Personalized Embodied Navigation for Portable Object Finding

Vishnu Sashank Dorbala*[1], Bhrij Patel*[1], Amrit Singh Bedi[2] and Dinesh Manocha[1]
[1]University of Maryland, College Park
[2]University of Central Florida

***Abstract*— Embodied navigation methods commonly operate in *static* environments with *stationary* objects. In this work, we present approaches for tackling navigation in dynamic scenarios with *non-stationary* targets. In an indoor environment, we assume that these objects are everyday portable items moved by human intervention. We therefore formalize the problem as a personalized habit learning problem. To learn these habits, we introduce two Transit-Aware Planning (TAP) approaches that enrich embodied navigation policies with object path information. TAP improves performance in portable object finding by rewarding agents that learn to *synchronize* their routes with target *routes*. TAPs are evaluated on Dynamic Object Maps (DOMs), a dynamic variant of node-attributed topological graphs with structured object transitions. DOMs mimic human habits to simulate realistic object routes on a graph. We test TAP agents both in simulation as well as the real-world. In the MP3D simulator, TAP improves the success of a vanilla agent by 21.1% in finding non-stationary targets, while also generalizing better from static environments by 44.5% when measured by Relative Change in Success. In the real-world, we note a similar 18.3% increase on average, in multiple transit scenarios. We present qualitative inferences of TAP-agents deployed in the real world, showing them to be especially better at providing personalized assistance by finding targets in positions that they are usually not expected to be in (a toothbrush in a workspace). We also provide details of our *real-to-sim* pipeline, which allows researchers to generate simulations of their own physical environments for TAP, aiming to foster research in this area.** [1]

I. Introduction

Embodied navigation involves an agent moving in unseen physical or virtual environments to carry out human-guided tasks such as locating objects [1]–[4], following instructions [5]–[8], or reaching a point in space [9], [10]. State-of-the-art methods in this field use Reinforcement Learning (RL) [10]–[12] and Large Language Model (LLM) based schemes [3], [13] to achieve superior task performance, and these policies also extend to multi-object finding [14], [15]. Usually, these agents operate with only local observations and not from additional sources like external cameras [4], [16].

However, a key feature of current embodied navigation paradigms [14], [15], [17], [18] is their reliance on static graphs, where objects within nodes remain stationary. This static assumption is limiting, as real-world environments are inherently dynamic, where users frequently move portable objects from place to place over long periods, such as shifting their phone or wallet between rooms throughout the day. Studies on robots in collaborative environments [19]–[21] suggest that human object placements follow structured patterns shaped by routines and habits, introducing a degree of entropy in object locations over time. Despite the importance of non-stationary environments, existing navigation methods fail to model temporal variability, limiting their deployment in real-world settings.

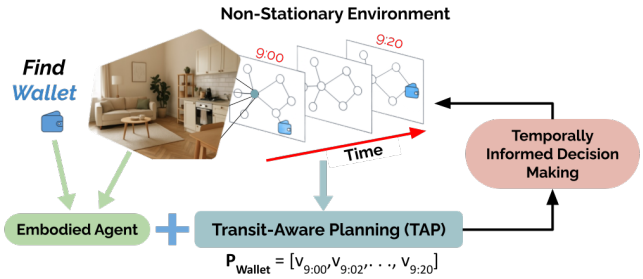

Fig. 1: **Transit-Aware Planning (TAP)**: We introduce TAP to tackle embodied navigation in non-stationary environments. In this figure, an embodied agent is tasked with finding a wallet that changes positions between 9:00 and 9:20. P_{wallet} represents the transit route of the wallet. TAP makes RL and LLM-based navigation agents "*transit-aware*" by augmenting their policies for temporally informed decision-making.

In the ObjectNav task [1], [17], for instance, SotA navigation approaches all involve finding *stationary* targets using RL [1], [10], [11], [22] and LLM-based zero-shot methods [13], [17], [23]. RL policies for multi-object settings [14], [15] have also been proposed, but these are also based on static environments. In contrast, navigating to non-stationary targets that move as the agent moves necessitates the temporal modelling of object transit to guide the agent.

For example, OVRL-V2 [11] uses CLIP text embeddings and ViT-processed RGB images, while DD-PPO [10] similarly relies on GPS readings and RGB-D sensors; both use only static scene observations. Their reward functions similarly assume target stationarity, with OVRL-V2 and DD-PPO penalizing absolute distance changes to the goal. Using these approaches directly in non-stationary environments shows poor adaptability, and this can be attributed to a *high-variance, noisy* signal hindering policy convergence.

For real-world deployment, these approaches must *generalize*, i.e., maintain performance in dynamic scenes with shifting targets. Prior embodied work describes generalizability in terms of measuring an agent's capacity to adapt to novel settings, but this usually refers to improved performance in *unseen* environments or unique synthesized guidance language [24]–[27]. Little work defines generalizability in the context of adapting to dynamic settings with non-stationary targets.

Main Results. To address these issues, we present *Transit-Aware Planning* (TAP) strategies that augment embodied

[1]Code and data for our benchmark will be made publicly available upon acceptance.

navigation policies with temporal "awareness" on target positions (Fig. 1). Rather than naively chasing target objects, TAP augmented agents learn to *intercept* target routes for better convergence. To evaluate TAP, we also introduce Dynamic Object Maps (DOMs), a dynamic variant of topological graphs incorporating *structured* object transitions. Figure 2 illustrates this concept. DOM node attributes change over time, i.e. target objects move from node to node at each timestep, while the edges remain fixed. A successful agent learns to exploit structured object paths in the environment, adapting itself to better find non-stationary targets over time. Our work seeks to redefine success from traditional embodied navigation tasks, by emphasizing not just **where** to navigate but also **when** to get there. Our contributions are summarized as follows:

Transit-Aware Planning (TAP): We propose TAP, a set of novel strategies that equip embodied agents with *non-stationary* target finding capabilities. TAP aims to adapt standard RL [10] and LLM [17]- based navigation policies, initially developed for static target finding, to non-stationary targets via TAP-RL and TAP-LLM.

Dynamic Object Maps (DOMs): We propose a *structured* formulation for object transitions on static topological graphs, transforming them into DOMs. Inspired by human habits, we define three **object transition scenarios** to govern the movement of portable objects in a scene, providing varying levels of entropy. DOMs apply to static TGs from datasets like Matterport3D (MP3D) [28] and HM3D [29]. DOMs provide standardized environments to test high-level planning by learning object transition paths.

Generalizability Benchmark: We evaluate 6 different navigation agents on a multi-object finding task across static and DOM environments. Our TAP-RL and TAP-LLM agents show an average improved Success Rate of 21.1% and generalize better by 44.5% in Relative Change in Success [27] over non-TAP counterparts.

Real-World Transfer: Finally, we present a method to transfer our work into the real-world by augmenting scene graphs. We infer that TAP-enabled agents are much better at finding targets at unexpected or unique locations, enabling more personalized assistance over non-TAP counterparts.

II. Related Works

Navigation in Dynamic Environments. Traditionally, navigation in dynamic environments has focused on *low-level* planning tasks, such as crowd avoidance [30]–[34] and socially aware navigation [35], [36], where agents continuously maneuver to avoid obstacles while minimizing trajectory length or time. In contrast, embodied navigation tasks employing *high-level* planning on graphs are predominantly studied in static environments with stationary targets, with limited research on handling dynamic targets [6], [37]. In our work, we introduce dynamism on a graph-level, where agents must reason over *long-term* spatio-temporal changes of objects for navigation decisions. While recent simulators like Habitat 3.0 [38] include object rearrangement tasks, these primarily focus on agents modifying the environment and not finding non-stationary targets on a graph. Planning in dynamic environments has been studied previously [39]–[43], with recent approaches even employing LLMs in conjunction with multi-arm bandits [44]. These schemes usually propose memory augmentations with hierarchical planning procedures and frame the problem from an obstacle avoidance standpoint [39], [45], [46]. In contrast, our work considers when objects themselves are non-stationary.

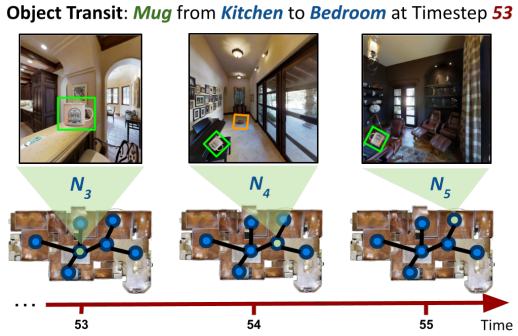

Fig. 2: **Object Transition with Dynamic Object Maps**: Objects move at various timesteps in accordance with their natural rooms and transit scenarios. A mug is placed on a kitchen node N_3 at timestep $T = 53$, but moves to a bedroom node N_5 at $T = 55$ via node N_4. If an agent reaches the kitchen after $T = 53$ (or the bedroom before $T = 55$), it fails to see the mug. Multiple objects could also be at the same node (hat and mug in $N4$).

Object Searching with Uncertainty. Rudra et. al. [2] define small, portable objects around the house and propose a contextual bandit scheme that aims to learn the likelihood of finding an object at various waypoints. In their case, however, object locations are shuffled only after each episode, meaning it finally boils down to finding stationary targets at each timestep. In contrast, we tackle a *truly* dynamic case, where objects move even *during* the episode. This definition adds a layer of complexity as the embodied agent must now plan to navigate towards a shifting target, requiring an explicit modeling of the object's transit. Kurenkov et. al. [47] introduced dynamizing household environments and experimented with scene graph memory to predict object locations. They also performed target object-finding experiments, but the target was stationary as the agent moved. In our work, the objects move as the agent moves, and we deal with topological graphs, not scene graphs. Wang et. al [48] used LLM-generated human activities to dynamize a topological graph of a household environment and performed ObjectNav experiments in them. In contrast, our work presents transit-aware planning for portable object finding.

III. Environment Formulation: Habitual Object Movement on Topological Graphs

Defining Transit Scenarios Λ: Human habit formation via object placement can be attributed to *cognitive offloading*, where individuals rely on the environment to reduce memory demands [49], [50]. This reliance on the environment influences the spatial distribution of objects. Frequently used objects (like keys or phones) might move more flexibly across rooms (high entropy), while function-specific objects (like dumbbells or toothbrushes) might be more constrained (low entropy). We use this structured transition behaviour to define an object's *transition entropy*, which we use to formulate

Λ. We consider three object transit scenarios at decreasing levels of *transit entropy*, — **Random**, **Semi-Routine**, and **Fully-Routine**, outlined in Table I.

Scenario (Λ)	Fixed Rooms	Fixed Paths	Entropy
Static	✓	N/A	None
Random	✗	✗	High
Semi-Routine	✓	✗	Medium
Fully-Routine	✓	✓	Low

TABLE I: **Object Transit Scenarios Λ**: Portable targets transition in the graph under structured scenarios inspired by human habits. In the random case, the portable objects can move to any room at any time during each episode. In the routine cases, the rooms that the target objects can travel to are fixed. The static scenario is a baseline with stationary items i.e., zero entropy.

Dynamic Object Maps (DOMs): To adapt static TG environments to test personalized, portable object finding, we define DOMs as node-attributed graphs where the attributes (objects) evolve. Let $G = (V, E)$ be a topological graph with nodes V representing locations, and static edges E are fixed paths between nodes. Let $\mathcal{O}$ be the set of portable objects on G. At each timestep t, $v \in V$ has a subset $O_v(t) \subseteq \mathcal{O}$ as attributes. Figure 2 illustrates a DOM where objects $\mathcal{O}$ move in G based on Λ. Let $\mathcal{P}$ be the set of object paths that characterize the evolving node attributes. Then, $\mathcal{P}_o \in \mathcal{P}$ is a node sequence $[v_1, \dots v_n]$ denoting the position of object o until time n. Given G, $\mathcal{O}$, and Λ, we generate a DOM to provide a setting to test methods for learning $\mathcal{P}$ needed for personalized assistance.

IV. Method: Transit-Aware Planning (TAP)

Now that we have defined our environments with portable objects, we describe our methods to learn the paths $\mathcal{P}$ from partial observability, a scenario seen in embodied tasks like ObjectNav [2], [16], for personalized decision-making. Many prior embodied navigation tasks work under the premise of "*static and stationary*", where targets are assumed to stay at one location throughout the episode. This premise is also true with zero-shot LLM-based approaches [13], [17]. With portable, non-stationary objects however, agents need to be made "aware" of target objects in transit to improve performance, and this involves integrating them with temporal knowledge of the scene. TAP agents to capture target transitions $\mathcal{P}$ in the environment as a learning objective.

TAP-RL Agent: We modify the observations, rewards, and actions on a standard RL-based embodied navigation policy. *Observations and Actions:* To make the RL agent transit-aware, at every timestep $t \in T$, we provide a snapshot of all the portable object positions $\mathcal{P}[t]$. Note, we never provide the entire object paths to the agent, as our objective is for them to learn these transitions $\mathcal{P}$. We employ a *maskable* action space to ensure the agent moves to a neighboring node of v_c. *Reward Objective:* Standard RL-based embodied navigation agents are trained with a reward to penalize the agent for being too far from the target [10], [11]: $\mathcal{R}_{\text{dist}} = -\alpha \parallel v_c - v_o \parallel$. This formulation works well with stationary targets, since an agent learns to steadily decrease its distance to v_o. In a non-stationary setting, however, v_o keeps changing, and as a result, even though R_{dist} is a dense reward, the reward signal fluctuates too much to enable meaningful learning. This fluctuation is reflected in the poor performance of DD-PPO [10] on DOMs and is discussed later in the results section.

For TAP-RL, we replace $\mathcal{R}_{\text{dist}}$ with a sparse *interception* reward $\mathcal{R}_I$ that triggers when the agent intersects an object path P_o. The reward is the number of objects in $\mathcal{O}$ with that being the case. Formally, $R_I = \sum_{o \in \mathcal{O}} \mathbf{1}_{v_c \in P_o}$, where $\mathbf{1}$ is an indicator function. This reward incentivizes the agent to *synchronize* its path with that of the target, rather than greedily "chasing" a target to minimize distance. $\mathcal{R}_I$ along with a standard target finding reward $\mathcal{R}_f$ allows for more informed agent decisions for finding portable objects.

TAP-LLM Agent: LLM-based navigation agents use *commonsense reasoning* about observations to predict navigable actions. In LLM [17] for instance, if the target is a *'leaf-shaped bowl'*, the agent uses observations of regular objects like *'table'* and *'sofa'*, to determine a direction that is more likely to have the target (the *'table'*).

Objects transit scenarios are grounded in common-sense (See Table 2 in the supplementary). passing previously seen regular objects to the LLM would help the agent make more transit-aware decisions. Since LLM agents are predominantly zero-shot and work directly during inference time with no training, we modify the system prompt to make them transit-aware. At each timestep ts_i, we gather the action sequence $a_{0->ts_i}$, and a set of objects in the scene (may include the target) o_t. In the subsequent timesteps, we pass this transit data $\mathcal{T}_i = [ts_i, a_{0->ts_i}, o_t]$ into the system prompt. A horizon is maintained to avoid token overflow by removing the earliest appended observation.

For both TAP-RL and TAP-LLM, Λ plays an important role in determining agent success. An agent finding an object in random transit Λ_{random} would find it a lot harder than one in cases $\Lambda_{\text{semi-routine}}$ or Λ_{routine}. For the former, even if an agent intersects during Λ_{random} and follows the trajectory, it may never find the target as Λ_{random} is spatio-temporally random.

V. Task: Multi-Object Finding

We focus on the task of finding multiple portable target objects over a fixed timespan. This setting adds realism to standard multi-object finding tasks such as Multi-ON [15] and GOAT [14] where targets are stationary. In our dynamic setting, we argue that multi-object finding is preferable over finding a single portable object for two reasons: **1) Realistic Setting**: In real-world homes, users frequently move multiple objects simultaneously [19], [20]. Agents must track and infer object movements to provide assistance when asked, requiring monitoring multiple portable objects (e.g., phones, watches) rather than treating each search as an isolated event. **2) Lifelong Navigation**: A multi-object setup enables an agent to adapt its search policy in real-time, by using knowledge gathered from finding previous objects. An agent deployed at homes will likely have to face similar conditions, with a user asking to find multiple targets without 'resetting' its position after each target, as with single-target tasks, similar to lifelong navigation [51].

First, we formalize navigating on a DOM. Let Λ describe the transitory motions of objects. Then, let O be the set of portable objects found over T timesteps for an agent starting at node r. Then $S = [r, T, \Lambda]$ represents a set of our experimental variables for this task. We then define our navigation objective as *"Finding the **maximum** possible number of portable objects O while traversing a dynamic environment represented by S"*. For downstream applications like embodied agents personalizing to households, finding multiple targets is a good task to study dynamic adaptability, due to dense reward signals [14]. The setting is also realistic as households have multiple people moving objects around simultaneously. We study the performance of different navigation policies with varying conditions S.

In each episode, the agent tries to find as many portable objects as possible on a DOM in T timesteps. Let τ be a finite trajectory of length T representing a sequential list of visited nodes on the DOM, G, and let $\tau(t)$ be the t-th node in τ. Now, define $O(\tau, t)$ as the set of portable objects found at $\tau(t)$. We define the set of portable objects found along τ as $O_\tau = \cup_{t=1}^{T} O(\tau, t)$. An agent with policy π explores the DOM by generating a trajectory τ_π, where each timestep t corresponds to a selected node: $\tau_\pi(t) \in V$. We now can write the policy optimization problem as finding an optimal policy π^* defined as $\pi^* = \arg\max_\pi |O_{\tau_\pi}|$. Here, π^* finds the most objects possible in T timesteps.

VI. Experiments and Results

We perform multi-object finding on the 4 different Λ scenarios defined in Table I. DD-PPO [10] and LLM [17] are baseline RL and LLM agents and are enhanced as described in Section IV to produce TAP-RL and TAP-LLM agents. We use GPT-4o for all LLM experiments. For PPO training, we use stable-baseline3[2] package and its default parameters. We also benchmark with Random and Oracle agents.

A. DOM Setup

We convert static topological graphs from the Matterport3D (MP3D) [28] dataset to obtain DOMs. Each node contains panoramic images representing the scene, and edges represent the distance between them. The $\mathcal{O}$ and Λ cases in our experiments are based on a pre-defined mapping between rooms and possible, portable objects. This mapping is provided in Table I in the supplementary, which is populated with commonsense knowledge on object locations. When generating the DOM, a pseudo-random seed s helps differentiate the movement of portable objects per episode. Letting e be the episode index, $s = e$ if Λ is random or semi-routine. For the fully routine Λ, $s = 1$ across episodes, so the object transition routes remain fixed.

We determine 4 hyperparameters that influence performance i.e., $H = [r, E, T, \Lambda]$, where $r \in G$ is a random starting node chosen from G, E is is the number of episodes per trial, T is the number of timesteps per episode and Λ is the transit scenario. Thus, each agent is subjected to a total of $[Stationary(1) + Transit(3)] \times r \times E \times T$ experiments. We define each trial as a set of episodes over a fixed time from a starting node r. The maximum possible number of trials for a scan is thus equal to the size of G. In our subset of 10 MP3D scans, $|r|$ ranges from $[20, 215]$. For each scan, we set $|r| = 10$, $E = 20$, and $T = 30$. Although different objects can have different transition scenarios, we set all objects to have the same Λ to better analyze the generalizability of a navigation scheme.

Ground Truth Paths: Given a starting node $r \in G$, for each episode $e \in E$, there exist multiple optimal policies $\pi^* \in \Pi^*$ over T generating ground truth paths τ_{π^*} that the agent could take for collecting the most objects. We compute these paths by brute force, simulating all possible trajectories from r over T timesteps, and use them in our evaluation.

B. Evaluation Metrics

Success Rate (SR): For each episode, we measure the object-finding performance of an agent by dividing the number of objects found in that episode by the maximum possible number of objects that could have been found with an optimal policy π^*: $SR = |O_{\tau_\pi}|/|O_{\tau_{\pi^*}}|$. This metric is similar to the Progress metric in [15].

Trajectory Alignment (TA): To measure path efficiency, we modify the standard SPL metric [52] to define TA as the maximum overlap between a ground-truth path and the agent's trajectory. Formally, $TA = \max_{\pi^* \in \Pi^*} |\tau_\pi \cap \tau_{\pi^*}|$.

Relative Change in Success (RCS): We use RCS [27] to measure the generalizability from static to dynamic environments. It is the percent relative change between static and dynamic navigation performance. We measure RCS for various Λ cases and report scores on SR and TA. Lower values indicate better generalizability, showing consistency in agent performance between static and dynamic environments.

C. Navigation Inference

Main Results: Table II presents the results of our agents on the navigation task on DOMs. We draw 4 key inferences to understand the generalizability of the navigation approaches:

1) TAP improves Generalizability: Figure 3 presents a comparison of all approaches on RCS. TAP-LLM and TAP-RL both have a relatively low RCS score, meaning that navigation agents perform comparably in stationary graphs and DOMs. This highlights the effectiveness of incorporating transit-awareness in portable objects. In contrast, the greedy heuristic Oracle agent and the zero-shot LLM lack transit information, with the former relying on privileged knowledge and latter utilizing commonsense cues about static observations for decision-making. We can infer TAP as vital for the real-world deployment of agents. Future work can explore the heuristic estimation of human habits as a prior for estimating transit paths for TAP.

2) Zero-shot Methods Camp to Find Objects: In the vanilla LLM approach, the agent finds more objects in the DOM than in the static environment, indicated by the higher DOM SR scores across all Λs. From Table II, in the Fully-Routine case for LLM, the RCS for SR is $+16.7\%$ but the RCS for TA is -10.8%. These results suggest that while the LLM agent is finding more objects in DOMs than in static environments, its trajectories show less overlap with Π^*. Upon inspection, we

[2]https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

Policy	Metric (%)	Λ Random			Λ Semi-Routine			Λ Fully-Routine			Λ Average		
		Static	DOM	RCS	Static	DOM	RCS	Static	DOM	RCS	Static	DOM	RCS
Random	SR	42.0	74.4	+43.5	42.0	54.7	+23.2	42.0	48.3	+13.0	42.0	59.1	+29.0
	TA	33.5	24.4	−37.3	33.5	36.6	+8.4	33.5	33.6	**+0.29**	33.5	31.2	−6.8
Oracle	SR	90.2	78.4	−13.0	90.2	75.1	−16.8	90.2	55.9	−38.0	90.2	69.8	−22.6
	TA	45.6	54.4	+16.2	45.6	50.6	+9.8	45.6	32.7	−28.3	45.6	45.9	**+0.6**
RL [10]	SR	61.8	33.4	−45.9	61.8	21.8	−64.7	61.8	12.2	−80.1	61.8	22.4	−63.7
	TA	59.8	36.5	−38.9	59.8	29.8	−50.1	59.8	24.1	−59.6	59.8	30.1	−49.6
TAP-RL	SR	59.1	55.2	−6.6	59.1	57.8	**-2.3**	59.1	55.2	−6.7	59.1	56.0	−5.2
	TA	41.6	37.3	−10.3	41.6	35.8	−14.0	41.6	32.1	−22.9	41.6	35.1	−15.7
LLM [17]	SR	28.5	43.3	+34.1	28.5	47.2	+39.6	28.5	34.2	+16.7	28.5	41.6	+31.4
	TA	24.3	30.7	+20.9	24.3	31.3	+22.3	24.3	21.7	−10.8	24.3	27.9	+12.9
TAP-LLM	SR	50.6	49.4	**-2.4**	50.6	53.2	+4.8	50.6	47.9	**-5.4**	50.6	50.2	**-0.9**
	TA	36.9	33.8	**-8.5**	36.9	34.6	**-6.3**	36.9	32.4	−12.2	36.9	33.6	−9.0

TABLE II: Navigation on DOMs: We evaluate 6 embodied navigation approaches on various DOMs to establish a benchmark. These values are used to compute RCS using the equation provided in [27]. An RCS value closer to 0% indicates good generalizability to new conditions. Highlighted in green are the best RCS values for SR, while ones in violet are the best values for TA. Observe the improved RCS scores of TAP-RL and TAP-LLM agents, indicating better generalizability to portable targets. Particularly note that the RL agent performs well in the static scenario, with a significant performance drop when subjected to DOMs. The LLM agent shows improved performance on DOMs, but we attribute this to multiple chance encounters with moving targets.

observe that this agent has learned to *camp* between privileged nodes that are in many object routes. One possible explanation is that the LLM is relying solely on common-sense knowledge and goes to areas where most objects can be found, rather than trying to learn object routes. This challenge opens up research directions in DOM navigation reward design.

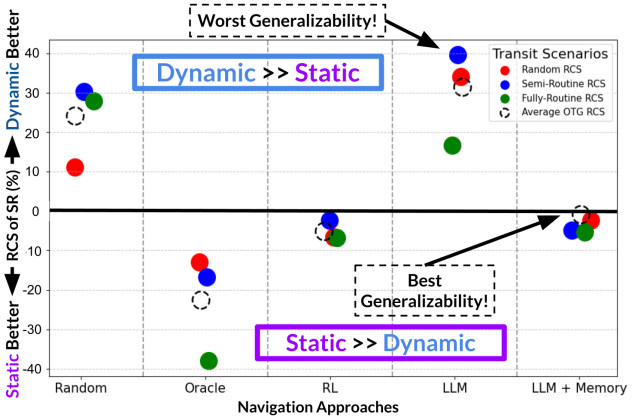

Fig. 3: Navigation Generalizability on DOMs: Relative Change in Success (RCS) [27] measures an agent's adaptability, comparing dynamic to static TG performance (optimal RCS is 0%). Positive values indicate better dynamic performance, while negative values reflect poor adaptability. Both our TAP-enhanced agents approach optimal RCS, indicating better target finding in dynamic scenarios.

3) Greedy Oracle Struggles on DOMs: The Oracle agent uses a greedy heuristic that works well in Λ_{static} by selecting a node towards the closest target object. On DOMs, however, it tends to take a suboptimal path because the agent keeps switching directions when another moving target object appears to be closer. This behavior is a direct approximation of the distance reward with DD-PPO that causes target chasing and highlights the need for transit awareness.

4) Entropy Influences Performance: In all policies barring the Random agent, the performance on a Λ_{routine} exhibiting low entropy is usually worse than on a $\Lambda_{\text{semi-routine}}$ or Λ_{random}. This result can be attributed to the agent encountering more objects in environments with higher transit entropy. Despite this, we observe a strong performance of the TAP-RL agent in this case, with similar SR and TA values across all transit cases. This indicates that TAP can enhance embodied RL agents to become agnostic to scenarios where all target objects follow the same Λ. Future work could study a mixed Λ with objects in the same DOM following distinct transit scenarios.

Ablation: Partial Observability with TAP-RL: We ran an ablation experiment where we only provided the TAP-RL agent with the number of objects for the current adjacent nodes. In terms of success rate (%), we see degradation in performance in all DOM scenarios, Λ_{routine} : $55.2 \rightarrow 12.6$, Λ_{semi} : $57.8 \rightarrow 12.7$, and Λ_{random} : $55.2 \rightarrow 14.3$, showing that an RL agent has better planning when given a larger observation space, indicating a need to use map-based planning with RL-based agents on DOMs.

D. Discussion: Real-World Utility

For a real-world agent to be accepted, it must slightly over-deliver on what it was intended to do [53]. Learning user routines is an important capability in this regard, as it enables personalized decision-making in the absence of humans.

Real-World Lab Experiment: Our objective is to collect a scene graph of the real-world, make it dynamic by adding portable objects to it using DOMs, and run TAP agents. The entire process is highlighted in Figure 4. We conduct real-world experiments as follows:

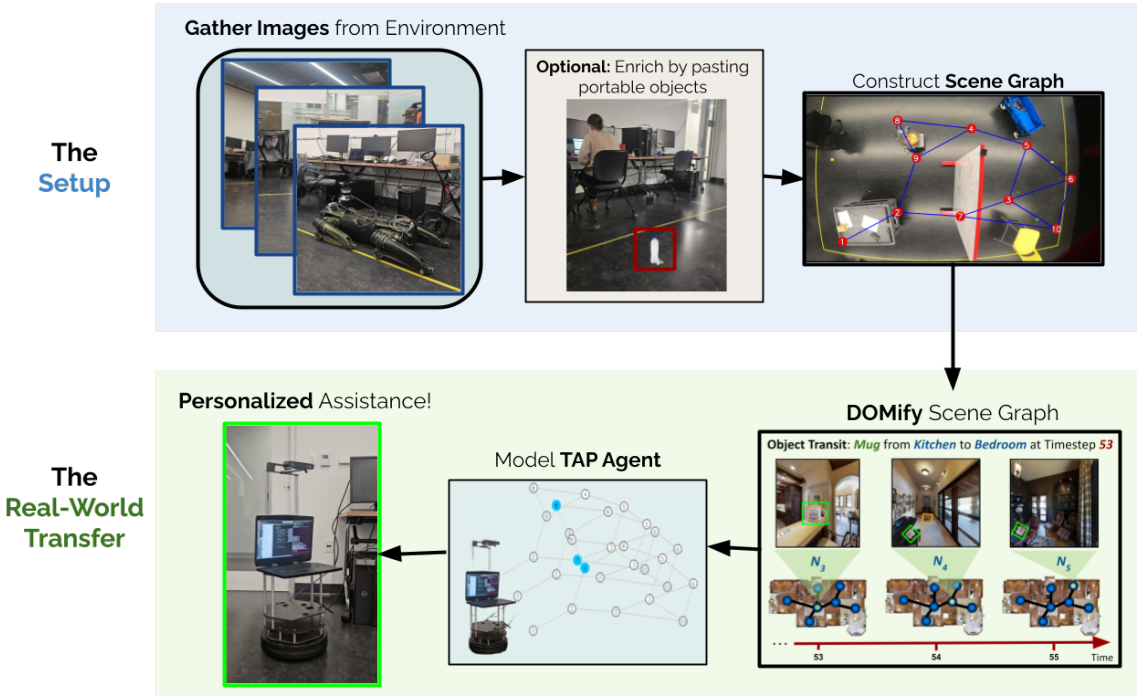

Fig. 4: Real World Transfer: We present a real-to-sim pipeline for a real-world transfer of TAP agents. Since TAP agents learn habits over large periods of time, we create a simulation on a topological graph of the environment to enable us to gather statistics. The first phase involves gathering images from the scene and constructing a topological scene graph. We also optionally add small portable targets into these images to enrich this data (bottle in the top row, second image). We then DOMify these scene graphs, making it dynamic with moving objects, using Algorithm 1, and model a TAP agent with this data. Finally, we transfer this to the real-world for personalized assistance. Table III reports our results on **5** diverse environments. Please refer to the video attached that describes our pipeline and provides visuals.

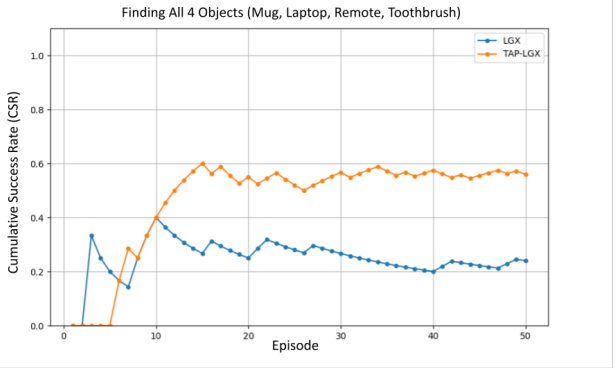

Fig. 5: Cumulative Success Rate (CSR): TAP-LLM outperforms the vanilla LLM overall in finding portable targets over time. Note the cumilitive success of finding these targets keeps increasing, since the TAP agent is made aware of transit behavior.

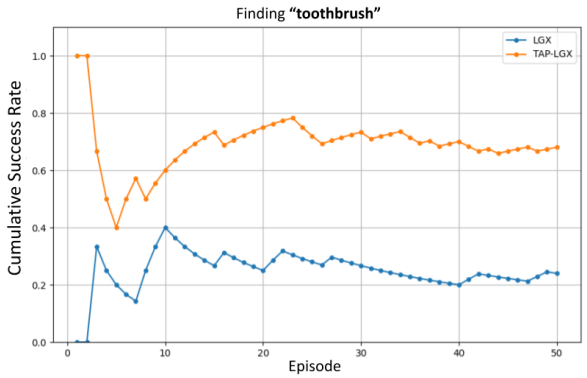

Fig. 6: Non-Commonsense Target Success: Observe the TAP-LLM agent can find non-commonsense targets like a toothbrush in our lab environment much better than its non-TAP counterpart.

1) **Scene Graph Construction**: For constructing a scene graph $\mathcal{S}$, we first collect images of our lab environment at different times of the day, and represent them across 10 nodes. Each node represents a small location in our lab, containing multiple viewpoints of the scene. These viewpoints cover the periphery and middle spaces of our lab where people often move objects around. We run YOLO on these images and the detected objects of different shapes and sizes are the node attributes. Edges represent distances.
2) **DOM Construction**: We place 4 portable target objects across the nodes, as text descriptions for the LLM agent's system prompt: **white mug, laptop, black remote, toothbrush**. The white mug and laptop are *commonsense* objects that are naturally present in labs. The *black remote* and *toothbrush* however, are out of place here, i.e., *non-commonsense* targets. We purposefully choose these items to demonstrate the effectiveness of TAP agents for personalized target finding. Additionally, we gather more data samples that correspond to first-person views of the agent by enriching these images by pasting portable targets in

	Random Λ (%)	Semi-Routine Λ (%)	Routine Λ (%)
LLM	37.5	45.0	57.5
TAP-LLM	**50.0**	**65.0**	**80.0**

TABLE III: Real World SR: We report average success rates for finding portable objects in our lab environment over 50 episodes. Note the improved performance of the TAP-LLM approach over a vanilla LLM scheme. Transit awareness helps improve performance in the routine cases, where paths have structure.

them, and this is discussed in Figure 4. Finally, we introduce shifting objects into the constructed scene graph to convert it into a DOM, as described in Algorithm 1. We use the routine case, where the target objects move in exactly the same way for each episode.

3) **Run TAP Agent:** We run both LLM and TAP-LLM to identify each of the 4 targets, seeking to demonstrate the benefits of our TAP module in real-world settings. During each episode, the agent moves between nodes and samples a random viewpoint from the gathered data. For a vanilla non-TAP agent, we use LLM in its vanilla form, which purely operates on commonsense knowledge on objects detected to decide where to go. For the TAP agent, we additionally augment LLM with memory to keep track of timesteps, actions, and objects detected, and this info is passed on across episodes.

Inferences: We report Cumulative Success Rate (CSR) values on 50 episodes for the TAP-LLM and a vanilla LLM agent in graphs 5 and 6. In graph 5, we highlight the significant improvement of the TAP agent over time, in learning to find targets better than the vanilla LLM agent (LLM). Figure 6 presents the success of specifically the **toothbrush**, an unusually placed object, that is usually not present in office environments. Observe that TAP-LLM shows superior performance in finding this target that does not follow commonsense placement, since it can learn routine object paths over relying on just commonsense. Please refer to the new supplementary video attached for a detailed explanation and more visuals. We also plot in Figure 7 the number of steps to find the first target object, or **Time-To-Find (TTF)**. Figure 7 shows the TTF between LLM and TAP-LLM. Observe that TAP-LLM finds targets much faster over time. This results supports what Figure 5 shows, with TAP-LLM improving its success over time to find objects over the vanilla LLM agent.

The "*real-to-sim*" transfer allows us to simulate multiple runs of TAP-LLM agents on this graph and validate the deployability of our agent in the real world. Further, it allows us to derive various statistics as we did with MP3D earlier. We run 50 episodes to obtain success rates in various object transit scenarios. These are shown in Table III.

VII. Conclusion, Limitations, and Further Work

We present a novel approach to perform embodied navigation in dynamic environments with portable objects. Our method enhances RL and LLM-based policies traditionally modelled for stationary targets with a TAP strategy that enables them to capture object transit information. To evaluate our TAP-enhanced agents, we introduce DOMs that translate static topological graphs into dynamic ones with shifting objects. Object transitions in DOMs aim to

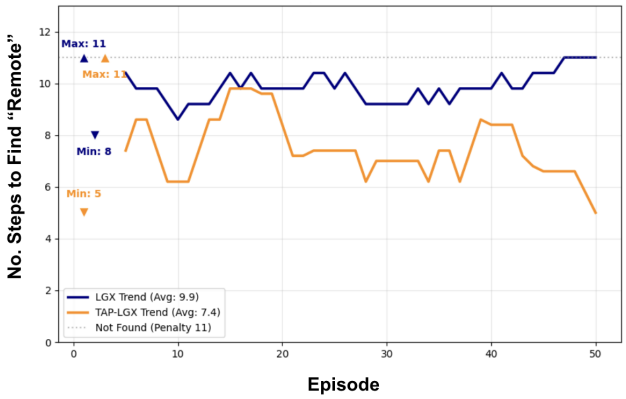

Fig. 7: Number of steps to find the "remote" object in the lab environment shown in Figure 4. Note the improved performance of TAP-LLM over time, in being able to find the remote much faster than the LLM agent. This highlights the influence of memory in enabling better non-stationary target finding over time.

simulate human object placement habits. Our work challenges the underlying *"static graph, stationary target"* assumption common with embodied navigation literature, and presents a navigation benchmark in a dynamic environment with non-stationary, portable objects. On MP3D, our TAP-enhanced agents outperform non-TAP counterparts by 21.1% when measured by average Success Rate in DOMs, while also drastically generalizing better to non-stationary environments by over 44.5% when measured by RCS. We further conduct a real-world transfer of our approach, and note the superior performance of TAP agents on objects that are not placed in commonsense locations (eg. toothbrushes in labs). Future work could study mixed object transit cases and incorporate agents with higher degrees of freedom. Other directions could include incorporating more sophisticated measures of semantic memory to reduce the context length while maintaining dense information on human habits. While foundational knowledge can be help, it also can hurt generalizability across cultural semantics, e.g., some households call a "*flashlight*" is called a "*torch*". Code and data will be made publicly available to foster research.

References

[1] 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," *arXiv*, 2020.

[2] S. Rudra, S. Goel, A. Santara, C. Gentile, L. Perron, F. Xia, V. Sindhwani, C. Parada, and G. Aggarwal, "A contextual bandit approach for learning to plan in environments with probabilistic goal configurations," in *2023 IEEE International Conference on Robotics and Automation*.

[3] 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*, vol. 33, pp. 4247–4258, 2020.

[4] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 890–18 900.

[5] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "Vln bert: A recurrent vision-and-language bert for navigation," in *Proceedings of IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021.

[6] K. Chen, J. K. Chen, J. Chuang, M. Vázquez, and S. Savarese, "Topological planning with transformers for vision-and-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 276–11 286.

[7] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3674–3683.

[8] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. van den Hengel, "REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments," Jan. 2020.

[9] A. J. Zhai and S. Wang, "Peanut: predicting and navigating to unseen targets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 926–10 935.

[10] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," 2020.

[11] K. Yadav, A. Majumdar, R. Ramrakhya, N. Yokoyama, A. Baevski, Z. Kira, O. Maksymets, and D. Batra, "Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav," *arXiv:2303.07798*, 2023.

[12] N. Kim, O. Kwon, H. Yoo, Y. Choi, J. Park, and S. Oh, "Topological semantic graph memory for image-goal navigation," in *Conference on Robot Learning*. PMLR, 2023, pp. 393–402.

[13] D. Shah, B. Osinski, B. Ichter, and S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action," Jul. 2022.

[14] e. a. Chang, Matthew, "Goat: Go to any thing," in *Robotics: Science and Systems*, 2024.

[15] S. Wani, S. Patel, U. Jain, A. Chang, and M. Savva, "Multion benchmarking semantic map memory using multi-object navigation," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[16] 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," *arXiv:2006.13171*, 2020.

[17] V. S. Dorbala, J. F. Mullen Jr, and D. Manocha, "Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation," *IEEE Robotics and Automation Letters*, 2023.

[18] K. Chen, J. K. Chen, J. Chuang, M. Vazquez, and S. Savarese, "Topological Planning with Transformers for Vision-and-Language Navigation," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun.

[19] M. Patel and S. Chernova, "Proactive robot assistance via spatio-temporal object modeling," *arXiv preprint arXiv:2211.15501*, 2022.

[20] E. Bartoli, F. I. Dogan, and I. Leite, "Streaming network for continual learning of object relocations under household context drifts," *arXiv preprint arXiv:2411.05549*, 2024.

[21] D. do Couto Teixeira, J. M. Almeida, and A. C. Viana, "On estimating the predictability of human mobility: the role of routine," *EPJ Data Science*, vol. 10, no. 1, p. 49, 2021.

[22] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[23] T. Campari, L. Lamanna, P. Traverso, L. Serafini, and L. Ballan, "Online learning of reusable abstract models for object goal navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 870–14 879.

[24] D. Zheng, S. Huang, L. Zhao, Y. Zhong, and L. Wang, "Towards learning a generalist model for embodied navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 13 624–13 634.

[25] J. Lin, H. Gao, X. Feng, R. Xu, C. Wang, M. Zhang, L. Guo, and S. Xu, "Advances in embodied navigation using large language models: A survey," *arXiv preprint arXiv:2311.00530*, 2023.

[26] J. Li, H. Tan, and M. Bansal, "Improving cross-modal alignment in vision language navigation via syntactic information," *arXiv*, 2021.

[27] V. S. Dorbala, G. A. Sigurdsson, J. Thomason, R. Piramuthu, and G. S. Sukhatme, "Clip-nav: Using clip for zero-shot vision-and-language navigation," in *Workshop on Language and Robotics at CoRL*, 2022.

[28] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[29] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," *arXiv:2109.08238*, 2021.

[30] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation*.

[31] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[32] K. Cai, C. Wang, J. Cheng, C. W. De Silva, and M. Q.-H. Meng, "Mobile robot path planning in dynamic environments: A survey," *arXiv preprint arXiv:2006.14195*, 2020.

[33] C. Park, J. Pan, and D. Manocha, "Real-time optimization-based planning in dynamic environments using gpus," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4090–4097.

[34] ——, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of the international conference on automated planning and scheduling*, vol. 22, 2012.

[35] E. Cancelli, T. Campari, L. Serafini, A. X. Chang, and L. Ballan, "Exploiting proximity-aware tasks for embodied social navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 957–10 967.

[36] C. I. Mavrogiannis and R. A. Knepper, "Decentralized multi-agent navigation planning with braids," in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 880–895.

[37] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

[38] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min *et al.*, "Habitat 3.0: A co-habitat for humans, avatars and robots," *arXiv*, 2023.

[39] V. Vladareanu, G. Tont, L. Vladareanu, and F. Smarandache, "The navigation of mobile robots in non-stationary and non-structured environments," *International Journal of Advanced Mechatronic Systems*, vol. 5, no. 4, pp. 232–242, 2013.

[40] Y. Zhou and H. W. Ho, "Online robot guidance and navigation in non-stationary environment with hybrid hierarchical reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 114, 2022.

[41] T. Morris, F. Dayoub, P. Corke, G. Wyeth, and B. Upcroft, "Multiple map hypotheses for planning and navigating in non-stationary environments," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 2765–2770.

[42] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[43] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: obstacles moving along arbitrary trajectories," in *Proceedings 2001 IEEE International Conference on Robotics and Automation*.

[44] J. de Curtò, I. de Zarzà, G. Roig, J. C. Cano, P. Manzoni, and C. T. Calafate, "Llm-informed multi-armed bandit strategies for non-stationary environments," *Electronics*, vol. 12, no. 13, p. 2814, 2023.

[45] W. Li, R. Hong, J. Shen, L. Yuan, and Y. Lu, "Transformer memory for interactive visual navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1731–1738, 2023.

[46] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, T. Yang, and C. Fan, "A deep bayesian policy reuse approach against non-stationary agents," *Advances in neural information processing systems*, vol. 31, 2018.

[47] A. Kurenkov, M. Lingelbach, T. Agarwal, E. Jin, C. Li, R. Zhang, L. Fei-Fei, J. Wu, S. Savarese, and R. Martın-Martın, "Modeling dynamic environments with scene graph memory," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 976–17 993.

[48] C. Wang, X. Li, D. Wang, H. Liu *et al.*, "Dynamic scene generation for embodied navigation benchmark," in *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

[49] E. F. Risko and S. J. Gilbert, "Cognitive offloading," *Trends in cognitive sciences*, vol. 20, no. 9, pp. 676–688, 2016.

[50] K. S. Smith and A. M. Graybiel, "Habit formation," *Dialogues in clinical neuroscience*, vol. 18, no. 1, pp. 33–43, 2016.

[51] R. R. Wiyatno, A. Xu, and L. Paull, "Lifelong topological visual navigation," *IEEE Robotics and Automation Letters*, vol. 7, 2022.

[52] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv:1807.06757*, 2018.

[53] [Online]. Available: https://rodneybrooks.com/rodney-brooks-three-laws-of-robotics/

SUPPLEMENTARY MATERIAL

VIII. DOMIFYING MP3D

We convert MP3D graphs into DOMs with the 21 different objects presented in Table 3. On each of the 10 scans chosen, we introduce routine, semi-routine and random object movement, and provide these as dictionaries with varying seeds. We will release these modified environments along with supporting code.

A. Implementing Object Placement

In this section, we give more details on how we convert Matterport3D environments into DOMs.

Matterport3D Modifications:

Each Matterport3D (MP3D) scan represents a household environment consisting of a set of panoramic view points. Along with the viewpoints (or nodes), we are also given the exact 3D position as well as the relative distance between them. For modifying the MP3D environment, we first construct topological graphs of each scan, with nodes containing the position and the panoramic image, and edges containing relative distance between them. We consider 10 different scans chosen from the REVERIE [8] and R2R [7] unseen validation splits for inference. We choose scans according to these datasets as they contain a variety of rooms for us to populate (Refer Table IV in the main paper).

The nodes of the topological graphs are then updated at each timestep with the portable objects according to the object placement scenario that has been chosen.

Strategy Overview:

We compute trajectories of the portable objects offline. For a fixed random seed s and for each portable object $o_p \in O_p$, we create a sequence of nodes from the graph. In the random transit scenario, we choose any node in the graph. In the routine and semi-routine scenarios, we only choose nodes from plausible rooms. If the node chosen is not the same as the current node, we find the shortest path between the current node and the target node and add the nodes in the path to the sequence representing the trajectory. Once o_p gets to the target node, it stays there for either 2 or 3 timesteps, determined by a random number generator. For each timestep o_p stays at the node, we add the node to the trajectory sequence. After the staying period is over we select a new node and repeat the process until we hit T timesteps.

We take the resulting trajectories and restructure the data to model an evolving graph over a period of T timesteps. The resulting structure is a nested dictionary representing the changing graph. The keys of the outer dictionary are the timesteps $[1, ..., T]$. The inner dictionary for each timestep t has the node string ids as the keys and the values are the list of portable objects at that node at T. At each timestep $t \in [1, ..., T]$ when the agent reaches a node, we simply use t and the node id to retrieve the list of portable objects the agent is currently observing.

B. TAP-PPO

We modify a DD-PPO agent with temporal information about target objects in its observation space. As mentioned in section IV, we gather object paths $\mathcal{P}$ for augmenting this space.

In our task, since we treat each object equally, i.e. the objective is simply to maximize finding targets, we include $\mathcal{P}$ as a single list $l = [v_1, v_2, \ldots, v_n]$ of size $n = |G|$. Each index of l corresponds to a different node in G. For each index i of l, v_i is the number of unique target objects at the i-th node in G. l is updated at each timestep, and passed to the observation space along with the agent's current position and timestep. The agent's action space is masked, with the entire graph being the action space, and the mask allowing only for selecting neighboring nodes.

Additionally, we swap out the distance reward R_{dist} with the intersection reward R_I described in the main text. R_I here is triggered whenever the agent encounters a node where a target object has been in the last 5 timesteps. We do not make any other modifications to the default setup.

C. TAP-LLM

We utilize the official implementation of LGX [3] and incorporate it into our modified MP3D environment. Briefly, at each timestep, LGX scans the node for objects, and asks an LLM for directions to reach a target. In our case, we seek to maximize finding portable targets. As such, we prompt GPT-4 with the following base prompts -

> **System Prompt** - "I am a smart robot trying to find as many portable objects as I can at home."
> **User Prompt** - "Which object from <*OBJECT_LIST*> should I go towards to find a new portable object? Reply in ONE word."

The <OBJECT_LIST> here contains a set of objects that have been detected using YOLO-v8, as proposed in LGX. Additionally, we if a portable object is present at a certain node at a given timestep, we add it to this list.

The LLM then predicts a target object from the list, which is mapped to an adjacent node using our customized MP3D functions. The agent then moves to that node.

For the TAP-LGX case, we modify the System Prompt, with memory as presented in Section IV. We additionally ask -

> **System Prompt**:
> "I have seen the following objects and taken the following actions so far -
> 1. <OBJECT_LIST>: ACTION
> 2. <OBJECT_LIST>: ACTION
> ..."
> **User Prompt**:
> "Which object from <*OBJECT_LIST*> should I go towards to find a new portable object? Reply in ONE word."

Note that we maintain a horizon to avoid token overflow, where the earliest appended observation is removed when the context length limit is reached.

IX. VIDEO, CODE AND DATASET

We provide an anonymous link to our Code: `https://anonymous.4open.science/r/otg-1AC0`. The

[3] `https://github.com/vdorbala/LGX`

Room	Portable Objects
Bedroom	Charger, Water Bottle, Smartwatch, Laptop, Notebook, Toothbrush, Mug, Flash Drive, Phone, Headphones, Hat
Garage	Screwdriver, Flashlight, Mug, Phone, Headphones, Hat
Dining	Salt and Pepper Shakers, Portable Speaker, Charger, Water Bottle, Mug, Bowl, Phone, Headphones, Hat
Office	Charger, Laptop, Hat, Notebook, Flash Drive, Mug, Phone, Headphones
Bathroom	Toothbrush, Phone, First-Aid Kit
Kitchen	Salt and Pepper Shakers, Hat, Mug, Bowl, Phone, Headphones, First-Aid Kit
Lounge	Playing Cards, Mug, Portable Speaker, Charger, Water Bottle, Laptop, Phone, Flash Drive, Dice, Headphones, Hat
Gym	Dumbbells, Jumprope, Smartwatch, Phone, Headphones, Hat
Outdoor	Jumprope, Smartwatch, Portable Speaker, Phone, Water Bottle, Headphones, Hat
Recreation	Playing Cards, Dice, Water Bottle, Headphones, Hat

TABLE IV: Rooms and Portable Objects Mapping for Experiments: We map 21 portable objects to a set of household rooms. We generate this mapping with GPT-3.5, asking it to provide us a list of rooms commonly found in a house and common objects you would typically find in them. This mapping is used to set destinations for object transit. During each episode, objects are placed in various rooms for a range of timesteps. Commonly moved objects such as *phone, headphones* are associated with 9 different rooms, while less commonly shifted ones such as *dumbbells* appear only in the Gym.

TABLE V: Summary statistics of scans used for Table II, including each scan's ID, number of nodes, edges, and rooms. If a scan has multiple rooms of the same type (e.g. two bathrooms), each instance is counted separately from the total.

Scan ID	# of Nodes	# of Edges	# of Rooms
QUCTc6BB5sX	145	248	28
8194nk5LbLH	20	32	6
TbHJrupSAjP	114	221	28
2azQ1b91cZZ	215	531	30
oLBMNvg9in8	111	185	31
zsNo4HB9uLZ	53	84	17
EU6Fwq7SyZv	78	166	19
X7HyMhZNoso	84	143	25
x8F5xyUWy9e	43	86	10
Z6MFQCViBuw	58	91	18

dataset will be present as a downloadable link in the code repository.

Furthermore, please see the attached video in the zip file for more information and demonstrations.