A Robust Framework for Open-Vocabulary Multi-Object Search with Visual-Language Understanding

Qianwei Wang*, Yifan Xu*, Vineet Kamat, and Carol Menassa

I. Introduction

Multi-Object Search (MOS) is a crucial task in robotics [1]. Consider a scenario where in a workplace setting, a robot may need to retrieve multiple objects to complete a task, such as gathering necessary documents, tools, or equipment for an assembly process. Similarly, in household environments, a robot making a coffee or preparing a hamburger must locate various ingredients and kitchen tools before assembling the final products.

In object search problems, one significant challenge is the uncertainty in observations [2].Error detection and recovery become especially crucial as perception reliability decreases with environmental complexity. Consequently, in MOS tasks—where multiple objects must be tracked, and the environment explored in depth—it is vital that agents continuously plan and reason about alternative actions to gather additional viewpoints and compensate for any missed detections.

Existing approaches to MOS can be broadly categorized into probabilistic planning, Deep Reinforcement Learning (DRL), and foundation model–based methods. Probabilistic planning methods [3]–[5], often formulated as Partially Observable Markov Decision Processes (POMDPs), manage uncertainty in object locations and perception by maintaining belief states and planning under partial observability. While effective for tracking multiple objects and enabling thorough exploration, these methods suffer from significant computational inefficiencies when scaling to large environments due to the complexity of belief state updates over extended planning horizons. DRL-based approaches [6]–[9], in contrast, train robots through repeated interactions with the environment to develop exploration strategies. Although capable of learning effective search policies, they often struggle with inefficient exploration and poor generalizability, particularly in novel environments. More recently, foundation model–based approaches, which encompass Large Language Models (LLMs) [10]–[13] or Vision-Language models (VLMs) [14], [15], have improved Single-Object Search (SOS) by leveraging object–scene and object–object relationships for more efficient target localization. However, existing multi-object search frameworks [16] incorporating foundation models often fail to address the challenges of observation uncertainty and complex environments, limiting their ability to efficiently navigate complex, partially observed spaces.

To address these limitations, we propose a novel framework that integrates VLMs, frontier-based exploration, and POMDP-based planning. Our approach utilizes VLMs to construct a multi-layer value map for different target objects. Instead of treating this map as static, we introduce a Bayesian-inspired decay mechanism, where the value of regions decreases over time if the detector repeatedly fails to locate objects. We then apply DBSCAN [17] clustering to transform the value map into belief representations and candidate points for POMDP-based action selection, effectively incorporating VLM-derived information into the POMDP formulation. Finally, we incorporate reward design for frontier-based exploration and solve the POMDP using Partially Observable Upper Confidence Trees (POUCT) [18].

Our main contributions are as follows:

- **A unified multi-object search framework:** We propose an open-vocabulary multi-object search framework that integrates VLMs, frontier-based exploration, and POMDP-based planning to achieve efficient and robust multi-object search.
- **VLM-guided POMDP:** We leverage VLMs to simplify POMDP belief updates and action selection, enabling effective uncertainty handling while mitigating the computational challenges of solving large-scale POMDPs.

II. Methodology

A. Problem Formulation

The MOS problem requires a mobile robot to search for a set of K static target objects in an unknown environment. The robot's state at time t is $\mathbf{x}_r(t) = (x, y, \phi) \in \mathbb{R}^3$, where (x, y) is its position and ϕ its orientation. The environment contains L static objects $\mathcal{O}_{\text{env}} = \{o_{s1}, o_{s2}, \ldots, o_{sL}\}$, among which the target objects $\mathcal{O}_{\text{tgt}} = \{o_{t1}, \ldots, o_{tK}\} \subseteq \mathcal{O}_{\text{env}}$ are to be located at unknown positions $\mathbf{x}_{tj}$. The objective of MOS is to locate all objects in $\mathcal{O}_{\text{tgt}}$ while minimizing the cumulative travel distance $d = \int_0^T \|\dot{\mathbf{x}}_r(t)\| dt$, where T is the total search time.

B. Mapping

1) Multi-Layer Value Map and Object Map: Following previous works VLFM [15] and Finder [16], we employ a pre-trained BLIP-2 [19] vision-language model to compute cosine similarity scores between the robot's current RGB observation and text prompts corresponding to each target object. At each time step, a cone-shaped confidence mask is

* Main Contribution

Funding for V. Kamat, C. Menassa and Y. Xu was provided by NSF Award No. 2124857.

The authors are with the University of Michigan, Ann Arbor, MI 48109, USA. {qweiw, yfx, vkamat, menassa}@umich.edu

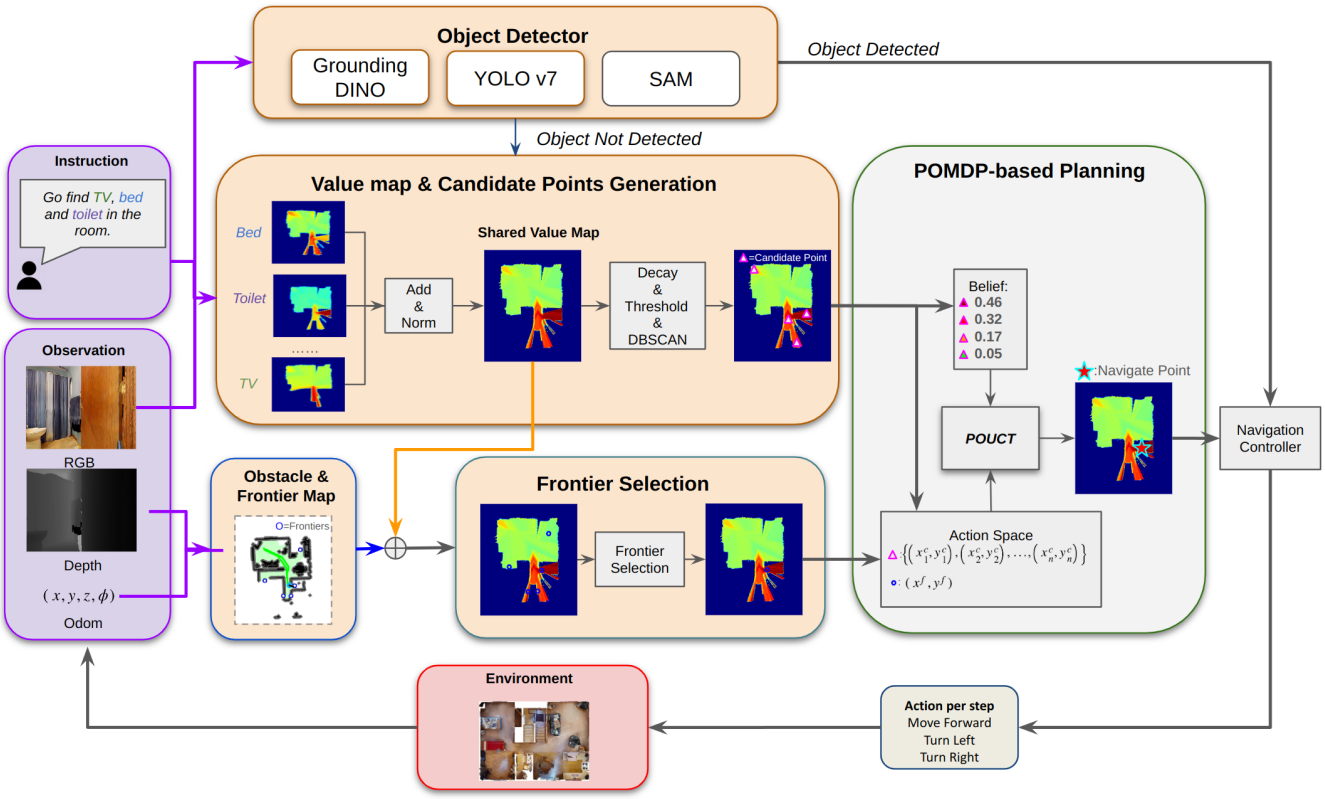

Fig. 1: Our framework consists of a *mapping module*, a *planning module*, and a *navigation controller*. The mapping module processes RGB-D inputs and textual prompts to generate an *object-value map*, which integrates detected objects and estimated potential object locations. If a target object is found in the object map, the robot navigates directly to it; otherwise, it relies on the value map to guide exploration. Additionally, the module constructs an obstacle map for navigation constraints and a frontier map to identify unexplored areas. The planning module maintains a belief representation of object locations and employs the POUCT algorithm to simulate and evaluate action sequences, selecting the one with the highest expected reward for execution. The *navigation controller* receives a target location from the planning module and outputs discrete movement commands (*move forward, turn left, turn right*) to guide the robot toward its destination.

generated to represent the camera's field of view (FOV). The confidence of each pixel is computed as:

$$c(i,j) = \cos^2\left(\frac{\theta}{\theta_{\text{FOV}}/2} \times \frac{\pi}{2}\right) \tag{1}$$

where θ is the angle between the pixel and the optical axis, and θ_{FOV} is the camera's horizontal field of view.

To handle overlapping observations over time, we apply a weighted averaging update for the value map:

$$v_{i,j}^{\text{new}} = \frac{c_{i,j}^{\text{curr}} v_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}} v_{i,j}^{\text{prev}}}{c_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}}} \tag{2}$$

where $v_{i,j}$ represents the value at pixel (i,j), and $c_{i,j}$ denotes the confidence score. The confidence update is computed as:

$$c_{i,j}^{\text{new}} = \frac{(c_{i,j}^{\text{curr}})^2 + (c_{i,j}^{\text{prev}})^2}{c_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}}} \tag{3}$$

which biases the update towards higher confidence values.

After obtaining the value maps for different target objects, we normalize and aggregate them to form a shared value representation. This step follows a similar approach to previous works [3]–[5] that assume independence across objects, where the joint belief is computed as the product of individual beliefs.

For object detection, we incorporate Grounding DINO [20], YOLOv7 [21], and Segment Anything Model (SAM) [22]. These models enable us to detect, segment, and store all identified objects in an object map throughout the search process.

2) Obstacle Map and Frontier Map: We utilize depth and odometry data to construct a top-down 2D obstacle map, representing regions that the robot has identified as non-traversable. Based on this obstacle map, we determine boundaries between explored and unexplored areas and extract midpoints along these boundaries as potential frontier waypoints. These frontiers guide exploration in unknown environments.

C. Planning

We model the planning process within an Object-Oriented POMDP (OO-POMDP) framework [3]. In our formulation, the state and observation spaces are decomposed with respect to a single *target object*, and multi-object search is achieved via our multi-layer value map using *add* and *norm* operations. Our approach introduces two key modifications: a novel action space formulation and an alternative belief

Algorithm 1 POUCT-based Planning $(\mathcal{P}, b_t, d) \rightarrow a$

Require: $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma \rangle$
where
$\mathcal{A} = MoveTo\{(x_1^c, y_1^c), \ldots, (x_n^c, y_n^c), (x^f, y^f)\}$
$b_t = \{((x_1^c, y_1^c) : p_1), \ldots, ((x_n^c, y_n^c) : p_n)\}$
Ensure: a: An action in the $\mathcal{A}$ of $\mathcal{P}$

1: **procedure** PLAN(b_t)
2: $\mathcal{G} \leftarrow$ GenerativeFunction($\mathcal{P}$)
3: $Q(b_t, a) \leftarrow$ POUCT($\mathcal{G}, h_t$)
4: **return** a
5: **end procedure**

update mechanism for real-world execution.

1) Update the Action Space and Belief with Candidate Points and Selected Frontier: Let the raw value map be denoted as $v(x, y)$ over the spatial domain. First, the frontier with the highest value is selected from $v(x, y)$ as a representative exploratory point (x^f, y^f). Then, to refine the value distribution for targeted search, we apply a decay function:

$$v'(x, y) = \frac{1}{1 + \exp\left(\frac{u(x,y) - \tau}{\kappa}\right)}, \tag{4}$$

where $u(x, y)$ is the update count at location (x, y), and τ and κ are constants controlling the decay rate. After thresholding $v'(x, y)$ to extract high-value regions, we employ DBSCAN [17] clustering to yield a set of candidate points (x_i^c, y_i^c):

$$\mathcal{C} = \{(x_i^c, y_i^c) \mid i = 1, \ldots, n\}.$$

The action space is then defined as:

$$\mathcal{A} = \mathcal{A}_{\text{cand}} \cup \mathcal{A}_{\text{frontier}},$$

with

$$\mathcal{A}_{\text{cand}} = \{\texttt{MoveTo}((x_i^c, y_i^c)) \mid i = 1, \ldots, n\},$$
$$\mathcal{A}_{\text{frontier}} = \{\texttt{MoveTo}((x^f, y^f))\}.$$

This formulation balances targeted search (via candidate points) with exploratory actions (via the frontier).

The belief over the target object's location is represented as a discrete distribution over candidate points:

$$b_t = \{\, ((x_i^c, y_i^c) : p_i) \mid i = 1, \ldots, n \,\},$$

where

$$p_i = \frac{v(x_i^c, y_i^c)}{\sum_{j=1}^{n} v'(x_j^c, y_j^c)}.$$

Here, p_i denotes the probability associated with the candidate point (x_i^c, y_i^c).

During the POMDP solution process (i.e., in the simulation phase), we still update the belief using the standard Bayesian rule:

$$b_{t+1}(s') = \eta \Pr(o \mid s', a) \sum_{s \in \mathcal{S}} \Pr(s' \mid s, a)\, b_t(s),$$

with normalization constant η. However, in real-world execution, after receiving an observation, we approximate the belief update using a *decayed value map*. This approach circumvents explicit reliance on the observation model while retaining adaptability, and thereby offers a more efficient method for adjusting the belief in practical settings.

The rationale for this approach is that the value map itself already encodes a highly reliable representation of the probability that the object is located at each candidate point. After incorporating detector information into the decayed value map, there is no longer a need to perform extensive POMDP simulations—where possible observations are hypothesized and used in Bayesian updates—to adjust the belief after receiving the real observation. Instead, the decayed value map directly reflects the updated likelihood of the object's location in a simpler, yet still effective, manner.

2) POMDP Components: We model our planning problem as a POMDP defined by the tuple

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma \rangle.$$

The components are defined as follows:

- **State Space ($\mathcal{S}$)**: An environment state s is represented as a combination of the robot's state s_r and the target object's state s_t:

  $$s = \{s_r, s_t\}.$$

  The robot state is defined as $s_r = (x_r, y_r)$ with $(x_r, y_r) \in \mathbb{R}^2$, representing its 2D position, while the target object's state is given by $s_t = (x_t, y_t)$ with $(x_t, y_t) \in \mathbb{R}^2$, indicating its position in the environment.
- **Observation Space ($\mathcal{O}$)**: The robot obtains an observation o from its RGB-D camera, modeled as a binary indicator of whether detected or not:

  $$o \in \{0, 1\}.$$

- **Action Space ($\mathcal{A}$)**: Actions are defined as movement commands toward a goal location:

  $$a = \texttt{MoveTo}(g)$$

  where g is chosen from candidate points extracted from the value map or the selected frontier.
- **Transition Model ($T(s, a, s')$)**: The target object is static (i.e., $s'_t = s_t$), and the robot state transitions deterministically:

  $$s'_r = f(s_r, a),$$

  where f updates the robot's 2D position based on the selected action.
- **Observation Function (O)**: The detection probability is defined as:

  $$\Pr(o = 1 \mid s', a) = \begin{cases} 1, \text{if } d(s'_r, s_t) \le \delta, \\ \exp\Big(-\beta \left(d(s'_r, s_t) - \delta\right)\Big), \\ \quad \text{if } d(s'_r, s_t) > \delta. \end{cases} \tag{5}$$

  where $d(s'_r, s_t)$ is the Euclidean distance between the

Methods	SR↑	MSPL↑
Random Walk	0.0%	0.0
VLFM	12.5%	0.075
Finder	28.3%	0.198
Ours	**55.0%**	**0.497**

TABLE I: Overall performance comparison of Random Walk, VLFM [15], Finder [16], and Our Method on MOS, including Success Rate (SR) and Multi-Object Success weighted by normalized inverse Path Length (MSPL).

robot and the target, δ is the detection threshold, and $\beta > 0$ controls the decay rate.

- **Reward Function** ($R(s, a)$): The reward function is defined as:

$$\begin{aligned} R(s,a) = & -\lambda_{\text{move}}\, d(s_r, s_r') \\ & + \lambda_{\text{frontier}}\, \mathbb{I}(a \in \mathcal{A}_{\text{frontier}}) \\ & + \lambda_{\text{target}}\, \mathbb{I}\Big(d(s_r', s_t) \le \delta\Big), \end{aligned} \tag{6}$$

  where $d(s_r, s_r')$ is the distance traveled by the robot, $\mathbb{I}(\cdot)$ is the indicator function, and $\lambda_{\text{move}}, \lambda_{\text{frontier}}, \lambda_{\text{target}} \ge 0$ are weight parameters balancing movement cost, exploratory incentive, and target proximity reward.
- **Discount Factor** (γ): A constant $\gamma \in (0, 1)$ is used to balance immediate and future rewards.

To solve the formulated POMDP efficiently, we employ the partially Observable Upper Confidence Trees (POUCT) algorithm [18]—a Monte Carlo Tree Search-based method. As illustrated in Algorithm 1, after the agent executes a step in the real environment, it obtains an updated map from which new candidate points and a frontier are extracted. These are then used to update the action space $\mathcal{A}$ and the belief b_t.

Then, it is straightforward to define a generative function

$$\mathcal{G}(s, a) \to (s', o, r),$$

using its transition, observation, and reward functions. Leveraging this generative function, POUCT builds a search tree to simulate transitions, observations, and rewards, thereby planning and selecting the next optimal action.

III. Experiment

A. Setup

We conduct our simulation experiments using the HM3D dataset's validation split. We select five scenes and construct a total of 120 episodes, all of which involve searching for *three objects*. At the beginning of each episode, the robot is initialized at a random location within the environment and provided with a list of target objects. The episode progresses as follows: each time the robot calls *stop*, it indicates that it has found an object. If the distance between the robot and the nearest target object is less than 1 m at this moment, the object is considered successfully found. The robot then proceeds to search for the remaining objects. The episode terminates when either all target objects have been found or the robot reaches the maximum step limit of 500.

B. Evaluation Metrics

To evaluate performance, we use three key metrics:

- **Success Rate (SR)**: The percentage of episodes in which the robot successfully finds all target objects.
- **Multi-Object Success weighted by normalized inverse Path Length (MSPL)**: Based on the SPL metric, MSPL is calculated as:

$$MSPL = \frac{1}{N} \sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)} \tag{7}$$

  where N denotes the total number of episodes, S_i is a binary indicator of success for episode i, l_i represents the optimal shortest path length from the start location to all target objects, and p_i denotes the actual path length traversed by the robot.

C. Comparison Baselines

We compare our approach against the following baseline methods:

- **Random Walk (Lower Bound)**: A naive strategy where the robot moves randomly in the environment.
- **VLFM [15](SOTA for SOS)**: This method combines a VLM-generated value map with frontier-based exploration, selecting the best frontier at each step. In our experiments, the MOS task is decomposed into a sequence of SOS tasks executed independently.
- **Finder [16] (SOTA for MOS)**: Built upon VLFM, this method utilizes value maps to compute the *scene-to-object* score and incorporates an additional *object-to-object* score to select the optimal frontier.

D. Results and Analysis

The experimental results are summarized in Table I. VLFM achieves only a 12.5% success rate (SR) with an MSPL of 0.075, which reflects its limitation of tracking only one object at a time. Finder improves on this by employing a multi-layer value map to simultaneously track multiple objects, resulting in an enhanced SR of 28.3% and an MSPL of 0.198. Our method further builds on Finder's strengths by integrating POMDP-based planning into the framework. This additional planning mechanism enables adaptive exploration and more informed decision-making, leading to a significant performance boost with an SR of 55.0% and an MSPL of 0.497. Thus, our method nearly doubles the success rate and more than doubles the MSPL compared to Finder, demonstrating its superior accuracy and efficiency in multi-object search tasks.

IV. Conclusion

In this work, we integrate VLMs, frontier-based exploration, and POMDP to develop a robust and efficient multi-object search framework. Our current approach is limited to 2D environments with a fixed camera viewpoint and simple actions. Future work will extend to 3D spaces, introduce complex actions like moving occlusions, and enable dynamic camera adjustments for enhanced adaptability and effectiveness.

REFERENCES

[1] K. Zheng, "Generalized object search," 2023. [Online]. Available: https://arxiv.org/abs/2301.10121

[2] S. Yenamandra, A. Ramachandran, M. Khanna, *et al.*, "Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 homerobot open vocabulary mobile manipulation challenge," 2024. [Online]. Available: https://arxiv.org/abs/2407.06939

[3] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, L. L. Wong, and S. Tellex, "Multi-object search using object-oriented pomdps," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7194–7200.

[4] K. Zheng, A. Paul, and S. Tellex, "A system for generalized 3d multi-object search," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1638–1644.

[5] K. Zheng, Y. Sung, G. Konidaris, and S. Tellex, "Multi-resolution pomdp planning for multi-object search in 3d," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2022–2029.

[6] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva, "Multi-on: Benchmarking semantic map memory using multi-object navigation," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[7] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces," in *The International Symposium of Robotics Research*. Springer, 2022, pp. 52–66.

[8] A. Sadek, G. Bono, B. Chidlovskii, A. Baskurt, and C. Wolf, "Multi-object navigation in real environments using hybrid policies," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 4085–4091.

[9] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8549–8556, 2023.

[10] K. Zhou, K. Zheng, C. Pryor, Y. Shen, H. Jin, L. Getoor, and X. E. Wang, "Esc: Exploration with soft commonsense constraints for zero-shot object navigation," 2023. [Online]. Available: https://arxiv.org/abs/2301.13166

[11] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[12] V. S. Dorbala, J. F. Mullen, and D. Manocha, "Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4083–4090, 2024.

[13] B. Yu, H. Kasaei, and M. Cao, "L3mvn: Leveraging large language models for visual target navigation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 3554–3560.

[14] Y. Kuang, H. Lin, and M. Jiang, "Openfmnav: Towards open-set zero-shot object navigation via vision-language foundation models," 2024. [Online]. Available: https://arxiv.org/abs/2402.10670

[15] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[16] D. Choi, A. Fung, H. Wang, and A. H. Tan, "Find everything: A general vision language model approach to multi-object search," 2024. [Online]. Available: https://arxiv.org/abs/2410.00388

[17] D. Deng, "Dbscan clustering algorithm based on density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, 2020, pp. 949–953.

[18] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23. Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2010/file/edfbe1afcf9246bb0d40eb4d8027d90f-Paper.pdf

[19] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 19 730–19 742. [Online]. Available: https://proceedings.mlr.press/v202/li23q.html

[20] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[21] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475.

[22] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.