DynaMem: Online Dynamic Spatio-Semantic Memory for Open World Mobile Manipulation

Anonymous Author(s)
Affiliation
Address
`email`

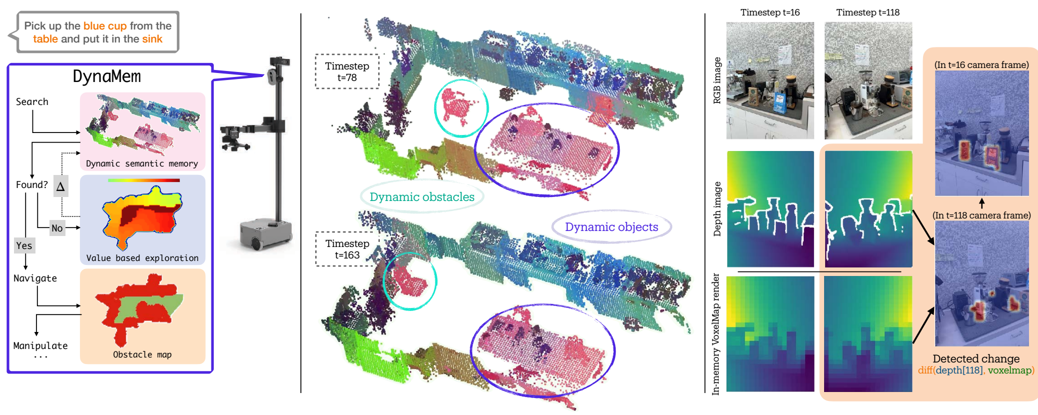

Figure 1: An illustration of how our online dynamic spatio-semantic memory DynaMem responds to open vocabulary queries in a dynamic environment. During operation and exploration, DynaMem keeps updating its semantic map in memory. DynaMem maintains a voxelized pointcloud representation of the environment, and updates with dynamic changes in the environment by adding and removing points.

Abstract: Significant progress has been made in open-vocabulary mobile manipulation, where the goal is for a robot to perform tasks in any environment given a natural language description. However, most current systems assume a static environment, which limits the system's applicability in real-world scenarios where environments frequently change due to human intervention or the robot's own actions. In this work, we present DynaMem, a new approach to open-world mobile manipulation that uses a dynamic spatio-semantic memory to represent a robot's environment. DynaMem constructs a 3D data structure to maintain a dynamic memory of point clouds, and answers open-vocabulary object localization queries using multimodal LLMs or open-vocabulary features generated by state-of-the-art vision-language models. Powered by DynaMem, our robots can explore novel environments, search for objects not found in memory, and continuously update the memory as objects move, appear, or disappear in the scene. We run extensive experiments on the Stretch SE3 robots in three real and nine offline scenes, and achieve an average pick-and-drop success rate of 70% on non-stationary objects, which is more than a **2**$\times$ improvement over state-of-the-art static systems.

1 Introduction

Recent advances in robotics have made it possible to deploy robots in real world settings to tackle the open vocabulary mobile manipulation (OVMM) problem [1]. Here, the robots are tasked with navigating in unknown environments and interacting with objects following open vocabulary language instructions, such as "Pick up **X** from **Y** and put it in **Z**", where X, Y, and Z could be any object name or location. The two most common approaches to tackling OVMM are using policies trained in sim-

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

ulation and deploying them in the real world [2, 3, 4], or training modular systems that combine open vocabulary navigation (OVN) [5, 6, 7, 8] with different robot manipulation skills [9, 10, 11, 12, 13]. Modular systems enjoy greater efficiency and success in real-world deployment [14] as they can directly leverage advances in vision and language models [9, 12], and are able to handle more diverse and out-of-domain environments with no additional training.

However, as recent analysis has shown, the primary challenge in deploying modular OVMM is that limitations of a module propagate to the entire system [9]. One key module in any OVMM system is the open vocabulary navigation (OVN) module responsible for navigating to goals in the environment. While many such OVN systems have been proposed in the literature [1, 8, 5, 11, 10, 9, 6, 7, 12, 13], they share a common limitation: they assume static, unchanging environments. Contrast this with the real world, where environments change and objects are moved by either robots or humans. Making such a restrictive assumption thus limits these systems' applicability in real-world settings. The primary reason behind this assumption is the lack of an effective dynamic spatio-semantic memory that can adapt to both addition and removal of objects and obstacles in the environment online.

In this work, we propose a novel spatio-semantic memory architecture, Dynamic 3D Voxel Memory (DynaMem), that can adapt online to changes in the environment. DynaMem maintains a voxelized pointcloud representation of an environment and adds or removes points as it observes the environment change. Additionally, it supports two different ways to query the memory with natural language: a vision-language model (VLM) featurized pointcloud, and a multimodal-LLM (mLLM) QA system. Finally, DynaMem enables efficient exploration in changing environments by offering a dynamic obstacle map and a value-based exploration map that can guide the robot to explore unseen, outdated, or query-relevant parts of the world.

We evaluate DynaMem as a part of full open-vocabulary mobile manipulation stack in three real world environments with multiple rounds of changes and manipulating multiple non-stationary objects, improving the static baseline by more than $2\times$ (70% vs. 30%). Additionally, we identify an obstacle in efficiently developing dynamic spatio-semantic memory, namely the lack of dynamic benchmarks, since many OVN systems use static simulated environments [15, 16] or static datasets [17, 18]. We address this by developing a new dynamic benchmark, DynaBench. It consists of 9 different environments, each changing over time. We ablate our design choices in this benchmark. To the best of our knowledge, DynaMem is the first spatio-semantic memory structure supporting both adding and removing objects.

2 Method

In this section, we define our problem setup, and then describe our online, dynamic spatio-semantic memory for open world, open vocabulary mobile manipulation. We introduce how to use this memory to localize text query and how to navigate to the target object in Appendix 6 and 7 respectively.

2.1 Problem Statement

We create our algorithm, DynaMem, to solve open vocabulary mobile manipulation (OVMM) problems in an open, constantly changing world. The goal in OVMM is for a mobile robot to execute a series of manipulation commands given arbitrary language goals. We assume the following requirements for the memory module for dynamic, online operation:

- **Observations:** The mobile robot is equipped with an on-board RGB-D camera, and unlike prior work [9], doesn't start with a map of the environment. Rather, the robot explores the world and use the online observed sequence of posed RGB-D images to build its map.

- **Environment dynamism:** The environment can change without the knowledge of the robot.

- **Localization queries:** Given a natural language query (i.e. "teddy bear"), the memory module has to return the 3D location of the object or determine that the object doesn't exist in the scene observed thus far.

- **Obstacle queries:** The memory module must determine whether a point in space is occupied by an obstacle. Both the location of the objects and obstacles can move, previous observations often contradict each other and must be resolved by the memory.

2.2 Dynamic 3D Voxel Map

Our answer to the challenge posed in the Section 2.1 is DynaMem. DynaMem is an evolving sparse voxel map with associated information stored at each voxel, as shown in Fig. 6. In each non-empty voxel, alongside its 3D location (x, y, z), we also store the observation count C (how many times that voxel was observed), source image ID I (which image the voxel was backprojected from), a high-dimensional semantic feature vector f coming from a VLM like CLIP [19] or SigLIP [20], and the latest observation time, t, in seconds.

To make this data structure dynamic, we describe the process with which we add and update with new observations and remove outdated objects and associated voxels.

Adding Points: When the robot receives a new set of observations, i.e. RGB-D images with global poses, we convert them to 3D coordinates in a global reference frame, and generate a semantic feature vector for each point. The global coordinates are calculated from the global camera pose and the backprojected depth image using the known camera transformation matrix. We calculate the point-wise image feature by first converting the images to object patches by using a segmentation model such as SAM-v2 [21], and then aggregating each patch feature over the output of a vision-language models like CLIP [19] or SigLIP [20]. For more details about image-to-feature vector mapping, we refer to earlier works [5, 9, 8]. Once we have calculated the points and associated features, we cluster the new points and assign them to the nearest voxel grids. In Fig. 7, we show how each voxel's metadata is updated. The count keeps track of the total number of assigned points to each voxel grid, and the feature vector keeps track of the weighted average of all feature vectors assigned to that voxel. Finally, the observation time and image ID are updated to keep track of the latest observation contributing to a particular voxel. If a voxel was empty before assignment, we assume its count $C = 0$ and feature vector $f = \overrightarrow{0}$.

Removing Points: When an object is moved or removed, its associated voxels in DynaMem may get removed. We use ray-casting to find the outdated voxels. The operation follows a simple principle: if a voxel falls within the frustum between the camera plane and the associated view point cloud, that voxel must be unoccupied. To reduce the impact of the depth noise at long range, we don't consider any pixel whose associated depth value is over 2m.

We illustrate a simplified 2D representation of this algorithm in Fig. 2. In practice, to speed up the intersection between the sparse voxelmap and the view frustum, we project each existing voxel to the camera plane and calculate the camera distance. If the image height and width are (H, W), the depth image is $\mathbf{D}$, and a certain voxel is projected to points (h, w) in the camera plane with depth d, it gets removed if both Eq. 1 and 2 hold.

$$(h, w) \in [0, H] \times [0, W] \tag{1}$$

$$d \in \big(0, \min(2, \mathbf{D}\big[h, w\big] + \epsilon)\big) \tag{2}$$

Where Eq. 1 ensures that the point falls within the camera view, and Eq 2 ensures that (a) the depth $d > 0$, or the object is in front of camera, (b) $d < 2$m, or the voxel isn't too far away from the camera, and (c) $d < \mathbf{D}[h, w]$ denoting the voxel is between the camera and the currently visible object.

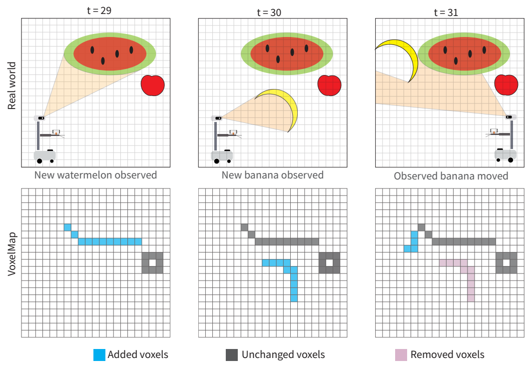

Figure 2: A high-level, 2D depiction of how adding and removing voxels from the voxel map works. New voxels are included which are in the RGB-D cameras view frustum, and old voxels that should block the view frustum but does not are removed from the map.

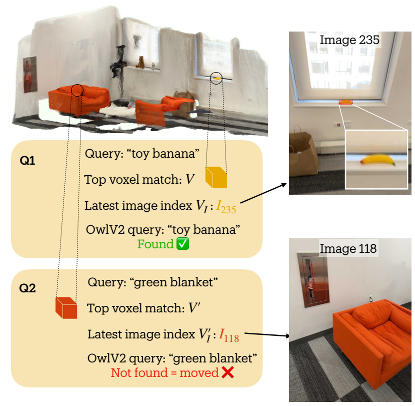

Figure 3: Querying DynaMem with a natural language query. First, we find the voxel with the highest alignment to the query. Next, we find the latest image of that voxel, and query with an open-vocabulary object detector to confirm the object location or abstain.

3 Experiments

We evaluate our method, DynaMem, on a Hello Robot: Stretch SE3 in real world environments. We also perform a series of ablation experiments in an offline benchmark in Appendix 8.

3.1 Real-world Experiments

As a baseline, we compare with OK-Robot [9], a state-of-the-art method for OVMM. OK-Robot uses a static voxelmap as its memory representation, and thus it highlights the importance of dynamic memory for OVMM in a changing environment. For DynaMem, we run two variations of the algorithm in the real world: one with VLM-feature based queries and one with mLLM-QA based queries.

We describe detailed experiment setup in Appendix 9.

Results: Our experiments in three dynamic environments and with 30 queries is summarized in Fig. 4. We find that DynaMem with both VLM-feature based and mLLM-QA based queries have a total success rate of 70%. This is a significant improvement over the OK-Robot system, which has a total success rate of 30%. Notably, DynaMem is particularly adept at handling dynamic objects in the environment: only 6.7% of the trials failed due to our system not being able to navigate to such dynamic objects in the scene. This is in contrast to the OK-Robot system, where 53.3% of the trials failed because it could not find an object that moved in the environment. In contrast, navigating to static goals fails in only 10% of the cases for DynaMem with VLM-feature, 13.3% for OK-Robot and 20% for DynaMem with mLLM-QA.

4 Conclusions

In this work, we introduced DynaMem, a spatio-semantic memory for open-vocabulary mobile manipulation that can handle changes to the environment during operation. We showed in three real world environments that DynaMem can navigate to, pick, and drop objects even while object and obstacle locations are changing.

References

[1] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, Z. Kira, M. Savva, A. Chang, D. S. Chaplot, D. Batra, R. Mottaghi, Y. Bisk, and C. Paxton. Homerobot: Open-vocabulary mobile manipulation, 2024. URL https://arxiv.org/abs/2306.11565.

[2] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv:2312.02976*, 2023.

[3] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17896–17906, June 2023.

[4] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs. Poliformer: Scaling on-policy rl with transformers results in masterful navigators, 2024. URL https://arxiv.org/abs/2406.20083.

[5] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory, 2023. URL https://arxiv.org/abs/2210.05663.

[6] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023. URL https://arxiv.org/abs/2309.16650.

[7] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone. Clio: Real-time task-driven open-set 3d scene graphs, 2024. URL https://arxiv.org/abs/2404.13696.

[8] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[9] 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, 2024. URL https://arxiv.org/abs/2401.12202.

[10] R.-Z. Qiu, Y. Hu, G. Yang, Y. Song, Y. Fu, J. Ye, J. Mu, R. Yang, N. Atanasov, S. Scherer, and X. Wang. Learning generalizable feature fields for mobile manipulation, 2024. URL https://arxiv.org/abs/2403.07563.

[11] B. Bolte, A. Wang, J. Yang, M. Mukadam, M. Kalakrishnan, and C. Paxton. Usa-net: Unified semantic and affordance representations for robot memory, 2023. URL https://arxiv.org/abs/2304.12164.

[12] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra, R. Mottaghi, J. Malik, and D. S. Chaplot. Goat: Go to any thing, 2023. URL https://arxiv.org/abs/2311.06430.

[13] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation, 2024. URL https://arxiv.org/abs/2403.17846.

[14] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot. Navigating to objects in the real world. *Science Robotics*, 8(79):eadf6991, 2023. doi:10.1126/scirobotics.adf6991. URL https://www.science.org/doi/abs/10.1126/scirobotics.adf6991.

[15] D. Z. Chen, A. X. Chang, and M. Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language, 2020. URL https://arxiv.org/abs/1912.08830.

[16] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017. URL https://arxiv.org/abs/1702.04405.

[17] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, et al. Habitat-matterport 3d semantics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4927–4936, 2023.

[18] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

[19] 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, 2021. URL https://arxiv.org/abs/2103.00020.

[20] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

[21] 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. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

[22] S. Yenamandra, A. Ramachandran, M. Khanna, K. Yadav, D. S. Chaplot, G. Chhablani, A. Clegg, T. Gervet, V. Jain, R. Partsey, R. Ramrakhya, A. Szot, T.-Y. Yang, A. Edsinger, C. Kemp, B. Shah, Z. Kira, D. Batra, R. Mottaghi, Y. Bisk, and C. Paxton. The homerobot open vocab mobile manipulation challenge. In *Thirty-seventh Conference on Neural Information Processing Systems: Competition Track*, 2023. URL https://aihabitat.org/challenge/2023_homerobot_ovmm/.

[23] N. Yokoyama, A. Clegg, J. Truong, E. Undersander, T.-Y. Yang, S. Arnaud, S. Ha, D. Batra, and A. Rai. ASC: Adaptive skill coordination for robotic mobile manipulation. *arXiv preprint arXiv:2304.00410*, 2023.

[24] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation, 2023. URL https://arxiv.org/abs/2210.05714.

[25] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, pages 259–274. Springer, 2020.

[26] J. Krantz, S. Lee, J. Malik, D. Batra, and D. S. Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances. *arXiv preprint arXiv:2211.15876*, 2022.

[27] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta. No rl, no simulation: Learning to navigate without navigating. *Advances in Neural Information Processing Systems*, 34:26661–26673, 2021.

[28] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4247–4258. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/2c75cf2681788adaca63aa95ae028b22-Paper.pdf.

[29] N. Yokoyama, S. Ha, and D. Batra. Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1562–1569, 2021. doi:10.1109/IROS51168.2021.9636743.

[30] X. Zhao, H. Agrawal, D. Batra, and A. G. Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16127–16136, October 2021.

[31] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *CoRR*, abs/2006.13171, 2020. URL `https://arxiv.org/abs/2006.13171`.

[32] A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi, A. PS, G. K. Yadav, R. Kala, and R. Haschke. Uniteam: Open vocabulary mobile manipulation challenge, 2023. URL `https://arxiv.org/abs/2312.08611`.

[33] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *International Journal of Robotic Research - IJRR*, 31:647–663, 04 2012. doi:10.1177/0278364911434148.

[34] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic slam. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1722–1729. IEEE, 2017.

[35] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song. Semantic slam based on object detection and improved octomap. *IEEE Access*, 6:75545–75559, 2018.

[36] L. Ma, J. Stückler, C. Kerl, and D. Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2017.

[37] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[38] H. Ha and S. Song. Semantic abstraction: Open-world 3d scene understanding from 2d vision-language models, 2022.

[39] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory, 2023.

[40] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[41] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. In *arXiv preprint arXiv:2209.09874*, 2022.

[42] K. M. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[43] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023.

[44] M. Ji, R.-Z. Qiu, X. Zou, and X. Wang. Graspsplats: Efficient manipulation with 3d feature splatting. *arXiv preprint arXiv:2409.02084*, 2024.

[45] O. Shorinwa, J. Tucker, A. Smith, A. Swann, T. Chen, R. Firoozi, M. D. Kennedy, and M. Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. In *8th Annual Conference on Robot Learning*, 2024.

[46] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[47] B. Bolte, A. Wang, J. Yang, M. Mukadam, M. Kalakrishnan, and C. Paxton. Usa-net: Unified semantic and affordance representations for robot memory, 2023.

[48] Y. Wang, Z. Li, M. Zhang, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li. D3 fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023.

[49] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[50] Z. Song, G. Zhang, J. Xie, L. Liu, C. Jia, S. Xu, and Z. Wang. Voxelnextfusion: A simple, unified and effective voxel fusion framework for multi-modal 3d object detection. *arXiv preprint arXiv:2401.02702*, 2024.

[51] W. Shi, J. Xu, D. Zhu, G. Zhang, X. Wang, J. Li, and X. Zhang. Rgb-d semantic segmentation and label-oriented voxelgrid fusion for accurate 3d semantic mapping. *IEEE transactions on circuits and systems for video technology*, 32(1):183–197, 2021.

[52] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level slam. In *2018 international conference on 3D vision (3DV)*, pages 32–41. IEEE, 2018.

[53] G. S. Krishna, K. Supriya, and S. Baidya. 3ds-slam: A 3d object detection based semantic slam towards dynamic indoor environments. *arXiv preprint arXiv:2310.06385*, 2023.

[54] E. Michael, T. Summers, T. A. Wood, C. Manzie, and I. Shames. Probabilistic data association for semantic slam at scale. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4359–4364. IEEE, 2022.

[55] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4018–4025. IEEE, 2023.

[56] A. Rosinol, J. J. Leonard, and L. Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023.

[57] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.

[58] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.

[59] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer. Airdos: Dynamic slam benefits from articulated objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8047–8053. IEEE, 2022.

[60] L. Cui and C. Ma. Sof-slam: A semantic visual slam for dynamic environments. *IEEE access*, 7:166528–166539, 2019.

[61] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari. Semantic monocular slam for highly dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 393–400. IEEE, 2018.

[62] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1168–1174. IEEE, 2018.

[63] S. Song, H. Lim, A. J. Lee, and H. Myung. Dynavins: a visual-inertial slam for dynamic environments. *IEEE Robotics and Automation Letters*, 7(4):11523–11530, 2022.

[64] P. Yu, C. Guo, y. Liu, and H. Zhang. Fusing semantic segmentation and object detection for visual slam in dynamic scenes. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–7, 2021.

[65] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018.

[66] L. Schmid, M. Abate, Y. Chang, and L. Carlone. Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments. In *Proc. of Robotics: Science and Systems*, 2024.

[67] J. C. Virgolino Soares, V. S. Medeiros, G. F. Abati, M. Becker, G. Caurin, M. Gattass, and M. A. Meggiolaro. Visual localization and mapping in dynamic and changing environments. *Journal of Intelligent & Robotic Systems*, 109(4):95, 2023.

[68] B. Bescos, C. Campos, J. D. Tardós, and J. Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *IEEE robotics and automation letters*, 6(3):5191–5198, 2021.

[69] M. Henein, J. Zhang, R. Mahony, and V. Ila. Dynamic slam: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129. IEEE, 2020.

[70] D. F. Henning, T. Laidlow, and S. Leutenegger. Bodyslam: Joint camera localisation, mapping, and human motion tracking. In *European Conference on Computer Vision*, pages 656–673. Springer, 2022.

[71] J. Haviland, N. Sünderhauf, and P. Corke. A holistic approach to reactive mobile manipulation. *IEEE Robotics and Automation Letters*, 7(2):3122–3129, 2022.

[72] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[73] Y. Du, D. Ho, A. Alemi, E. Jang, and M. Khansari. Bayesian imitation learning for end-to-end mobile manipulation. In *International Conference on Machine Learning*, pages 5531–5546. PMLR, 2022.

[74] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022.

[75] S. Uppal, A. Agarwal, H. Xiong, K. Shaw, and D. Pathak. Spin: Simultaneous perception, interaction and navigation, 2024. URL https://arxiv.org/abs/2405.07991.

[76] O. Team. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[77] G. T. Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

[78] J. Yang, X. Chen, S. Qian, N. Madaan, M. Iyengar, D. F. Fouhey, and J. Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent, 2023. URL https://arxiv.org/abs/2309.12311.

[79] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[80] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16488–16498, 2024.

[81] M. Minderer, A. Gritsenko, and N. Houlsby. Scaling open-vocabulary object detection, 2024. URL https://arxiv.org/abs/2306.09683.

[82] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48. IEEE, 2024.

[83] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023.

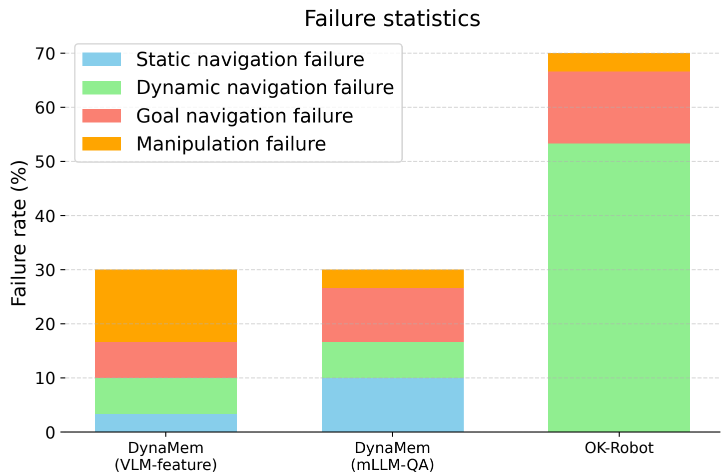

Figure 4: Statistics of failure, broken down by failure modes, in our real robot experiments. Statistics are collected over three environments and 30 open-vocabulary pick-and-drop queries on objects whose locations change over time.

5 Related Works

5.1 Open Vocabulary Mobile Manipulation (OVMM)

Navigating to arbitrary goals in open ended environments and manipulating them has become a key challenge in robotic manipulation [22, 23]. This line of query follows Open-Vocabulary Navigation systems [5, 24], which builds upon prior object and point goal navigation literature [14, 25, 26, 27, 28, 29, 30, 31, 12] which attempted navigation to points, or fixed set of objects and object categories. OVMM is a naturally harder challenge as it requires an ability to handle arbitrary queries, and "navigation to manipulation" transfer – which means unlike pure navigation, the robot needs to get close to the environment objective and obstacles. In the OVMM challenge [22], modular solutions such as [1, 32, 13] outperformed the competition. More recently, OK-Robot [9] performed extensive real-world evaluations of the challenges in OVMM and demonstrated a system that achieves 58.5% success rate in static home environments. We extend this work by enabling manipulation in changing environments.

5.2 Spatio-semantic Memory

Early works in spatio-semantic memory [33, 34, 35, 36, 37] created semantic maps for limited categories based on mostly ad-hoc deep neural networks. Later work builds upon representations derived from pre-trained vision language models, such as [38, 39, 40, 41, 42, 43, 6, 7]. These works use a voxel map or neural feature field as their map representation. Some recent models [44, 45] have used Gaussian splats [46] to represent semantic memory for manipulation. Most of these models show object localization in pre-mapped scenes, while CLIP-Fields [5], VLMaps [24], and NLMap-SayCan [41] show integration with real robots for indoor navigation tasks. Some recent works [47, 48, 10] extend this task to include an affordance model or manipulation primitives. Our work builds upon the voxel map based spatio-semantic memory literature and extends them to dynamic environments where both objects and obstacles can change over time.

5.3 Mapping and Navigating Dynamic Environments

For robot navigation, Simultaneous Localization and Mapping (SLAM) [49] methods are crucial. However, practical SLAM instances based on voxels [50, 51], objects [52, 53], landmark [34, 54], NeRF [55, 56], and Gaussian splats [57, 58] tend to make the simplifying assumption that the world

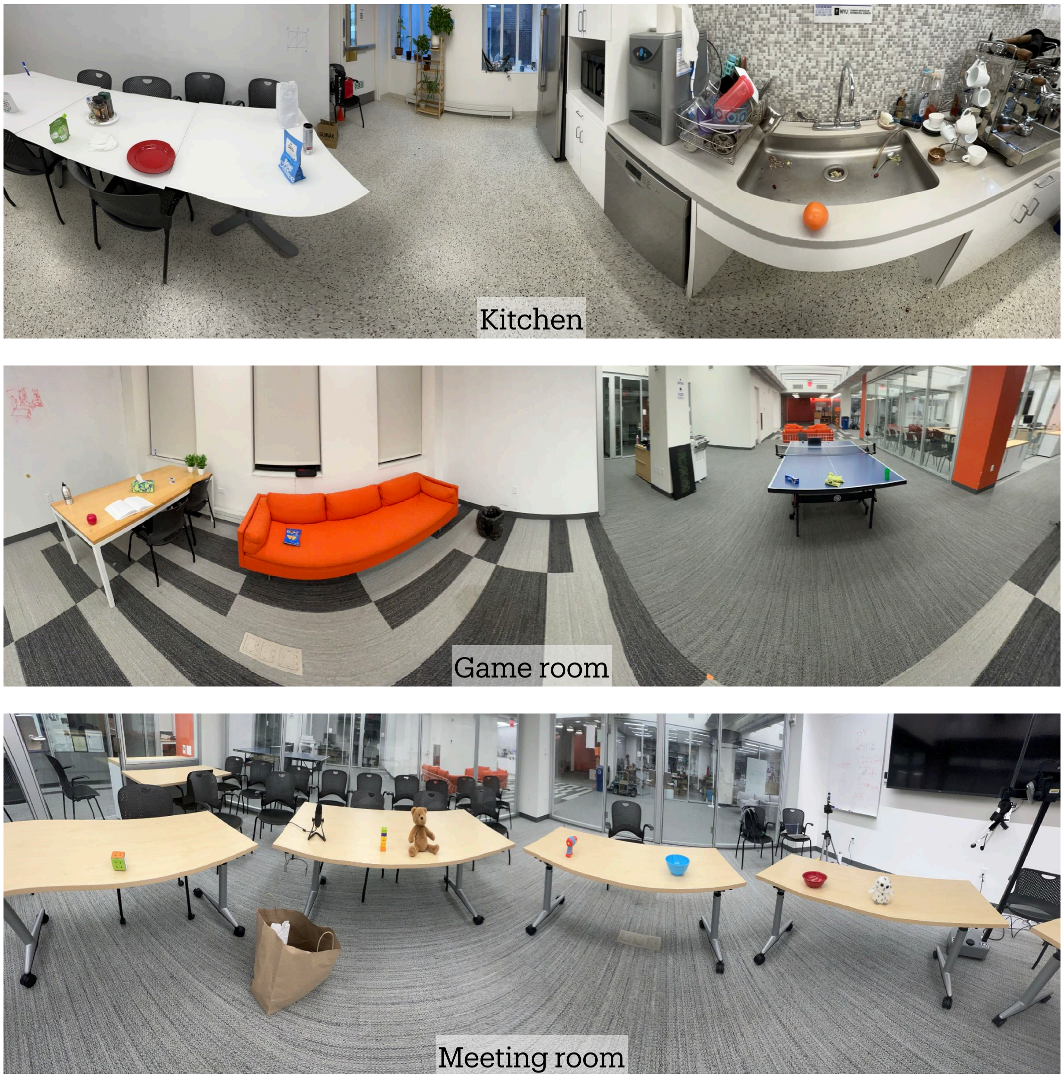

Figure 5: Real robot experiments in three different environments: kitchen, game room, and meeting room. In each environment, we modify the environment thrice and run 10 pick-and-drop queries.

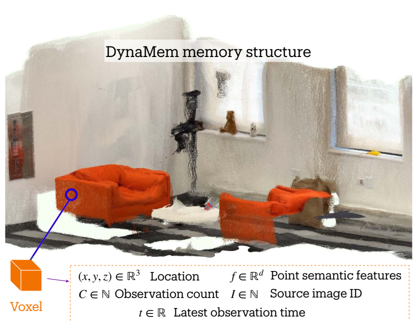

Figure 6: DynaMem keeps its memory stored in a sparse voxel grid with associated information at each voxel.

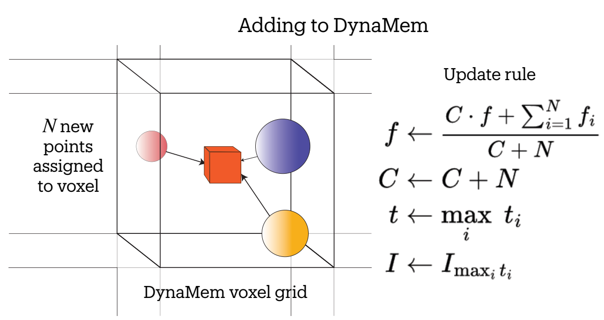

Figure 7: Updating DynaMem by adding new points to it, alongside the update rules for the stored information.

is static. Some sparse SLAM methods improve on dynamic environments by estimating underlying state [59, 60, 61, 62, 63, 64, 65, 66, 67] or explicitly modeling moving objects [68, 69, 70]. Some methods also forego a map and rely on reactive policies to navigate dynamic environments [71, 72, 73, 74, 75], although they generally tackle local movement and not global navigation. Our work relies on SLAM systems that are stable under environment dynamics, and focuses on building a dynamic semantic memory based off of online exploration and observations.

6 Querying DynaMem for Object Localization

As described in Section 2.1, we define the object localization or 3D visual grounding problem as a function mapping a text query and posed RGBD images to either the 3D coordinate of the query object, or $\emptyset$ if the object is not in the scene. Unlike previous work, we abstain from returning a location when an object is not found. To enable this, we factor this grounding problem into two sub-problems. The first is finding the latest image where the queried object could have appeared. The second is identifying whether the object is actually present in that image. For the first sub-problem, we propose two alternate approaches of visual grounding: one using the intrinsic semantic features of DynaMem, and another using state-of-the-art multimodal LLMs such as GPT-4o [76] and Gemini 1.5 Pro [77].

Embedded Vision Language Features: Vision Language Models (VLMs) such as CLIP [19] and SigLIP [20] possess an ability to embed both images and languages into the same latent space, where the similarity between an image and a text object can be calculated by simply taking the dot product between the two latent representation vectors. We use this property of the embedding vectors to query our voxel map with open-vocabulary text queries.

As described in Section 2.2, we convert the incoming images to point-wise image features, and embed them into our voxels. When we have a new language query, we calculate its latent embedding using the VLM text encoder, and find the voxel whose feature has the highest dot product with the text embedding. Once we find the right voxel, we simply retrieve its associated latest image from our data structure as shown in Fig. 3.

As a bonus feature, we can also return $n > 1$ possible objects for a single query. We do this by using a DBSCAN clustering of voxels similar to [78], and returning the images associated with the most aligned voxel in top-n clusters.

Multimodal Large Language Models (mLLMs): We note that the problem of finding the latest image where an object may appear is similar to the problem of visual question-answer (VQA) [79]. Since we fully rely on pretrained models to build our map, we pose this multi-image VQA problem as an mLLM QA problem similar to OpenEQA [80].

We show in Fig. 8 how we query the mLLMs to solve the visual grounding query. We give the model a sequence of our latest environment observations images and ask the model for the index of the last image where the queried object was observed. We additionally instruct the model to respond "None" if the object was not observed in any image. Note that, unlike OpenEQA [80], we only pass the RGB images to the mLLM, and not the depth or camera pose. Similarly, we only ask for an image index, and not a full textual answer.

Handling Absence of Object: Several previous methods [5, 8, 9] assume that the queried object is always present in the scene, and always responds with the object that is the best match to the query. However, this often results in high false-positive failure cases. For example, in a scene with no red cups and a blue cup, the method may respond with the location of the blue cup in response to the query "red cup".

For this reason, we locate objects in two stages. First, we find the best candidate image where the object may have been seen (Section 6). Then, we use an open-vocabulary object detector model such as OWL-v2 [81] to search that image for the queried object (Fig. 3). If we don't find the queried object, we assume that the object has either moved, or the response from the voxelmap or mLLM was inaccurate, and respond with "object not found". If OWL-v2 returns an object bounding box, we find the median pixel from the object mask and return its 3D location.

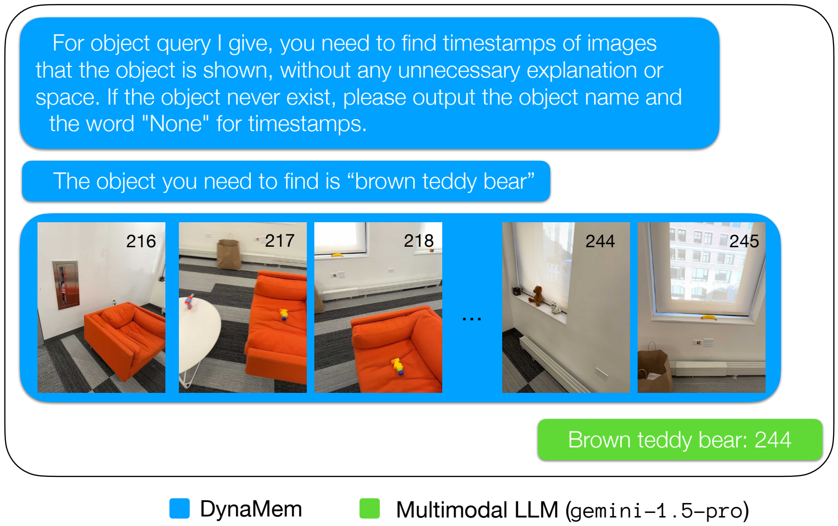

Figure 8: The prompting system for querying multimodal LLMs such as GPT-4o or Gemini-1.5 for the image index for an object query.

One important hyperparameter for this mLLM query is the maximum number of images included in the prompt. Longer context needs longer processing time and potentially includes outdated information, while short context might not include all information and thus will miss objects. We optimize the context by excluding completely outdated images: all images I with no voxel pointing to them are deleted. This filtering increases mLLM context utilization. We set Gemini as our base model and 60 as our query image limit since Gemini context can fit 60 images, which is twice as many as GPT-4o.

7 Robot Navigation and Exploration

To navigate in a real-world environment, robots use an obstacle map in conjunction with a navigation algorithm like A* in [24, 9]. We use a simple voxel-projection strategy to build an obstacle map. Due to the depth observation noise, we simply set a threshold for the ground (0.2m for our experiments), and project all the voxels above that z-threshold as the obstacles in our map. The voxels below the threshold are projected into the 2D obstacle map as navigable points. Finally, the points in the map that are not marked as either obstacle or navigable are marked as explorable points.

Exploration Primitives: Since our robot does not start with an environment map, it explores the environment with frontier based methods to build the map. We can further accelerate this process by providing exploration guidance. Based on the current status of the map, DynaMem provides an exploration value function to accelerate the exploration process both for building and updating the map.

We provide two value-based exploration maps: one time-based, and one semantic-similarity-based [82]. The time-based value map prioritizes the least-recently seen points. If the current time is T, and the last-seen time of voxel (x, y, z) is $t_{x,y,z}$, the temporal value map $\mathbb{V}_T$ is expressed as:

$$\mathbf{T}^*[x, y] = \max_z (T - t_{x,y,z})$$
$$\mathbb{V}_T[x, y] = \sigma\left(-\beta_T\left(\mathbf{T}^*[x, y] - \mu_T\right)\right)$$

where β_T, μ_T are hyper-parameters and σ is the sigmoid function. Similarly, if the VLM feature at voxel (x, y, z) is $f_{x,y,z}$, and the VLM feature for the language query is f_q, then the similarity-based value map $\mathbb{V}_S$ is be expressed as:

$$\mathbf{S}^*[x,y] = \max_z (f_q \cdot f_{x,y,z})$$
$$\mathbb{V}_S[x,y] = \sigma\left(-\beta_S\big(\mathbf{S}^*[x,y] - \mu_S\big)\right)$$

where once again β_S, μ_S are hyperparameters. We may also linearly combine $\mathbb{V}_T, \mathbb{V}_S$ to balance our exploration between last seen time and semantic similarity.

Finally, since the environment may be dynamic, we convert our navigation algorithm from open-loop to closed-loop. The robot, instead of executing the entire navigation plan generated by A*, stops after the first seven waypoints (approx. 0.7 to 1 meters). Then, the robot scans the environment, updates the map, and moves according to a new plan. The robot repeats these steps until its distance to the target is lower than a predefined threshold.

8 Ablations on an Offline Benchmark

Running real robot OVMM experiments can be expensive and time-consuming. So, we developed an offline benchmark called DynaBench to easily evaluate dynamic 3D visual grounding algorithms on dynamic environments and perform algorithmic ablations. The benchmark isolates the query-response part of the dynamic semantic memory without robot navigation, exploration, and manipulation.

8.0.1 Data Collection

In the real world, the robot collects its own map-building data by exploring the environments. Following this, we collect the robot's runtime sensor data from three environments. To further enrich our benchmark, we simulate this process by taking posed RGB-D images on an iPhone Pro in six more environments. In all cases, we emulate environment dynamics by moving objects and obstacle locations in three successive rounds.

8.0.2 Data Labelling and Evaluation

We manually annotate queries and responses in the dataset. Each query has an associated natural language label q, object location $\vec{X} = (x, y, z)$, and an object radius ϵ. Since the environment is dynamic, each query also has an associated time t. For evaluation, at time t (i.e. after the memory algorithm has observed all the input data points with timestamp $< t$), we query the model with q. If it predicts an object location $\vec{X'} = (x', y', z')$, it's a success if $||X - X'||_2 \leq \epsilon$ and a failure otherwise. Since the robot may also encounter queries for objects it has not observed yet, we emulate negative queries by adding queries for objects (a) that have not been observed yet, or (b) that have been observed but were subsequently removed. For both of these query types, the model must respond with *not found*; otherwise it's counted as a failure.

8.0.3 Evaluation Results

Using our offline benchmark, we ablate design decisions of DynaMem as discussed in Section 2. Among these design decisions, the primary are: using feature embedding-based vs. mLLM-QA based language grounding, ablating components such as point removal or abstentiation from the algorithm, and trying different mLLMs. Due to API costs, we only evaluate Gemini models on the benchmark. We present our results in Table 1.

We see that performance of VLM-features and mLLM-QA follows the same order in the real world in the benchmark, corroborating the benchmark design. The best design choices are to both add and remove points, and to cross check with OWL-v2 on top of similarity thresholding for VLM-feature based grounding. For mLLM-QA based grounding, Gemini Pro outperforms Gemini Flash, and voxelmap based image filtering benefits the method.

Table 1: Ablating the design choices for our query methods for DynaMem on the offline DynaBench benchmark. We also present results from five human participants to ground the performances.

Query type	Variant	Success rate
Human	(average over five participants)	**81.9%**
VLM-feature	default (adding and removing points)	**70.6%**
	only adding points	67.8%
	no OWL-v2 cross-check	59.2%
	no similarity thresholding	66.8%
mLLM-QA	default (Gemini Pro 1.5)	**67.3%**
	Gemini Pro 1.5, no voxelmap filtering	66.8%
	Gemini Flash 1.5	63.5%

9 Experiment Setup

We evaluate DynaMem and its impact on open-vocabulary mobile manipulation in three real-world dynamic environments (Fig. 5). In each environment, we set up multiple objects as potential manipulation targets, change the environment in three rounds, and execute 10 pick-and-drop queries over the rounds We use the Hello Stretch SE3 as our mobile robot platform, and use its head-mounted Intel RealSense D435 RGB-D camera to collect the input data.

To build a complete pick-and-drop system around DynaMem, we follow the system architecture in OK-Robot [9]. In particular, we use the AnyGrasp [83] based open-vocabulary grasp system and use the heuristic based dropping system. However, we use DynaMem's exploration primitives let the robot build the map of the environment and allow the robot to explore when an object is not found in the memory.