PPENDIX
SUPPLEMENT RY DET ILS: PERPETU *

.1 Perpetua* lgorithm

In lgorithm 1, we present the update and prediction routines for a single receptacle in Perpetua*, omitting the receptacle index k for clarity. During the update step, we compute the likelihood, conditional evidence, and posterior weights for both the persistence and emergence models. In the prediction step, we generate the Perpetua* prediction at an arbitrary query time using our proposed Bayesian model selection criterion.

lgorithm 1: Predict and Update Functions for Perpetua* (Single Receptacle)

Parameters: Models $m \in \{\mathcal{M}_{\mathrm{P}}, \mathcal{M}_{\mathrm{E}}\}$, Decay rate $_0$, Forgetting factor γ for receptacle k

▷ `Updates the state of each model`

Update Input: Observation tuple $(y_t \quad _1, t_{N+1})$

Function `Update(`$y_t \quad _1, t_{N+1}$`)`**:**

  for $m \in \{\mathcal{M}_{\mathrm{P}}, \mathcal{M}_{\mathrm{E}}\}$ **do**

    Update likelihood ℓ_{N+1} in (3) with γ:

    $\ell_{N+1} \leftarrow \ell_N^{\gamma} \, P_M^{1 \quad y_t \quad 1}(1 \quad P_M)^{y_t \quad 1}$;

    for $l \leftarrow 1$ **to** L **do**

      Update accumulator $H(\mathcal{Y}_{1:N+1} \,|\, c_l, m)$ via (5);

      $p(\mathcal{Y}_{1:N+1} | c_l, m) \leftarrow H_N + \ell_{N+1}[1 \quad F_l(t_{N+1})]$;

    Update posterior weights w_l via (6);

  $N \leftarrow (N+1)$

▷ `Predicts persistence state at time` t

Predict Input: Query time t for receptacle k

Function `Predict(`t`)`**:**

  for $m \in \{\mathcal{M}_{\mathrm{P}}, \mathcal{M}_{\mathrm{E}}\}$ **do**

    Evaluate predictions $p(x_t \,|\, c_{l^*}, \mathcal{Y}_{1:N}, m)$ via (7);

  $(t) \leftarrow \exp(\quad _0(t \quad t_N))$;

  Compute model posterior $p(\mathcal{M} \mid \mathcal{Y}_{1:N})$ via (9);

  Compute final prediction $p(x_t \,|\, \mathcal{Y}_{1:N})$ via (10);

  return $p(x_t \mid \mathcal{Y}_{1:N})$;

Tool 1: Semantic Object Search

Input: `query_text` (string)
Output: List of ⟨ `object_id`: string, `score`: float ⟩ or `None`

Description: Performs an open-vocabulary search over PredictiveGraphs using CLIP. It compares the embedding of the `query_text` against the visual embeddings of all mapped objects, returning the `object_ids` with the highest cosine similarities.

Tool 2: Predict Object Receptacle

Input: `object_id` (string), `query_time` (float)
Output: Most likely receptacle name (string), probability distribution over receptacles (list of floats)

Description: Queries the temporal object map (PredictiveGraphs) using Perpetua* to predict the location of the target object at the specified `query_time`. The tool applies a confidence threshold (default $\delta = 0.2$) to filter receptacles. If no receptacle exceeds the threshold, the object is reported as absent.

Tool 3: Navigate to Receptacle

Input: `receptacle_id` (string), `object_id` (string)
Output: Success status (boolean), final observation

Description: Navigates to the `receptacle_id` while actively searching for the target `object_id`. During navigation, the agent continuously: (1) executes path planning toward the receptacle, (2) updates PredictiveGraphs with RGBD observations, and (3) checks whether the target object is in view. Navigation terminates when either the object is found or the target receptacle is reached.

PPENDIX B
SUPPLEMENT RY & IMPLEMENT TION DET ILS

B.1 LLM Tools

In this section, we provide the formal specifications for the three core tools that enable embodied planning in PredictiveGraphs: *semantic search* (Tool 1), *location prediction* (Tool 2), and *active navigation* (Tool 3). For each tool, we define the required inputs, the expected outputs, and the underlying logic used to interface with PredictiveGraphs.

B.2 LLM Prompt - Switching Prior

We present the full LLM prompt used to query the LLM-based switching prior described in Sec VI- 1. We employ a few-shot prompting strategy to leverage the semantic knowledge of `gemini-2.5-pro`, enabling it to infer probable weekly schedules for objects based on their household context and potential usage patterns. The prompt is shown in Prompt 1.

Prompt 1: LLM Prior for Perpetua*

Consider a household environment composed of two parents and two children. The environment contains various objects located in different places, where objects can appear when needed for an activity (e.g., plates for eating) or disappear when taken by a member (e.g., keys/bags when leaving). During weekdays, the parents typically leave for work in the morning and return in the evening (07:30 - 18:00), while the children attend school during the day (08:00 - 17:00). On weekends, the family tends to stay home more often, engaging in activities such as cooking, cleaning, and leisure time together.

Task: Generate a weekly schedule (Mon-Sun) for the specific object at the specific location.

Output Rules:

- Output ONLY a Python list of strings.
- Format: ['Day: HH:MM - Day: HH:MM', 'Day: HH:MM - Day: HH:MM', ...]
- Use 3-letter day abbreviations (Mon, Tue, Wed, Thu, Fri, Sat, Sun).

Example 1:
Input:
Object: Plates
Location: On Dinner Table
Type: ctive Location (appear when used)
Output:
Schedule: ['Mon: 19:00 - Mon: 19:30', 'Tue: 19:00 - Tue: 19:30', ... , 'Sun: 18:00 - Sun: 18:30']
Current task:
Input:
Object: {object_name}
Location: {object_location}
Type: {object_type}
Output:
Schedule: [...]

B.3 Perpetua Implementation Details*

In this section, we detail the parameters and modeling choices for Perpetua*.

a) Hyperparameters: The prior p_l in (1) is modeled as a mixture of log-normal distributions. We set the forgetting factor $\gamma = 0.99$, which maintains an effective memory of approximately 250 observations. The decay rate is set to $_0 \approx 0.01$, causing the model to revert almost entirely to the prior after approximately 300 time units.

b) Model Selection: Following the configuration in Perpetua [10], we search for the optimal number of mixture components (up to a maximum of 5) for both persistence and emergence filters using the kaike information criteria (IC) [36]. The observation noise parameters (P_M, P_F) are estimated directly from dataset statistics.

c) Switching Priors: We evaluate Perpetua* with three variants of the switching prior $f(t)$:

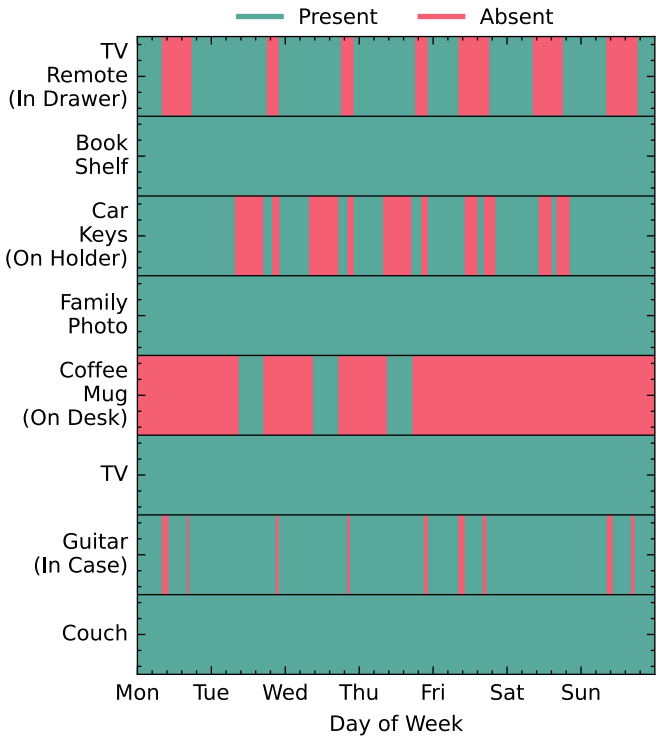

Fig. 8. **Weekly Test Schedule:** One-week snapshot of the test set with out-of-distribution dynamics for the different objects used in the Perpetua evaluation presented in Sec.VI- . Note how Friday and Monday follow the same dynamics as the weekend, emulating a "long weekend".

1) **FreMEn Prior:** Based on (8), this variant computes 1000 Fourier coefficients and selects the optimal subset via a held-out validation set of one week.
2) **LLM Prior:** Generated via `gemini-2.5-pro` [14] using the formulation in (8). The full prompt is shown in ppendix B.2.
3) **Oracle Prior:** Uses step functions that perfectly match the ground truth dynamics of the training set.

B.4 Schedule

While the training set used to train Perpetua* in Sec VI- reflects a standard weekly routine with distinct weekday and weekend patterns, the test set introduces a distributional shift to evaluate adaptation. Specifically, we construct a "long weekend" scenario where Monday and Friday exhibit weekend-like dynamics. This modification forces the model to adjust to sudden schedule changes, serving as a stress test for real-time adaptation. snapshot of these shifted dynamics is provided in Fig. 8.

PPENDIX C
SUPPLEMENT RY RESULTS: PERPETU *

*C.1 Complexity nalysis and Illustrative Example: Perpetua**

In Fig. 9, we report the results of a complexity analysis of Perpetua* versus Perpetua. Perpetua* is more computationally

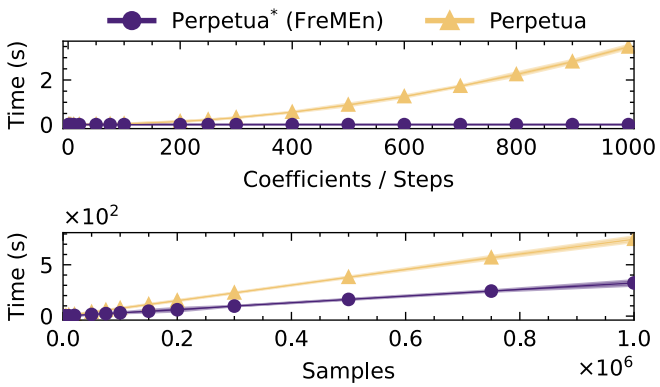

Fig. 9. **Computational Complexity:** prediction (top) and update (bottom) steps for Perpetua with a FreMEn prior versus Perpetua. Top: prediction time as a function of Fourier coefficients in Perpetua and simulation steps in Perpetua. Bottom: update time with up to 1 million samples. Results are averaged over five random seeds. Perpetua exhibits faster computation in both cases by replacing Perpetua's state machine with our Bayesian model selection mechanism.

T BLE II
MEMORY FOOTPRINT OF PERPETU

	Quantity	Value
(a) Per mixture component	Base cost	27 89 KB
	Cost per component	47 KB
(b) Per edge	Base cost	KB
	Cost per edge	28 13 KB

(a) Per-edge cost as a function of mixture components. (b) Total cost as a function of edges using mixture models with five mixture components and a FreMEn prior with 1,000 coefficients.

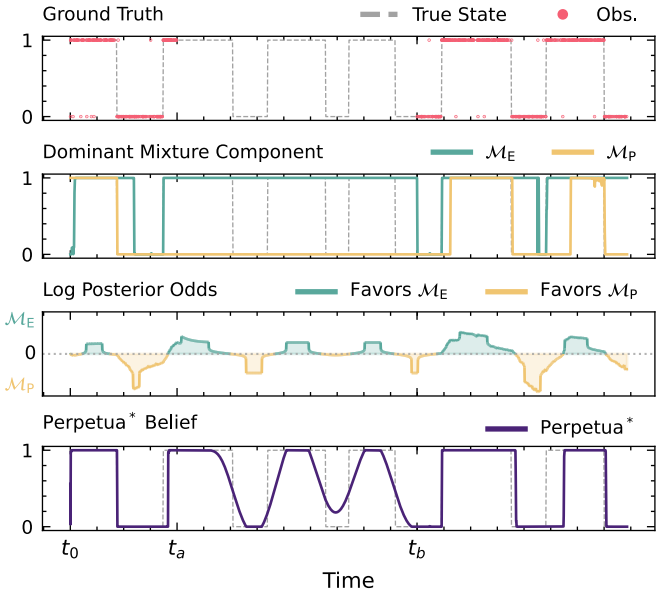

Fig. 10. **Illustrative Example Perpetua** : n object undergoes semi-static changes at varying rates (top row). We maintain a mixture of persistence and emergence filters (each comprising three components in this example) that *adapt* their belief as new observations arrive. The second row shows the dominant component of the persistence and emergence filters (see (7)) at each time step. s time progresses, our Bayesian model selection criterion (see (9)) identifies the mixture that best explains the observed data (third row). Notably, between times t and t_b, where no observations are available, the log posterior odds keep evolving due to our switching prior, whereas the dominant mixture component remains frozen. The bottom row shows the final Perpetua prediction, obtained by mixing the output of both mixture models at each time step, as shown in (10). By leveraging the switching prior (cf. Fig. 2 and Fig. 4), Perpetua maintains accurate predictions even in the absence of observations.

efficient at both prediction and belief updates due to its switching mechanism based on Bayesian model selection, which avoids Perpetua's reliance on expensive simulation steps.

Table II shows the memory footprint of adding extra mixture components or edges to the graph. The per-mixture cost is negligible (0.047 KB), while the per-edge cost is 28 KB for a model with five mixture components and a FreMEn prior with a thousand coefficients. New edges are only added when a new object-receptacle pair is observed, so memory scales linearly with the number of discovered relationships rather than with deployment time. For example, in a scene with 250 objects appearing in up to 4 different locations each (1,000 edges), the total overhead is **28 MB** on top of the base scene graph, which can itself reach several GB due to stored object point clouds. Overall, the additional memory and compute overhead of Perpetua* is negligible compared to modern mapping approaches. Furthermore, in Fig. 10 we present an illustrative example showing how the different components (mixture models, Bayesian model selection) interact to yield the final Perpetua* prediction even in the absence of data.

PPENDIX D
SIMUL TION DET ILS: PREDICTIVEGR PHS

D.1 Hierarchical Scheduling Model

The spatial distribution of each pickupable object over time is governed by a hierarchical temporal model inspired by human activity patterns. For each pickupable, we generate a schedule that specifies which receptacle contains that object at each timestep. This schedule uses a hierarchical structure with different temporal scales (e.g., year → month → week → day → hour) and different divisions for each hierarchical level. Transitions between hierarchical levels and receptacles follow a Markov process. For instance, a schedule whose hierarchical levels are day and hour, with 5 different day-level patterns, will have a one day-level transition matrix and 5 different hour-level transition matrices. Following Kurenkov et al. [21], the transition matrices are defined using weights computed from the relationship between "pickupable" and "receptacle" objects defined in ProcTHOR. Duration in each receptacle is sampled from a uniform distribution, also informed by the ProcTHOR semantic prior. This hierarchical structure allows us to capture realistic patterns: objects stay in certain locations for hours or days, but may transition to different receptacles across longer timescales.[3]. Figure 11 shows a two-week generated schedule. The first week shows standard train-time dynamics, while the second week exhibits shifted test-time dynamics, as used during our adaptation experiments in Sec. VI-B4.

[3]The source code will be made publicly available soon.

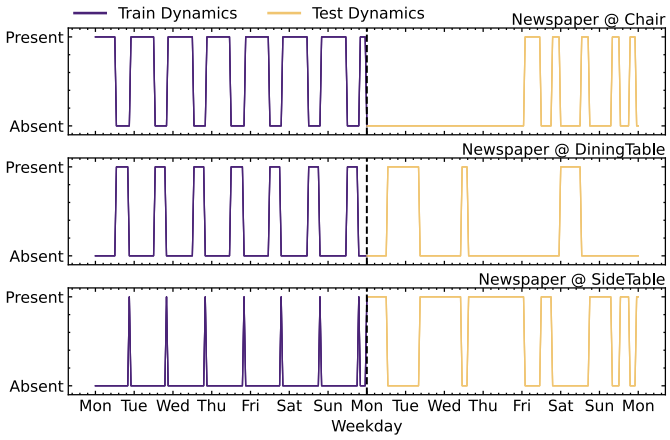

Fig. 11. **Dynamic Shift in ProcTHOR daptation Experiments:** Illustration of the train and test time dynamics for the adaptation experiments in Sec.VI-B4. The first week displays standard training dynamics for the "newspaper" object across three receptacles. In the second week, these dynamics change, introducing a shift that challenges predictive methods, requiring approaches capable of real-time adaptation such as Perpetua .

D.2 Temporal Resolution

Observations are collected at a fixed temporal δt of 1 hour. We generate continuous trajectory data by simulating 4 weeks of time per environment, yielding a total of 672 timesteps per environment (partitioned into two weeks for training, one for validation, and one for testing). We use 15 different ProcThor scenes, which we select to have at least 5 rooms each and a navigable space between $100\,\mathrm{m}^2$ and $150\,\mathrm{m}^2$. top-down view of the scenes is shown in Fig. 16.

D.3 Relationship Between Groundtruth and Privileged Data

The groundtruth assignment indicates the simulator's "true" state (the receptacle that contains the object), while the privileged information filters this through the current observability constraints imposed by the camera viewpoint and rendering. Formally:

$$\text{privileged}_{s,r,t} = \begin{cases} 1 & \text{if } \text{gt}_{s,r,t} = 1 \wedge \text{obj. visible}, \\ 0 & \text{if } \text{gt}_{s,r,t} = 0 \wedge \text{rec. visible}, \\ 1 & \text{if rec. hidden}, \\ 2 & \text{if } r \text{ invalid for } p; \end{cases} \tag{12}$$

where `1` denotes presence, `0` absence, `-1` missing data, and `-2` that the receptacle r is invalid for object s.

PPENDIX E
RE L-WORLD DET ILS: PREDICTIVEGR PHS

E.1 Real-World Details

For our real-world evaluation, we construct a dataset where we manually relocate objects according to a known ground-truth schedule. We deployed an gilex Ranger Mini 2.0 robot equipped with an Intel RealSense D435 camera and teleoperated it through multiple trajectories to gather RGB-D observations. RT B-Map [34] was used as the pose estimation backend and mapping tool. To evaluate PredictiveGraphs under realistic perception noise, we generated object-receptacle training labels for Perpetua* using a custom perception pipeline based on S M3 [37] and the association function described in Sec.V- .

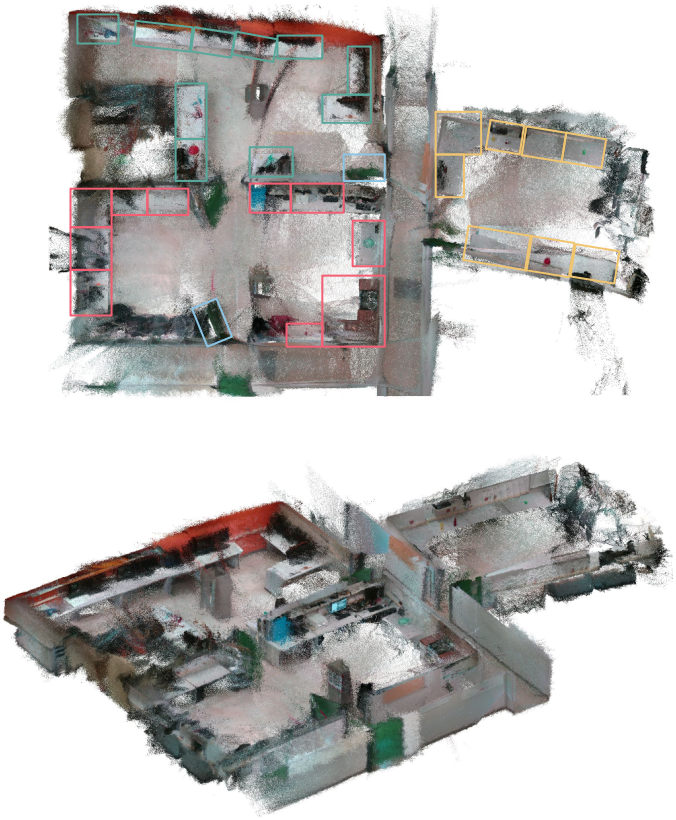

Fig. 12. **Real-World Environment**: (Top) Top-down view showing receptacle and door bounding boxes. Receptacles are color-coded by room: pink (first room), teal (second room), and mustard (third room). Door bounding boxes are highlighted in blue. (Bottom) n isometric view of the three-room laboratory setup used for all real-world experiments.

The environment consists of a three-room laboratory connected by two doors (Fig. 12), containing a total of 28 receptacles distributed as 10, 10, and 8 across the rooms, respectively. Throughout the experiments, the two doors were periodically opened and closed to vary the environment's topological connectivity. Scans were taken bi-hourly from 9:00 M to 5:00 PM over a span of three weeks, yielding a dataset of 84 scans. For each scan, we provide RGB-D frames and camera poses in a common origin frame, along with manually annotated 3D bounding boxes for the receptacles.

During data collection, we used the following 21 objects: `greenbowl`, `bell`, `panda`, `penguin`, `redbowl`, `bluecup`, `yellowcup`, `greencube`, `bluecube`, `headphones`, `duck`, `egg`, `carrot`, `hairbrush`, `redbottle`, `redmug`, `redplate`, `woodenspoon`, `spoon`, `spatula`, `blueball`.

E.2 Hardware

ll simulation experiments were conducted on an institutional cluster equipped with Nvidia RTX 8000 and L40S

T BLE III
D PTIVE N VIG TION EXPERIMENT: DET ILED VERSION OF FIG 6 (LEFT).

		Static Setting (80)		**Dynamic Setting** (45)				**Overall** (125)	
		Presence (80)		Presence (38)		bsence (7)		verage	
Perception	Method	Success (%) ↑	SPL (%) ↑	Success (%) ↑	SPL (%) ↑	R (%) ↑	Dist (m) ↓	Success (%) ↑	SPL (%) ↑
Privileged	CG [15]	52 ±9.9	51 6±9.8	±0.0	±0.0	66 7±19.2	97±0.53	37 6±8.1	36 3±8.4
	HOMER	63 3±2.5	62 7±2.6	25 7±10.3	24 9±10.3	66 7±33.3	6 ±2.0	53 6±2.4	5 8±2.1
	FlowMaps	61 3±6.9	59 3±6.9	2 3±7.3	19 ±7.0	88 9±11.1	4 3±2.2	5 4±3.2	46 7±2.3
	Ours (FreMEn [11])	79 3±2.6	78 1±3.0	53 ±13.4	48 8±11.9	44 4±29.4	9 3±2.7	72 ±3.6	7 2±2.8
	Ours	**89.3**±2.7	**83.4**±1.3	**65.4**±10.4	**56.3**±9.9	**88.9**±11.1	**7.7**±2.3	**83.2**±4.3	**75.9**±2.9
CLIP + LLM	CG [15]	47 4±8.5	47 ±8.4	±0.0	±0.0	66 7±19.2	97±0.53	34 4±7.4	33 1±7.7
	HOMER	46 2±4.8	45 7±4.8	16 4±4.4	16 4±4.4	66 7±33.3	7 2±3.0	39 2±4.3	36 3±3.0
	FlowMaps	4 5±7.0	39 4±6.9	9 9±2.8	9 4±2.7	88 9±11.1	6 8±3.7	32 8±5.1	29 3±3.8
	Ours (FreMEn [11])	57 1±4.8	55 9±4.9	37 2±9.4	36 2±8.7	44 4±29.4	1 9±3.8	52 ±5.7	5 7±5.2
	Ours	**58.5**±3.6	**57.3**±3.8	**38.6**±8.7	**37.2**±8.1	**100.0**±5.1	**9.8**±4.3	**55.4**±5.1	**51.6**±4.4

The adaptive capabilities of Perpetua enable better performance even under distributional shifts in environment dynamics. While perception errors degrade performance by introducing spurious belief updates, Perpetua maintains its overall advantage due to its adaptation capabilities.

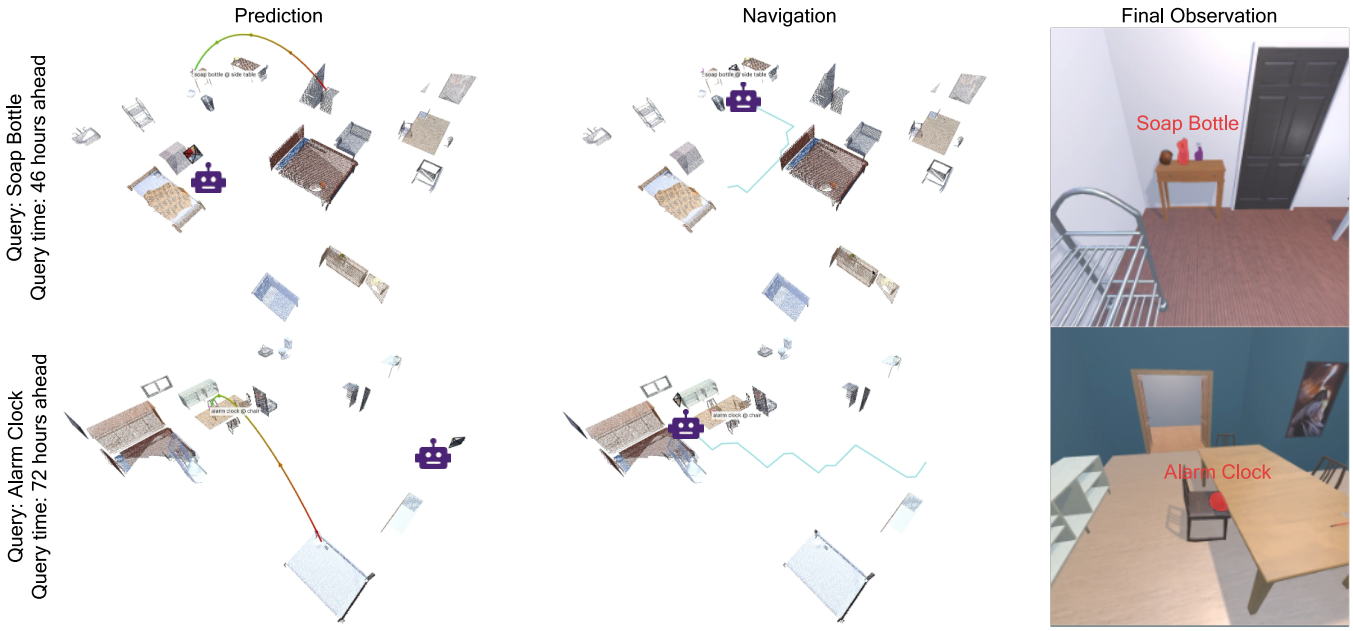

Fig. 13. **Qualitative Navigation Results in ProcTHOR:** The top row shows a navigation sequence where PredictiveGraphs is queried 46 hours after the last map update (top-down map shown in the first row, third column of Fig. 16). In the first column, our method predicts that the `soap bottle` has moved from the dresser (red spline tip) to a side table (green spline tip). The second column shows the robot navigating to the predicted location, with its path marked in blue. The final column displays the successful detection of the soap bottle. The bottom row depicts a similar example with an `alarm clock` queried 72 hours after the last map update (top-down map shown in the third row, first column of Fig. 16).

T BLE IV
RE L-WORLD N VIG TION: PREDICTIVEGR PHS EN BLES ROBUST DYN MIC N VIG TION MIDST PERCEPTION ERRORS, EVEN WHEN TR INED ON NOISY RE L-WORLD D T .

	Static (40)	**Dynamic** (40)	**bsent** (20)	**Overall** (100)
Estimator	SR (%)	SR (%)	R (%)	SR (%)
Ours (noisy oracle)	85	77 5	1	85
CG [15]	**77 5**		5	41
DualMap [27]	62 5	35	35	46
HOMER	45	6	65	55
Ours	7	**65**	**75**	**69**

GPUs. The gilex Ranger Mini 2.0 was equipped with an Nvidia Jetson Orin for onboard control, but model inference was performed offboard, on an Nvidia RTX 2080. For both the LLM-based navigation agent and the LLM-based predictive baselines we use `gpt-5.2`.

PPENDIX F
EXTENDED RE L-WORLD RESULTS

F.1 Simulation results

Table III presents the comprehensive adaptive navigation results, expanding on the results provided in Sec. VI-B4. Here, we ablate PredictiveGraphs by comparing the fully-predictive FreMEn estimator against our adaptive Perpetua* method. Under privileged perception, PredictiveGraphs with

T BLE V
PREDICTIVE N VIG TION RESULTS

Perception	Method	**Static** - Presence (157)		**Dynamic** - Presence (75)		**Dynamic** - bsence (18)	**Overall** (250)	
		Success (%) ↑	SPL (%) ↑	Success (%) ↑	SPL (%) ↑	R (%) ↑	Success (%) ↑	SPL (%) ↑
Privileged	CG [15]	51 4±7.0	49 3±6.6	±0.0	±0.0	46 4±16.3	35 2±4.8	33 2±4.9
	CG + LLM	88 1±3.7	81 1±3.5	68 5±6.3	44 7±6.5	2 2±10.6	76 4±3.3	68 4±3.3
	CG + LLM + Hist.	94 4±2.2	88 5±2.5	65 6±4.8	41 7±5.3	14 3±9.9	8 ±3.0	73 2±3.7
	HOMER	89 4±4.1	83 9±4.6	86 1±5.5	69 1±7.8	58 3±17.5	87 1±2.9	8 4±3.6
	FlowMaps	9 2±4.0	85 ±5.7	61 ±5.3	54 1±5.2	42 9±17.4	78 3±3.3	75 9±4.2
	Ours (FreMEn [11])	**97.9**±1.1	**93.5**±1.3	**92.8**±3.1	**81.5**±4.1	**90.5**±6.1	**95.2**±1.4	**88.9**±2.3
	Ours	**97.9**±1.1	**93.5**±1.3	**92.8**±3.1	81 1±3.9	**90.5**±6.1	**95.2**±1.4	**88.9**±2.3
CLIP + LLM	CG [15]	43 ±6.7	41 ±6.4	±0.0	±0.0	46 4±16.3	29 6±4.5	27 4±4.6
	CG + LLM	63 5±4.9	56 9±4.6	55 6±6.5	35 2±4.9	16 7±10.9	57 2±4.5	49 7±4.0
	CG + LLM + Hist.	65 3±5.6	61 6±5.7	51 2±5.5	3 5±4.1	4 5±15.4	58 4±4.5	51 4±4.9
	HOMER	61 9±5.0	59 6±5.3	71 7±6.3	57 4±6.8	58 3±17.5	65 1±2.9	59 7±3.0
	FlowMaps	62 7±6.9	59 9±7.0	5 ±6.3	45 ±6.6	42 9±17.4	57 5±4.7	55 2±5.4
	Ours (FreMEn [11])	**68.4**±4.0	**65.2**±4.4	73 8±7.6	65 3±6.5	**90.5**±6.1	**71.6**±2.6	65 ±3.7
	Ours	**68.4**±4.0	**65.2**±4.4	**76.2**±8.3	**67.0**±7.0	**90.5**±6.1	**71.6**±2.9	**65.1**±3.1

PredictiveGraphs with Perpetua or FreMEn estimators outperforms baselines under both privileged and noisy perception, including new HOMER and FlowMaps baselines.

Perpetua* achieves superior navigation performance, effectively adapting to the shift in test-time dynamics. Even when perception errors degrade performance, particularly in the dynamic setting, Perpetua* maintains its advantage over the FreMEn variant. s demonstrated by the privileged results, we expect that as perception pipelines improve, the performance of PredictiveGraphs will scale accordingly.

dditionally, Fig. 13 presents qualitative results for the ProcTHOR navigation experiments from Sec. VI-B4. This figure shows how PredictiveGraphs accurately predicts the location of target objects at different query times. This capability allows the agent to exploit predictive information to successfully navigate to its target, even when the underlying scene graph has not been updated for several hours.

In Table V we provide the same results shown in Fig. 5 in table form.

F.2 Real-World results

In this section, we provide additional qualitative results to supplement the real-world navigation experiments discussed in Sec. VI-B5. Figure 14 presents an extended version of Fig. 7, demonstrating the gilex Ranger Mini 2.0 successfully navigating to its target, a hair brush. By anticipating that the shortest path is blocked, our method proactively plans a longer, yet feasible, alternative route to circumvent the obstacle.

Similarly, Fig. 15 illustrates a search for a "healthy vegetable" where the initial predicted location is empty. Instead of relying on the outdated map, PredictiveGraphs correctly predicts a new location for the object, allowing the agent to recover and successfully complete the task.

These experiments are also shown in the first section of the video that we include as part of our supplementary material.

In Table IV we provide the same results shown in Fig. 6 in table form.

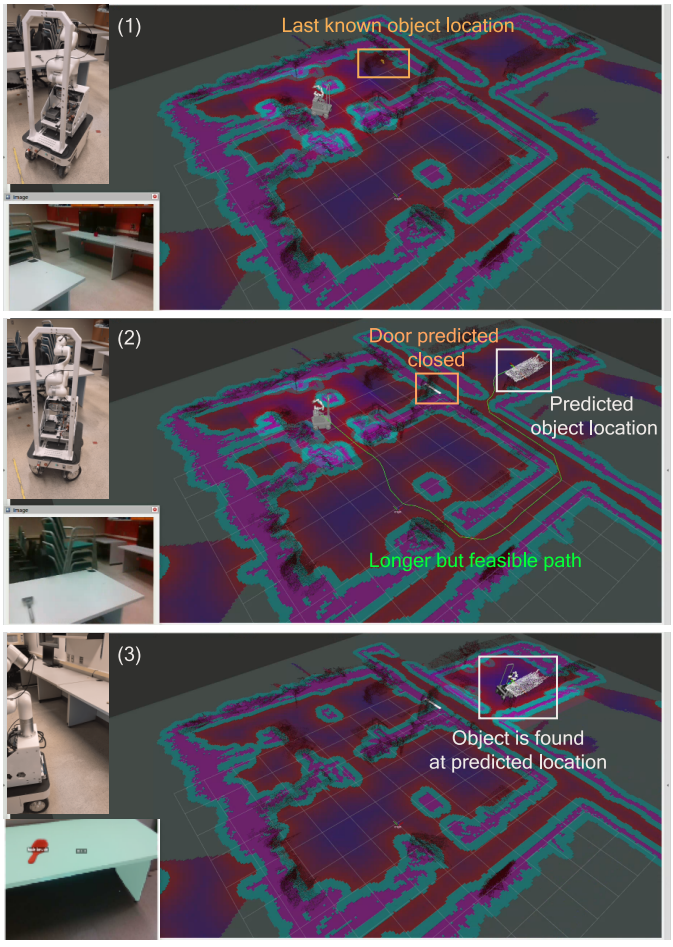

Fig. 14. **Qualitative Navigation Result in Real-World - Topological Graph Change:** Extended version of Fig. 7. The open-vocabulary query is "`Is there something useful to comb my hair?`", with query time 146 hours after the last map update. (1) Shows the last known position of the relevant object identified by PredictiveGraphs (a hair brush). (2) Our method predicts that the hair brush is in another room and detects that the door along the shortest path is closed; this updates the cost map, prompting the planner to generate a longer but feasible path. (3) The agent successfully locates the target object.

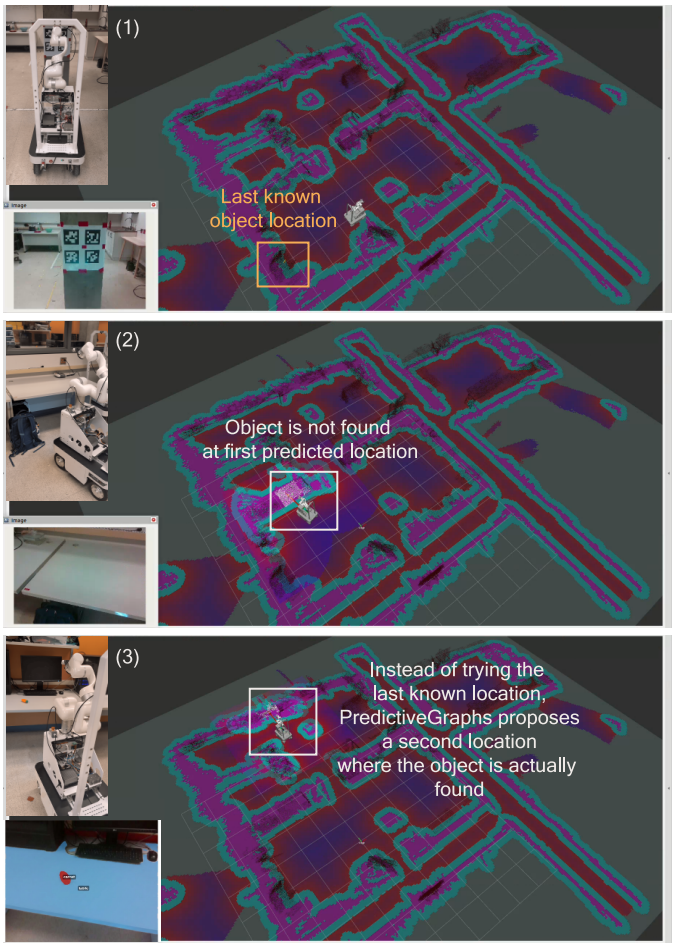

Fig. 15. **Qualitative Navigation Result in Real-World - Replanning:** Open-vocabulary query " `ould you get me a healthy vegetable?`" with query time 29 hours after the last map update. (1) PredictiveGraphs identifies the target object as a carrot. (2) PredictiveGraphs proposes an initial candidate location; however, upon navigating to it, the object is not found. (3) Instead of reverting to the object's last known location, PredictiveGraphs predicts a new location where the carrot is successfully found, highlighting the robustness of our tempo-spatio-semantic scene graph in dynamic scenes.

Fig. 16. **ProcTHOR environments:** Top-down view of the different 15 ProcTHOR scenes used in the simulation experiments in Sec.VI- 2. Each scene has at least 5 rooms and a navigable space between 1 m and 15 m .