feeding back prior observations into the policy. The authors demonstrate state-of-the-art performance on tasks requiring spatial memory. However, as the authors note, the approach has several shortcomings. SAM2ACT has a fixed-length memory that requires per-task tuning. The model's recurrent nature requires a specialized training procedure. In contrast, the approach proposed in *mindmap* has no explicit temporal limits. Past information is aggregated spatially, rather than stored in a temporal buffer, and so the computational requirements remain bounded given a bounded volume of space. Furthermore, the approach is not recurrent, and so can be plugged directly into a standard diffusion policy training pipeline.

3 Problem Statement

Given a sequence of observations $\mathcal{O} = \{\mathbf{o}_i\}_{i=0}^{t}$ we aim to find a policy π that outputs a robot action $\mathbf{a}_t$ such that $\mathbf{a}_t = \pi(\mathbf{o}_0, \mathbf{o}_1, ..., \mathbf{o}_t)$. Our observations $\mathbf{o}_i$ take the form of $\mathbf{o}_i = \{\mathcal{I}_i^j, \mathcal{D}_i^j, \mathcal{S}_i\}_{j=0}^{N}$,

for N cameras, where $\mathcal{I}_i^j$ are RGB images, $\mathcal{D}_i^j$ are corresponding posed depth images, and $\mathcal{S}_i$ is the robot state $\mathcal{S}_i = \{\mathbf{p}_i^k, \mathbf{q}_i^k, c_i^k, \gamma_i\}_{k=0}^M$, for M end-effectors. We consider several robot embodiments, but in general, the robot state $\mathcal{S}_i$ is a composition of the 3D positions $\mathbf{p}_i^k \in \mathbb{R}^3$, rotations $\mathbf{q}_i^k \in$ SO(3), the closedness $c_i^k \in \{0, 1\}$ of one or more robot end-effectors, and for humanoid embodiments, the head yaw $\gamma_i \in (-\pi, \pi]$. Our action $\mathbf{a}_i$ lives in the same space as our state $\mathcal{S}_i$, i.e. we command end-effector poses, closedness, and head yaw. Our policy π is a deep neural network which we learn from human demonstrations consisting of observation-action pairs $\mathcal{T} = \{(\mathbf{a}_0, \mathbf{o}_0), (\mathbf{a}_1, \mathbf{o}_1), ..., (\mathbf{a}_T, \mathbf{o}_T)\}$. We build a reconstruction $\mathcal{R}_t$ by accumulating past visual observations $\mathcal{R}_t(\mathcal{I}_0, ..., \mathcal{I}_t, \mathcal{D}_0, ..., \mathcal{D}_t)$. Our policy depends on the current observations directly, a finite sequence of K past states, and on past visual observations through the reconstruction $\mathbf{a}_t = \pi(\mathcal{I}_t^j, \mathcal{D}_t^j, \mathcal{S}_{t-K}, ..., \mathcal{S}_t, \mathcal{R}_t)$

4 Method

In this section we describe our approach, firstly describing our extensions to 3D Diffuser Actor [1] (Section 4.1), and then explaining how we build reconstructions (Section 4.2). See Fig. 2 for an overview.

4.1 Network Architecture

Our approach follows recent work [1, 9, 3] and structures our policy as a denoising transformer that generates robot actions based on observations of the scene. In particular, we extend 3D Diffuser Actor [1], which iteratively denoises an end-effector trajectory, conditioned on posed RGB-D images. In the following, we highlight the key differences between *mindmap* and 3D Diffuser Actor.

Reconstruction tokens: *Mindmap*'s diffusion transformer takes as input RGB-D images and a featurized reconstruction, in the form of 3D vertices extracted from a reconstructed mesh (see Section 4.2). This allows the network to attend to both the current RGB-D observation and the reconstruction, which aggregates past observations. We found that this approach led to better results than providing the reconstruction alone (see Section 6). The reconstruction is continuously updated as new images arrive.

The featurized RGB-D image and the reconstruction are passed through separate encoders to project them from VFM feature dimension to the token embedding dimension (see Fig. 2). Reconstruction and RGB-D tokens are then concatenated and passed through cross and self-attention layers, as in 3D Diffuser Actor. We found that the use of separate encoders led to higher performance than passing both sets of points through a joint encoder. This makes intuitive sense: it allows attention mechanisms to differentiate tokens originating from instantaneous observations and those coming from the reconstruction.

VFM Features: Diffuser Actor uses a pre-trained CLIP ResNet50 image encoder [23] combined with a trainable Feature Pyramid Network (FPN) [24] for feature extraction. The reconstruction process in *mindmap* is non-differentiable and as a result gradients are unable to flow back to the image encoder. We therefore replace CLIP+FPN with a frozen pre-trained VFM, AM-RADIO [25].

Bimanual embodiments: We extend 3D Diffuser Actor, which was designed to control a single robotic arm, for bimanual manipulation tasks using a humanoid robot. We therefore modify the model from predicting single end-effector poses and closedness to (optionally) predict bimanual end-effector poses and closedness. We concatenate the past states of multiple end-effectors to form the proprioceptive history, and we modify the prediction heads in the network to predict the next states for multiple end-effectors (as suggested in [26]).

Controlling head orientation: We additionally allow the policy to control the head orientation of humanoid robots. This allows *mindmap* to complete tasks in which not all task-relevant objects can be held in a single view of the scene. In such situations, the robot must gather information from several views in order to complete the task. To achieve this, we add an additional decoder

for the head orientation. In training, the head orientation is supervised by the tele-operator's head orientation, captured by a virtual reality device (see Section 5).

4.2 Reconstruction

We compute a reconstruction of the scene from all past robot observations using the publicly available *nvblox* library [7], which we extend for metric-semantic mapping in PyTorch. This library fuses posed RGB-D sensor data into a Truncated Signed Distance Field (TSDF) in real-time. For each incoming RGB-D frame, *nvblox* projects the 3D grid into the depth image and updates the distance values and weights of affected voxels (described in [27]). Figure 5 (Appendix 8.1) shows reconstructions for tasks introduced in Section 6.

Geometry: From the distance field, we extract a representation of the 3D surface. In particular, *nvblox* applies the marching cubes algorithm [28] to compute a mesh that represents the zero-level isosurface of the distance field. In this work, we only keep the mesh vertices, i.e. triangle and normal data are discarded. The result is a dense point cloud, build from the fusion of previous visual observations.

Features: To generate a metric-semantic representation of the environment, we also fuse VFM image features into the reconstructed voxel map. In particular, we extract 2D feature maps $\mathcal{F}_i$ from the incoming RGB images $\mathcal{I}_i$, using a pre-trained VFM ϕ:

$$\mathcal{F}_i = \phi(I_i), \quad \mathcal{F}_i \in \mathbb{R}^{h \times w \times f} \tag{1}$$

where f is the channel depth of the feature produced by the VFM. The feature associated with each voxel is updated by projecting the voxel center $\mathbf{p} \in \mathbb{R}^3$ into the feature map and reading the feature vector at the projected image point:

$$\mathbf{f}_i = \mathcal{F}_i[\Pi(\mathbf{p})], \quad \mathbf{f}_i \in \mathbb{R}^f \tag{2}$$

Here, $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the camera projection function, and $[.]$ denotes nearest-neighbour pixel lookup. We found that simply overwriting the existing voxel feature during updates yields similar results as to fusing the incoming feature with the existing one (see Section 6.2). Similar to the TSDF reconstruction, we handle occlusions by only updating voxels in a narrow truncation band around non-occluded surfaces (set to ± 4 voxels in our experiments). Finally, the mesh vertices are featurized by looking up their closest feature vector in the voxel map. Appendix 8.3 gives implementation details about achieving this with *nvblox*.

5 Implementation

In this section, we provide details about the implementation of our method.

Figure 3: Environments introduced to evaluate policies' spatial memory. From left to right: ***Cube Stacking*:** stack three cubes (initial cube positions are randomized), ***Mug in Drawer*** move mug into drawer containing mugs (positions of objects on kitchen counter are randomized and the destination drawer position is permuted), ***Drill in Box*:** put hand drill into open box (drill position is randomized and open/closed boxes are permuted), ***Stick in Bin*:** put candlestick into bin (stick and bin positions are randomized). In all tasks, policies are provided a single ego-centric camera view from which the entire task space cannot fit into the FOV.

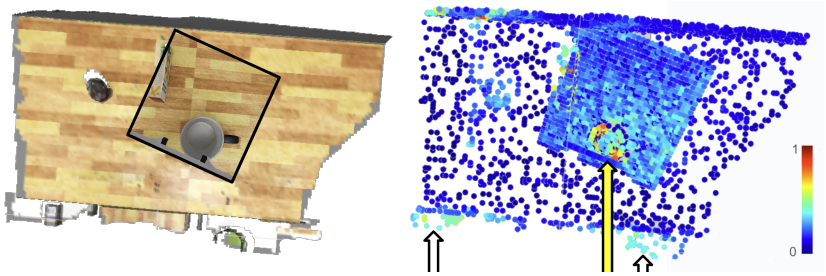

Figure 4: **Attention Visualization:** Top-down visualization of 3D attention weights (right) and reconstruction (left) for the *Mug in Drawer* task. The inset shows the current camera view. Extrema appear in regions of interest to the task, such as the mug (yellow arrow) and the drawers in the bottom left/right (white arrows). The high concentration of points in the center is generated by the current view of the camera, while points outside this region are from the reconstruction.

Demonstration Data Collection: We simulate several tasks in IsaacLab [29] to evaluate *mindmap* (see Section 6). We collect demonstration trajectories through teleoperation using IsaacLab Mimic[4] (based on MimicGen [30]), an Apple Vision Pro for the humanoid robot, and a space-mouse for the robot arm. The human demonstration trajectories are multiplied to generate a larger dataset. For each task, we train on 100 trajectories and evaluate on 100 distinct randomizations.

Reconstruction Data Generation: During training, we select a random timestamp in the demonstration trajectory and attempt to predict the next keypose based on the RGBD observation, the state history, and the reconstruction. We therefore need random access to reconstructions associated with each timestamp in the demonstration trajectories. To achieve this, we perform mapping for each demonstration trajectory and save a per-timestamp reconstruction before training. Producing a dataset of 100 trajectories (from 10 human demonstrations), including running the IsaacLab, RTX raytracing, and *nvblox* reconstruction, takes 4 hours on a single L40 GPU node, producing approximately 3000 reconstructions. TSDF reconstruction is performed at 1 cm voxel resolution.

Training: Training runs are performed on a 2-GPU H100 node for 150k iterations, taking approximately 2 days.

6 Results

In this section, we aim to validate the hypothesis of this paper, that *mindmap* improves performance on tasks that require spatial memory.

Evaluation Environments: Existing benchmarks like RLBench [31] focus on table-top manipulation tasks in which all task-relevant objects remain in view at all times. These tasks do not require spatial memory for completion because the entire state of the task can be determined from a single view.

We therefore introduce four challenging tasks on which to evaluate policies for spatial memory use (see Fig. 3). We restrict policies to ego-centric observations of the scene: the wrist camera for robot arm tasks, and to a head camera for humanoid tasks. An ego-centric camera is practical, as the robot is freed from a reliance on external infrastructure, which will become increasingly important as robots are expected to mix manipulation with movement through the environment. In our tasks, the robot is unable to see all task-relevant objects within its field of view at all times (see Fig. 6). As a consequence, the policy needs to remember the spatial layout of the scene to complete the task with a high success rate (see Appendix 8.2 for descriptions of the tasks). While this type of task is somewhat novel for manipulation policy evaluation, it is very common in everyday life; humans are frequently required to reason about out-of-view objects.

[4] isaac-sim.github.io/IsaacLab/v2.1.0/source/overview/teleop_imitation.html

Method	Average	Task			
		Cube Stacking	*Mug in Drawer*	*Drill in Box*	*Stick in Bin*
Mindmap	**79% (80%)**	**60%**	**97%**	**78%**	**82%**
3D Diffuser Actor [1]	20% (18%)	0%	46%	21%	14%
GR00T N1 [2]	- (54%)	-	-	46%	62%
Privileged (external cam) 3D Diffuser Actor [1]	85% (85%)	74%	97%	86%	83%

Table 1: **Key Findings - Evaluation in Simulation**. *Mindmap* is compared against 3D Diffuser Actor [1] and GR00T N1 [2] in simulated tasks that require spatial memory to complete with a high success rate. We also evaluate a method that uses an external camera as privileged information. The bracketed average is over humanoid tasks only.

Baselines: We compare *mindmap* with 3D Diffuser Actor [1]. For humanoid tasks, we also compare against GR00T N1 [2]. To match *mindmap*, we modify 3D Diffuser Actor to utilize AM-RADIO [25] features rather than CLIP [23], which we found to increase performance. We also compare to a version of 3D Diffuser Actor that is provided with an external camera to remove the requirement for memory on our tasks. *mindmap* and 3D Diffuser Actor are trained from scratch, while GR00T N1 is fine-tuned on each task. We attempted to fine-tune GR00T N1 on the robot arm tasks, but were unable to achieve non-zero success rates, likely because ego-centric-only robot arm tasks are not in its pretraining data. We omit these results.

6.1 Key Findings

Table 1 shows quantitative results comparing *mindmap* to the baseline methods. *mindmap* achieves an average success rate of 79%, an improvement of 59% (absolute) over 3D Diffuser Actor and 26% over GR00T N1 (on humanoid tasks). Further, *mindmap* performs only slightly (6% absolute) worse than the method that is provided with privileged information. These results, taken together, indicate the efficacy of *mindmap* at solving tasks that require spatial memory.

Three of the four tasks (*Mug in Drawer*, *Drill in Box*, and *Stick in Bin*) involve a binary decision about out-of-view objects. A policy without spatial memory is reduced to guessing between the two options seen in the training data. The results, therefore, align with expectations: allowing for the random decision, GR00T N1 achieves close to the best possible performance. Qualitatively, observation of policy roll-outs confirms this: the policy is very effective at picking up objects; however, it often ($\sim$50% of cases) makes the wrong binary decision. By contrast, *mindmap* rarely makes the wrong decision, and failures typically originate from object pick-up.

Figure 4 shows the attention weights for the *Mug in Drawer* task from the first cross-attention layer in *mindmap*. The figure indicates that network assigns a high weight to the mug to be transported, and both of the drawers, one of which is the target location. This aligns with intuition: the network attends to task-relevant parts of the scene. Note that only the mug is within the current camera view. The assignment of high weight to points outside of the current camera view also indicates the importance of the reconstruction in completing the task.

Lastly, GR00T N1 is outperformed by *mindmap* by 26% (absolute). It is pre-trained on a large dataset and is a much larger model than *mindmap* ($\sim$1B trainable parameters, plus $\sim$1B in the frozen VLM vs. *mindmap*'s $\sim$3M trainable, plus $\sim$100M frozen in the image encoder). We believe that these results indicate the potential for improving VLAs through spatial memory mechanisms.

6.2 Ablations and variations

Table 2 shows the results of varying various design decisions in our method, evaluated on our robot arm tasks.

Reconstruction only: We restrict our method to access the reconstruction only by removing the RBGD pointcloud input to our model. This leads to a 16% lower success rate. Qualitatively, we observe an increased frequency of failure during pick-up. This aligns with intuition: the wrist

Ablation	Average	Task	
		Cube Stacking	*Mug in Drawer*
mindmap (baseline)	79%	60%	97%
Reconstruction only	63%	33%	93%
No VFM	45%	31%	59%
Feature blending	74%	50%	98%

Table 2: **Ablations and Variations**. Variations of design parameters of *mindmap* and their corresponding success rates on the robot arm tasks introduced in Section 6. *Reconstruction only*: removal of RGBD observations. *No VFM*: features replaced with RGB triplets. *Feature blending*: blends VFM features over time, rather than taking the latest observed feature.

camera provides high-resolution information during object pick-up, which is likely important for accurate grasping.

No VFM: RADIO-AM features are replaced with RGB triplets extracted from the images. This leads to a 34% lower success rate. The relative reduction in success is less pronounced for *Cube Stacking* than for *Mug in Drawer*, likely due to the distinct RGB colors of the cubes providing sufficient information for the model in most cases. In general, compared to semantically rich features like RADIO-AM, raw RGB does not take any contextual or semantic information into account, and its values strongly depend on lighting conditions and viewing direction.

Feature blending: During reconstruction, our baseline method overwrites existing feature vectors with the most recently extracted ones. As an alternative, we explored fusing new measurements with old ones. Here, we update the feature associated with each voxel by applying an exponential filter:

$$\mathbf{f}_{\text{voxel}}(\mathbf{p}) \leftarrow \alpha \cdot \mathcal{F}[\Pi(\mathbf{p})] + (1-\alpha) \cdot \mathbf{f}_{\text{voxel}}(\mathbf{p}) \tag{3}$$

We use $\alpha = 0.1$, i.e., a new measurement contributes 10% to the updated value. We found that this modification leads to no significant change in performance.

6.3 Limitations

Our method has several limitations. Firstly, our model is small (3 million trainable parameters), is trained on a small dataset, and in a task-specific regime. Policies of this kind [1, 9, 20, 15, 14, 21, 22] are convenient to perform research on, however, do not in general, generalize out of their training environment. It is an interesting research direction to scale up *mindmap* to a larger dataset such as DROID [17]. Secondly, our model produces end-effector keyposes as output. Keypose extraction from VR teleop data is non-trivial and task-specific. Altering the model to predict trajectories using action-chunking [11], as is common in VLAs, has the potential to remove the limiting step. Lastly, our reconstruction process is non-differentiable. The result is that we store a full VFM feature *per-voxel*, which requires substantial amounts of storage during training and memory during inference. There is an opportunity, with a differentiable reconstruction process, to do learned dimensionality reduction before reconstruction to reduce memory consumption.

7 Conclusions

In this paper, we present *mindmap*, a manipulation policy that diffuses robot trajectories from a reconstruction of the observed scene. We showed that tasks involving spatial memory are challenging for methods that compute trajectories based on the current observation only. *Mindmap* is able to utilize past information, in the form of the metric-semantic reconstruction, in order to complete tasks that involve reasoning about out-of-view objects. The result is that *mindmap* significantly improves performance on spatial memory evaluations. We contribute our tools for metric-semantic mapping and for training reconstruction-based diffusion policies to spur further research in this direction. We foresee a growing importance of spatial memory as learned manipulation policies move beyond the tabletop tasks, in particular to tasks that combine locomotion and manipulation.

Acknowledgments

We would like to thank the 3D Diffuser Actor [1] authors for open-sourcing their code, in particular Nikolaos Gkanatsios for his generosity with his time, and for the fruitful discussions about 3D manipulation policies.

References

[1] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[2] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[5] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[6] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *arXiv preprint arXiv:2502.10550*, 2025.

[7] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart. nvblox: Gpu-accelerated incremental signed distance field mapping. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2698–2705, 2024.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[10] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[11] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[12] K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[13] L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

[14] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[15] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[16] R. Yang, G. Chen, C. Wen, and Y. Gao. Fp3: A 3d foundation policy for robotic manipulation. *arXiv preprint arXiv:2503.08950*, 2025.

[17] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[18] 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 *7th Annual Conference on Robot Learning*, 2023.

[19] O. Shorinwa, J. Tucker, A. Smith, A. Swann, T. Chen, R. Firoozi, M. Kennedy III, and M. Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. *arXiv preprint arXiv:2405.04378*, 2024.

[20] 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 *Conference on robot learning*, pages 284–301. PMLR, 2023.

[21] H. Fang, M. Grotz, W. Pumacay, Y. R. Wang, D. Fox, R. Krishna, and J. Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

[22] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

[23] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[25] M. Ranzinger, G. Heinrich, J. Kautz, and P. Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12490–12500, 2024.

[26] T.-W. Ke, N. Gkanatsios, J. Xu, and K. Fragkiadaki. Bi3d diffuser actor: 3d policy diffusion for bi-manual robot manipulation. In *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.

[27] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[28] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In M. C. Stone, editor, *SIGGRAPH*, pages 163–169. ACM, 1987. ISBN 0-89791-227-6. URL http://dblp.uni-trier.de/db/conf/siggraph/siggraph1987.html#LorensenC87.

[29] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[30] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[31] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

8 Appendix

8.1 Example Reconstructions

Reconstructions of our environments can be seen in Fig. 5.

8.2 Evaluation task descriptions

We introduce several tasks specifically designed to test for systems for their ability to leverage spatial memory. See Fig. 6 for visualizations of the tasks. In particular, we introduce:

- ***Cube Stacking*** **(robot arm):** Requires the policy to stack three cubes in order. Cube positions are randomized. The policy only has an egocentric view, and as a result, the policy must remember the position of the ongoing stack during cube transport, during which time the camera is blocked.
- ***Mug in Drawer*** **(robot arm):** The goal of the task is to return a mug to a drawer that contains mugs. The target drawer is permuted between two options. The policy only has an egocentric view, and as a result, the policy must remember which of the two drawers is correct during transport of the mug.
- ***Drill in Box*** **(humanoid):** This task requires the humanoid robot to pick up an electric drill off the shelf and place it in an open box. Which box is open is randomly permuted among four options. To identify which is the correct box, the humanoid must actively scan its surroundings by rotating its head to detect the open box, memorize its location, and subsequently transport the drill to that position.
- ***Stick in Bin*** **(humanoid):** Similar to above. The humanoid robot must place a candlestick in a bin. The bin is randomly placed in a position around the robot. Successful task completion requires first scanning the scene, memorizing the layout, before transporting the stick.

8.3 Reconstructing with nvblox-PyTorch

nvblox [7] is an open source library for real-time 3D reconstruction, designed for robotic applications. It provides functions for building, manipulating and querying 3D reconstructions directly on the GPU. The following snippet demonstrates how *mindmap* makes use of the recently added PyTorch bindings to generate a featurized 3D reconstruction.

```python
# Install nvblox_torch from pip
from nvblox_torch import Mapper, FeatureMesh

# Create a mapper.
mapper = Mapper(voxel_sizes_m=[0.01])

# Add depth and feature frames to the reconstruction.
for depth_frame, feature_frame, pose, intrinsics in dataset:
    mapper.add_depth_frame(depth_frame, pose, intrinsics)
    mapper.add_feature_frame(feature_frame, pose, intrinsics)

# Compute a surface mesh representation of the scene.
mapper.update_feature_mesh()
mesh = mapper.get_feature_mesh()

# Obtain features and vertices as PyTorch tensors.
vertices = mesh.vertices()
features = mesh.features()
```

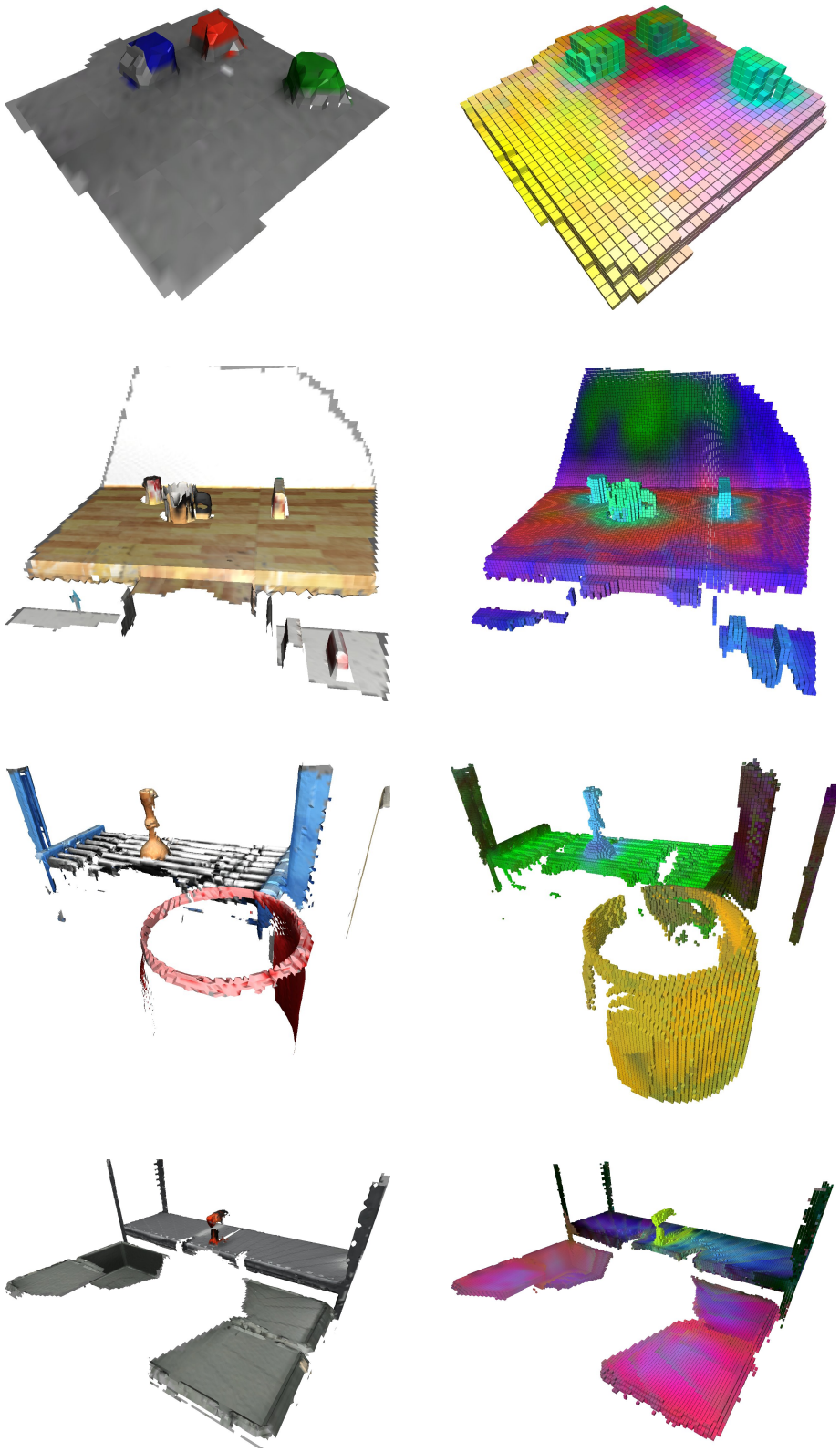

Figure 5: Reconstructions of the four environments presented in Section 6. For each environment, we have an RGB-colored mesh (top) and the voxel grid containing VFM features colored by PCA (bottom).

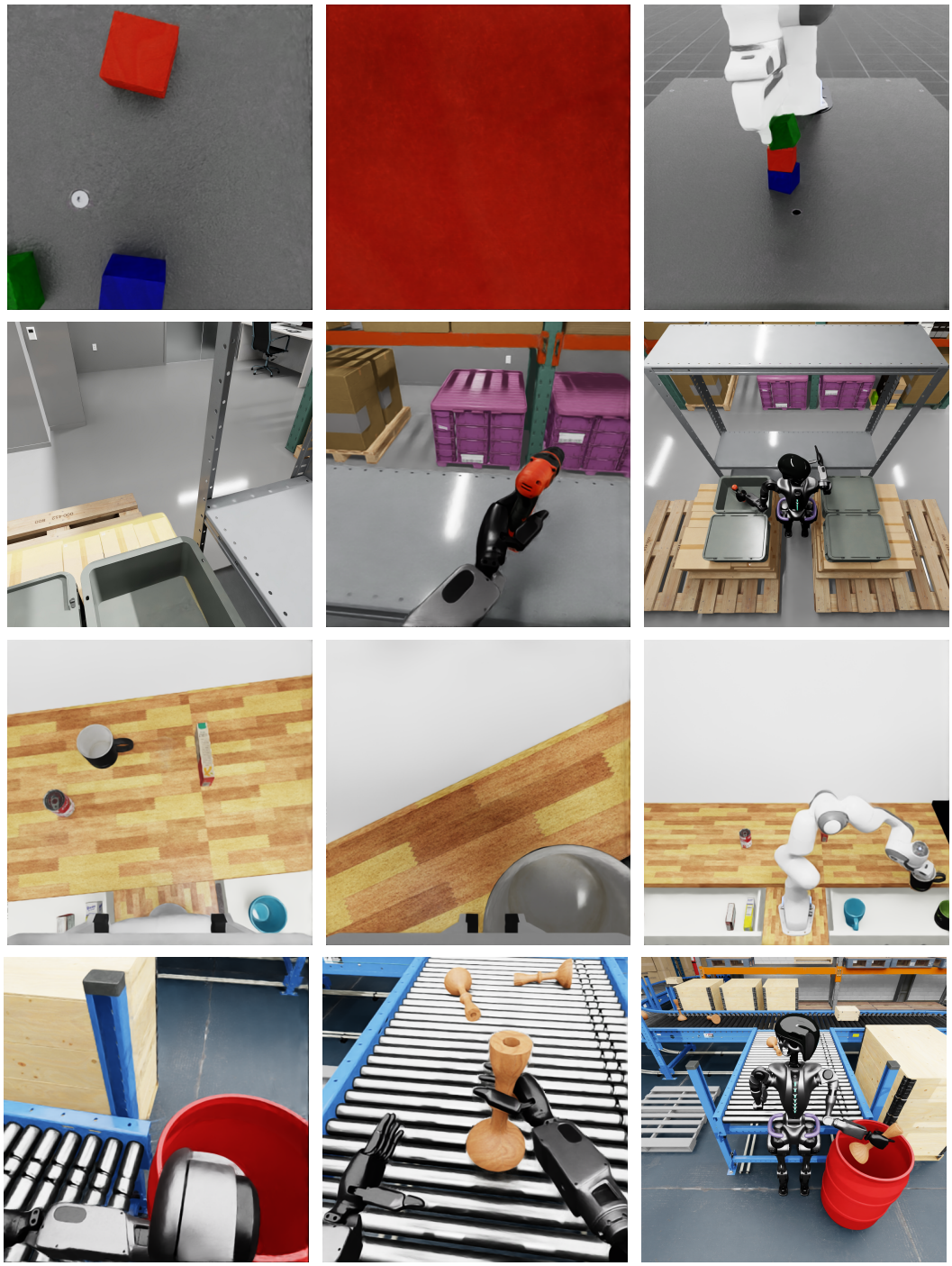

Figure 6: Views from four evaluation environments. Each row corresponds to a distinct environment. The first column presents an ego-centric perspective of the drop off locations, whose positions must be memorized. The second column shows ego-centric observations during task execution, where parts of these objects are no longer visible. The third column presents a third-person view of the robot performing the task.