TANGO: Training-free Embodied AI Agents for Open-world Tasks

Supplementary Material

1. Additional Experiments

1.1. Embodied Question Answering

We conducted preliminary experiments with TANGO on Embodied Question Answering (EQA), utilizing an older dataset for the task [1]. In EQA, the agent is queried with a natural language question and it must navigate to the target location described in the query and answer accordingly. In the case of the MP3D-EQA dataset [1], this task is regarded as a classification task aimed at determining the most suitable answer from a set of pre-defined possibilities (i.e., class labels). For this dataset, we have: 24 colors, 18 rooms, and 25 objects with perfect correspondence between train and test set. However, TANGO utilizes a different approach, in which the agent is free to provide an answer independently of predefined target categories, because it is not trained specifically for the task. Nevertheless, Table 1 shows that our model yields comparable results against trained methods both in Answer Accuracy (QA) and Distance to goal (d_T) (row 5), while requiring no training. T_i is related to 10, 30, 50, random steps away from the target object. In particular, Neural Modular Controller (NMC) methods [2] employ specific navigation policies trained with RL (rows 3-4). Furthermore, these models are trained from a predefined list of possible answers. In contrast, our model can provide answers using natural language. For example, in episodes where the correct color is described as *"off-white"*, while our agent's output is *"white"*, these instances yield an Answer Accuracy of 0%, despite being similar answers. Therefore, we evaluated our model with an "answer constraint" mechanism in which similar answers were grouped into the same answer. Results show that our method provides the best results among 30 and 50 steps away, surpassing by 4% the best performing method, confirming that TANGO is able to navigate towards the target object even when it is far from it in the initial position. Furthermore, in the "answered constraint" scenario, the primary reasons for failure can be attributed to misclassification or the failure of the object detector to detect the object despite its presence (i.e. failure of the "detect" module).

1.1.1 EQA Datasets Comparison

We compared the two primary datasets used in the EQA task within embodied environments [1, 4], aiming to discuss a key limitation of these datasets to approach the given task. As shown in Figure 1, the more recent OpenEQA dataset (row 2) lacks certain important features present in the older MP3D-EQA dataset, such as target localization, which is useful for calculating the agent's path efficiency, as well as a dedicated training set. Conversely, the older dataset lacks open-vocabulary questions and the novel LLM-scoring metric. Therefore, we advocate for further enhancement of these datasets to create a comprehensive and well-designed resource for the EQA task, where all of these characteristics are present. Moreover, it is reasonable to hypothesize that a model excelling in this task would likely perform well in other purely navigational tasks (e.g., ObjNav, Lifelong Navigation) without additional training, given the generalization required to interpret the input question, navigate the environment, and provide an answer.

	Modalities			A(ctive)	Target Localization	Contains Training set	Open Vocab	LLM Scoring
	RGB	Depth	Camera					
MP3D-EQA	✓	✓	✓	✓	✓	✓	✗	✗
OpenEQA	✓	✓	✓	✓	✗	✗	✓	✓

Figure 1. **EQA Datasets comparison.** Dataset comparison between MP3D-EQA [1] and OpenEQA [4].

1.2. Multi ObjectGoal Navigation

Given the zero-shot nature and the need for broad applicability across diverse tasks, our approach requires the use of open-set object detectors. We recognize that fine-tuning these detectors to specific tasks or datasets could significantly enhance performance. However, such fine-tuning would compromise the system's generality. To further demonstrate the robustness of our approach, we fine-tuned the YOLO object detector [6] on a custom dataset generated by placing the target objects (colored cylinders) from the MultiON dataset [7] into HM3D and MP3D environments. Following this fine-tuning, we evaluated our approach using the fine-tuned detector in episodes where each consisted of three sequential targets. The results, as shown in Table 2, illustrate the benefits of this fine-tuning strategy.

2. Exploration Policy using Memory targets

As described in Section 3, we extend the exploration policy from [8] by incorporating a memory mechanism based on a stored feature vector map. At each step, the current RGB observation is processed through a vision-language model (BLIP2 in our implementation [3]), updating the current view angle (a triangular-shaped region of pixels on the map) with feature vectors for each pixel. If the navigation target changes, the current value map is updated with a new value map specific to the new target. This new map is calculated

Method	Trained	Navigation ($d_T \downarrow$)				QA (Top-1 ↑)			
		T_{10}	T_{30}	T_{50}	Random	T_{10}	T_{30}	T_{50}	Random
PACMAN (BC) [1]	✓	1.19	4.25	8.12	N.A.	48	40	40	N.A.
PACMAN (BC + RF) [1]	✓	**1.05**	4.22	8.13	N.A.	50	42	41	N.A.
NMC (BC) [2]	✓	1.44	4.14	8.43	N.A.	43	41	39	N.A.
NMC (BC + A3C) [2]	✓	1.06	**3.72**	7.94	N.A.	**53**	**46**	44	N.A.
TANGO (ours)	✗	3.43	4.50	**5.26**	**7.28**	42	40	38	**37**
TANGO (ours) +answer constraint	✗	3.43	4.50	**5.26**	**7.28**	52	**50**	**48**	**45**

Table 1. **MP3D-EQA results.** Comparison of Navigation and QA performance across methods.

TANGO	Finetune	SR(↑)	PRG(↑)	SPL(↑)	PPL(↑)
Owl-ViT2	✗	24	43	10	19
YOLOv8m	✓	**45** (+21%)	**65** (+22%)	**20** (+10%)	**28** (+9%)

Table 2. Object detector fine-tuning ablation on MultiON dataset.

by applying cosine similarity between the text or image features of the new target (obtained from the vision-language model) and each pixel's feature vector in the map. Figure 2 illustrates the following:

$$cos\left(\text{map}_{(i,j)}^{val}\right) = \frac{\vec{\text{map}}_{(i,j)}^{feat} \cdot \vec{\text{E}}^{target}}{||\vec{\text{map}}_{(i,j)}^{feat}|| \cdot ||\vec{\text{E}}^{target}||} \forall (i,j) \in \text{map}^{val} \quad (1)$$

where $\vec{\text{E}}$ is the embedding vector of the new target (either specified through text or image), map^{feat} is the feature map storing the vector embeddings for each pixel, and map^{val} is the value map used during exploration. Frontiers are retrieved from an obstacle map calculated on the fly [8].

After obtaining the new value map, we assess whether the agent may have already encountered the target object. We sample the highest value in the map, and if it exceeds a defined threshold, we consider the target "remembered" and navigate directly to it. If the target object is not found at the expected location, exploration resumes following the standard policy from [8]. Overall, this approach enables more efficient navigation as the model continues to explore, and future work aimed at exploring clustering of high-value regions could be promising.

2.1. Ablation Studies

To evaluate our memory mechanism, we conducted preliminary experiments on multi-target object-goal navigation to identify the optimal threshold, which was then used in the experiments detailed in Section 4. Table 3 presents the results on the Multi-Object Goal Navigation dataset [7]. In this task, similar to ObjectNav, the goal is to navigate to target objects. However, in contrast to ObjectNav, a single

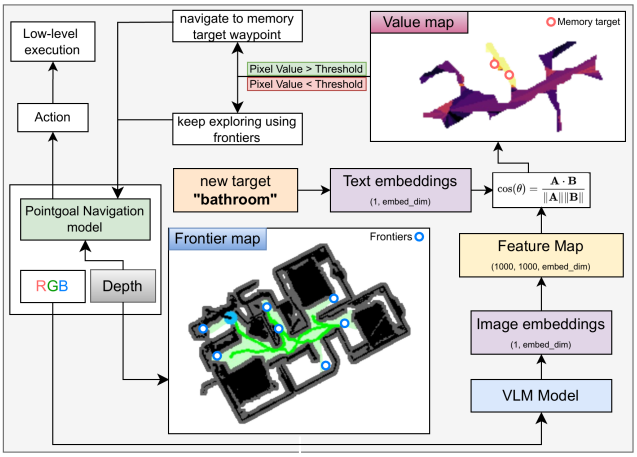

Figure 2. **Exploration Policy.** Illustration of the implemented memory mechanisms in TANGO, when a new sequential target is found.

episode consists of multiple sequential targets. We evaluated agent performance using three different target types, which are cylindrical objects distinguished by color. The agent is required to locate all targets within a maximum of 2500 steps. If the agent incorrectly calls the "Found" action on a target, the entire episode is deemed incorrect. Hence, serving as an effective testbed for evaluating our memory strategy performance. The ablation study investigates the effect of the memory threshold on the agent's ability to "remember" previously encountered objects and navigate to them. Since the value map is normalized between 0 and 1, the memory threshold can take any float value within this range. A threshold of 1 indicates that no memory is used, as it is too high, while a threshold of 0 means that memory is always utilized, potentially leading to incorrect memory-based target selections. The results show that the optimal memory threshold value is 0.4, as it yields the best overall performance. We evaluate the baseline of our approach without memory (row 1) and observe that incorpo-

rating memory significantly benefits the agent, improving the success rate by $+5\%$ and path efficiency by $+2\%$ (row 4). Additionally, the Progress metric, which tracks the percentage of targets found within a single episode, increases by 4%. Overall, the memory mechanism helps enhance the agent's performance. Notably, thresholds are highly dependent on the normalization applied during value map calculation. For instance, as shown in Table 3, the results peak around threshold values of 0.3 and 0.4. As highlighted in Section 5, further exploration of diverse sampling strategies for high-value region pixels is encouraged.

Memory	Threshold	SR↑	Progress↑	SPL↑	PPL↑
✗	✗	19	39	8	17
✓	0.2	21	39	10	18
✓	0.3	23	42	10	**20**
✓	0.4	**24**	**43**	**10**	19
✓	0.5	19	40	8	17

Table 3. **TANGO Ablation Study**. Results on the MultiON dataset [7], with 3 sequential targets.

2.1.1 Object Detector

A key challenge with open-set object detectors is the high rate of false positives, particularly for objects in long-tail or less common categories. To tackle this issue, we introduced a verification step, detailed in Section 3. This step employs an open-set classifier, such as CLIP, to determine whether the image within the detected bounding box accurately matches the predicted object category. The verification is implemented as a simple confidence score threshold given the two categories: the target category and "other". The effectiveness of this method is shown in Table 4.

TANGO	CLS	SR(↑)	SPL(↑)	DTG(↓)
Owl-ViT2	*w/o*	26	13	4.1
Owl-ViT2	*w*	**28** (+2%)	**15** (+2%)	**3.8** (+7%)

Table 4. Object detection, false positives ablation. The detection are fed to a *CLS* model for category checking.

3. Failure analysis

As previously outlined, we extracted a significant subsample for the task and manually classified the instances where the model failed. Here we complete the analysis adding the failures in the purely navigational case (see the bottom of Figure 3). For this last case, the majority of failures stem from issues in the navigation and detection modules (9.8%), rather than planning errors by the Large Language Model

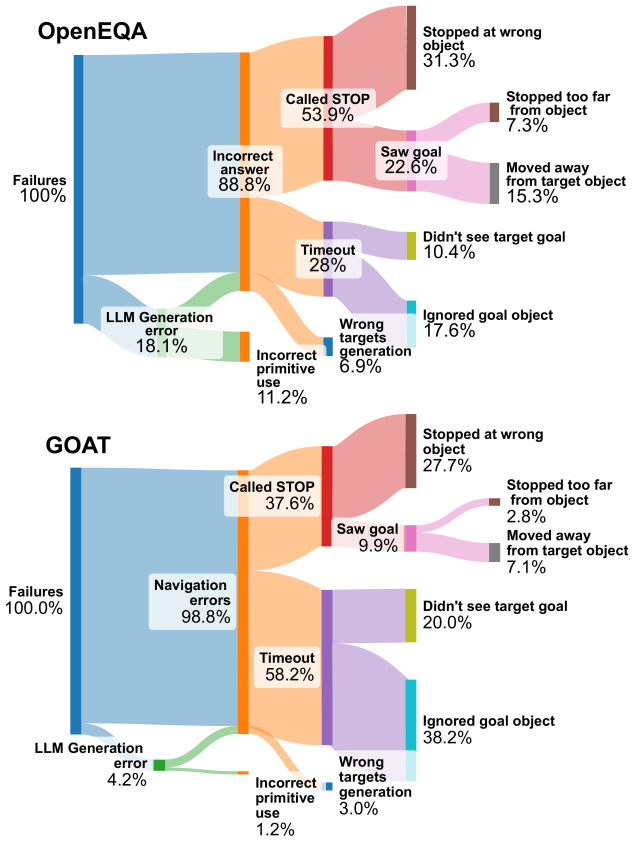

Figure 3. **Failure analysis.** TANGO failure analysis. (top) OpenEQA failures, (bottom) Goat-Bench, Life Long Multimodal navigation task failures.

(LLM) (4.2%). This discrepancy is due to the fact that navigation tasks inherently involve simpler prompts, such as *"navigate to the chair in the kitchen"*, which clearly specify two distinct targets. These are easier for the LLM to interpret and sequence effectively. Furthermore, we observe that the "Timeout" category is more prevalent in Navigation tasks compared to EQA tasks. This is particularly evident in Open-set ObjNav, where targets are often highly ambiguous, making it difficult for the open-set object detector to identify them in simulated 3D environments. Notably, the "Ignored goal object" category accounts for 38% of failures, significantly higher than the 17.6% observed in EQA tasks. In contrast, the "Didn't see target" category remains consistent across both tasks, accounting for approximately 20% of failures—half the size of the "Ignored goal object" category. This consistency indicates that the navigation policy associated with this category performs reliably for these tasks. Concerning navigation tasks, we also identified instances where the definition of success threshold distance to the goal appeared overly stringent. Some episodes witnessed the agent halting within 1 meter of the object with

the object in view. However, these instances were deemed failures due to the sparse sampling of viewpoints, indicating potential areas for enhancement in the evaluation protocol. Specifically, in the case of ObjNav, the same issue was also highlighted in [5].

4. LLM Prompts.

TANGO utilizes LLMs to parse input prompts and generate synthetic pseudocode for task completion. As described in the conclusions section, the LLM is provided with 15 in-context examples spanning various tasks and is tasked with independently composing the appropriate primitives to solve the given task. The example programs range from simple object-goal navigation (e.g. *"I've lost my laptop, where is it?"*) to multiple EQA question-program pairs (e.g. *"Can you tell me if I left the TV on?"*), with the latter being the most challenging to accurately transform into pseudocode, as illustrated in Section 5. The rationale for including diverse tasks is to encourage the LLM to generalize effectively and learn correct module compositions for specific problems. Therefore, it is the LLM's job to understand the current prompt and generate correct pseudocode to tackle the related task. In particular, in the case it is fed an image as input, it has to first extract the semantic object it represents. Figure 4 illustrates the initial prompt structure, including example-program pairs. Moreover, instructing the LLM to comment on its own code enhances explainability, which is particularly useful when the model outputs incorrect targets for the task, resulting in a failure.

Figure 5 illustrates an example of TANGO successfully transforming a target description into a sequence of ordered subtasks.

You are a smart assistant that answers questions by providing Pseudo-code programs.

I will show you examples of Pseudo-code in response to various questions. For each new question, only respond with the Pseudo-code program that directly answers the question. Only use functions you have seen in the examples provided. Comment your lines where needed.

Examples:

Question: {Question here}
Program: {Pseudo-code here}
. . .

. . .
Question: {Question here}
Program: {Pseudo-code here}

Now respond to the new question based on these examples. Only include the related Pseudo-code.

Question: What is to the right of the fridge?
Program:

Figure 4. **Initial prompt.** Initial prompt fed to the LLM to generate the new pseudo-code used in navigation.

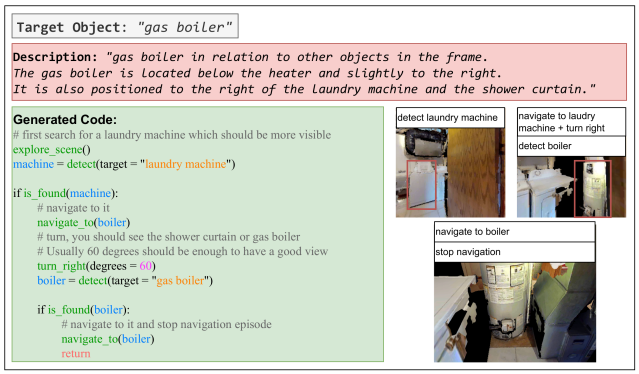

Figure 5. **GOAT subtask example**. The target is *gas boiler* and the agent has to follow the set of primitives generated by the LLM. Moreover, the comments on the code helps understand the LLM's thought process.

5. Dynamic Program Updating

We want to emphasize that continuously updating the program at each step—by feeding the LLM the current observation along with a history of what it has seen and the current program state—could greatly enhance the system by enabling dynamic adjustments as needed. However, this approach is quite expensive since it requires making an API call to the model at each step. A possible direction is to minimize the number of tokens fed to the LLM during this process. This potential direction for future work would necessitate rethinking the memory mechanism so that it can be queried directly by the LLM during its decision-making process. In other words, the module would need to be redefined based on the information provided by memory when the LLM "thinks" about the next possible step. This integration of memory and LLM reasoning could be a promising avenue for combining the strengths of semantic navigation with advanced language-based reasoning.

References

[1] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[2] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *Proc. of the International Conference on Robot Learning (CoRL)*, 2018. 1, 2

[3] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 1

[4] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio 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. 1

[5] Sonia Raychaudhuri, Tommaso Campari, Unnat Jain, Manolis Savva, and Angel X Chang. Mopa: Modular object navigation with pointgoal agents. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5763–5773, 2024. 4

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[7] Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *Advances in Neural Information Processing Systems*, 33:9700–9712, 2020. 1, 2, 3

[8] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette 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. 1, 2