FlowMaps: Modeling Long-Term Multimodal Object Dynamics with Flow Matching

Francesco Argenziano[1] **Miguel Saavedra-Ruiz**[2,3] **Sacha Morin**[2,3]
Charlie Gauthier[2,3] **Daniele Nardi**[1] **Liam Paull**[2,3]

[1]Sapienza University of Rome, Rome, Italy
[2]Université de Montréal, Montréal, QC, Canada
[3]Mila - Quebec AI Institute, Montréal, QC, Canada

{argenziano,nardi}@diag.uniroma1.it
{miguel-angel.saavedra-ruiz,sacha.morin}@mila.quebec
{charlie.gauthier,paulll}@mila.quebec

Abstract: Joint spatial and temporal understanding of 3D scenes is a crucial requirement for robots deployed in everyday household environments. Such agents must not only comprehend and navigate spatial layouts, but also reason about how these spaces evolve over time. In particular, humans interact with objects daily, causing them to change position throughout the environment and making it difficult for robots to reliably associate current observations with previously seen objects. However, these interactions are not random: human habits and routines induce spatio-temporally consistent patterns in object locations, which robotic agents can potentially learn and then exploit for downstream tasks such as navigation. To this end, we introduce `FlowMaps`, a latent flow matching model for estimating multimodal distributions over the future locations of dynamic objects in a continuous 3D space. By learning the implicit dependencies among objects and their temporal evolution, `FlowMaps` predicts likely changes in object locations conditioned on past human interactions, while supporting generalization across previously unseen environments that share similar object routines. To demonstrate the utility of this method, we deploy `FlowMaps` in a downstream dynamic Object Navigation task in both simulated and real-world environments. Across more than 600 episodes, `FlowMaps` outperforms state-of-the-art approaches, showing that modeling object dynamics through continuous, multimodal spatio-temporal distributions improves robotic search and navigation in changing household environments. Code and additional material is available at https://fra-tsuna.github.io/flowmaps/.

Keywords: Flow matching, Dynamic environments, Object navigation

1 Introduction

Environments inhabited by humans are intrinsically dynamic. This dynamism should not be attributed solely to the people living in them, but also to the everyday objects they contain. Through repeated human interactions, objects are frequently moved, displaced, and relocated over time, posing major challenges for robots expected to deliberate and act in such environments [1, 2]. Under these conditions, even a seemingly simple task (*e.g.,* "help me find my glasses" [3]) becomes highly challenging: although the robot may have observed an object in the past, there is no guarantee that it will still be found at its previous location, and reasoning about its possible displacement requires considering where it may have moved in physical space. Humans constantly move objects within and across rooms, creating significant difficulties for navigation and retrieval. Nevertheless, humans tend to follow repetitive behavioral patterns [4]. For instance, a pair of glasses may be placed on

the nightstand before sleep, moved to the bathroom sink before a shower, and later returned to the nightstand. These routines are referred to as *semantically consistent patterns* [5, 6], and can potentially be leveraged to recover from failed retrieval attempts by directing the robot toward likely human-induced object placements.

We hypothesize that, given sufficient observations of how objects move through environments over time, a generative model can learn latent regularities induced by human routines and exploit them to predict likely future object placements. Crucially, our key insight is that predicting where objects will be found does not require explicitly identifying the human activity that caused their displacement; instead, such structure can emerge directly from data. To be useful beyond a single observed home, these predictions should generalize to environments with different layouts and object arrangements, while remaining grounded in actual object locations rather than fixed, predefined alternatives such as receptacle labels. Since object displacements are inherently multimodal (*i.e.,* they admit multiple *modes*, each corresponding to a distinct plausible placement), modeling them requires a distributional approach over continuous space.

To this end, we present `FlowMaps`, a latent flow matching (FM) model that recovers multimodal spatio-temporal distributions of common household objects directly in continuous 3D space. Unlike prior dynamic object location models that primarily operate over discrete receptacle-level relations, `FlowMaps` models human-induced household object relocalization as a multimodal distribution in continuous 3D space. `FlowMaps` is composed of two main modules: *(i)* a Variational Autoencoder (VAE) that learns latent representations of object geometry and semantics, and *(ii)* a latent Diffusion Transformer (DiT) [7] that predicts likely object locations over time. To obtain training data at scale, we employ ProcTHOR [8] to generate meaningful dynamic object trajectories across procedurally generated household environments. We then demonstrate the usefulness of the learned prior on Object Navigation (ObjNav), using it as a representative downstream robotic task among the broader set of applications that can benefit from predictions of likely object placements. In this setting, `FlowMaps` is trained on dynamic trajectories from a set of training environments, while ObjNav performance is evaluated in disjoint, previously unseen homes, explicitly testing whether the learned prior transfers beyond the scenes observed during training. Across more than 600 ObjNav episodes in ProcTHOR, we compare `FlowMaps` against state-of-the-art approaches and show superior performance in retrieving target objects. We further validate our approach on a real robotic platform, showing how it can be deployed in a real-world setting.

Contributions statement. Our contributions are three-fold: ***(i)*** `FlowMaps`, a FM-based architecture for representing and predicting continuous, multimodal spatio-temporal object distributions in dynamic environments; ***(ii)*** as an exemplary downstream robotic application, we demonstrate the use of `FlowMaps` for ObjNav in dynamic scenes, enabling a robot to reason about where target objects are likely to be found over time, while outperforming both zero-shot baselines and trained expert policies; ***(iii)*** an extensive quantitative and qualitative evaluation of both the distributional properties and practical applicability of `FlowMaps`, covering more than 600 simulated ObjNav episodes in disjoint environments together with real-world deployments, and demonstrating the soundness, effectiveness, cross-environment generalization, and practical viability of the proposed approach.

2 Related Work

Learning human habits and patterns. Learning human habits enables robots to move from reactive command execution to proactive assistance [9, 10]. Prior work models human activities for anticipation [11, 12], uses interaction histories for personalized collaboration [13], and studies anticipatory robot assistance [14, 15, 16]. The closest work to ours is HOMER [17], which uses a Graph Neural Network (GNN) to predict object displacements from recurring household activities. Unlike HOMER, which trains a separate model per household and mainly evaluates within-environment generalization, we train and test on disjoint environments sharing semantically consistent patterns, assessing generalization to unseen household layouts.

Object navigation in dynamic environments. ObjNav aims to locate target objects in known or unknown environments. Existing methods often rely on Large Language Models (LLMs) or Visual Language Models (VLMs) for zero-shot reasoning [18, 19], or learn end-to-end policies [20], but typically assume static scenes. Recent dynamic approaches incorporate object trajectories [21, 18], probabilistic object-location estimates [3, 22], or dynamic scene graph memories [23], yet they depend on costly VLM reasoning, hand-designed priors, online estimation, LLM-interpreted activity hints, or explicit graph structures, and have often been evaluated in limited settings. In contrast, `FlowMaps` models object dynamics as continuous distributions over future object placements in 3D, rather than discrete temporal link prediction over object-location relations [17, 23]. It requires no explicit scene graph structure at training or inference time, remains efficient through its FM formulation, and outperforms prior dynamic ObjNav methods.

Flow matching in robotics. Generative models are increasingly used in robotics, supported by large cross-embodiment datasets such as Open-X Embodiment [24]. Recent work uses diffusion models [25, 26], FM [27, 28], and vision-language-action models [29, 30] mainly for policy learning. In contrast, we use FM for posterior inference, recovering spatio-temporal and multimodal distributions over plausible object placements rather than generating actions. To the best of our knowledge, this is the first use of FM for posterior inference in this setting.

3 Background and problem formulation

Vector fields and flows. An Ordinary Differential Equation (ODE) is defined by a time-dependent *vector field* $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$, which assigns a velocity $u_t(x)$ to every position $x \in \mathbb{R}^d$ at each time $t \in [0,1]$. A solution of the ODE is a *trajectory* $X : [0,1] \to \mathbb{R}^d$, where X_t denotes the position of the system at time t. In this work, we are particularly interested in *flows* as solutions of the ODE. A flow is a time-dependent map $\psi : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$, written $\psi_t(x) = \psi(x,t)$, whose evolution is governed by

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)), \qquad \psi_0(x_0) = x_0.$$

The vector field u_t is said to generate a *probability path* $(p_t)_{0\le t\le 1}$ if the corresponding flow transports samples from the initial distribution p_0 to the distribution p_t at each time t. Equivalently, for $X_0 \sim p_0$, we have $X_t = \psi_t(X_0) \sim p_t$.

Conditional flow matching. Conditional flow matching (CFM) learns such a vector field by regressing onto tractable conditional velocities. The goal is to learn u_t^θ whose flow transports a simple base distribution p_{init} to the data distribution p_{data}. Instead of directly constructing the marginal vector field, namely the vector field that generates the full probability path from p_{init} to p_{data}, CFM defines conditional paths $p_t(\cdot \mid y)$ indexed by data samples $y \sim p_{data}$, together with tractable target velocities $u_t^{target}(\cdot \mid y)$. The training objective is

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,x,y}\left[\left\| u_t^\theta(x) - u_t^{\text{target}}(x \mid y)\right\|_2^2\right], \quad t \sim U[0,1],\ y \sim p_{data},\ x \sim p_t(\cdot \mid y). \tag{1}$$

Under standard assumptions, this objective has the same gradient as the corresponding marginal FM objective, so minimizing (1) learns the marginal vector field without requiring its explicit evaluation. For a more complete explanation and formalization, we refer the reader to [31, 32].

Latent flow matching. In *latent flow matching*, the same learning objective is applied in a latent representation space. Given an encoder E and decoder D, data samples $x \sim p_{data}$ are mapped to latents $z = E(x)$, inducing a latent data distribution p_{data}^z. A vector field u_t^θ is then trained to transport samples from a simple latent prior p_{init}^z to p_{data}^z by minimizing the CFM objective in latent space. Sampling draws $Z_0 \sim p_{init}^z$, integrates $\frac{d}{dt}Z_t = u_t^\theta(Z_t)$, and decodes the terminal latent variable as $\hat{x} = D(Z_1)$. This reduces the dimensionality of the transport problem and lets the flow operate on compact, semantically structured representations, while the decoder maps generated latents back to the data domain.

Problem formulation. At time $\tau \geq 0$, we represent the map as $M_\tau = (O_\tau, O_{\text{BG}})$, where $O_\tau = \{O_{i,\tau}\}_{i=1}^{N_O}$ denotes the set of dynamic objects observed in the scene, and O_{BG} denotes the static background, which is assumed to remain unchanged over time. Each dynamic object is represented as $O_{i,\tau} = (\mathbf{b}_{i,\tau}, l_i)$, where $\mathbf{b}_{i,\tau} \in \mathbb{R}^6$ denotes the object's 3D axis-aligned bounding box in center-size format, $\mathbf{b}_{i,\tau} = (c^x_{i,\tau}, c^y_{i,\tau}, c^z_{i,\tau}, s^x_i, s^y_i, s^z_i)$, with $(c^x_{i,\tau}, c^y_{i,\tau}, c^z_{i,\tau})$ denoting the box center at time τ, and (s^x_i, s^y_i, s^z_i) denoting its spatial extent. The label $l_i \in \mathcal{L}$ denotes the semantic class of the object. In each scene, there is at most one dynamic object per semantic class, so no two dynamic objects share the same label. Background elements in O_{BG} are represented analogously, but their spatial states are fixed across time. Given a prediction horizon $\Delta\tau \geq 0$ and an object query label l_q identifying a dynamic object in O_τ, our goal is to infer the distribution over future bounding boxes of the queried object at time $\tau_f = \tau + \Delta\tau$. Namely, we want to infer the distribution $p\left(\mathbf{b}_{q,\tau_f} \mid M_\tau, l_q, \tau_f\right)$.

4 FlowMaps

Directly computing this posterior is intractable, as the queried object's future state depends on rich scene context and may admit multiple plausible outcomes. We approximate it with FM, learning a conditional transport from a Gaussian base distribution $p_0 = \mathcal{N}(0, I)$ to the target distribution

$$p_1 = p\left(\mathbf{b}_{q,\tau_f} \mid M_\tau, l_q, \tau_f\right).$$

Starting from $z_{\text{init}} \sim p_0$, integrating the learned flow produces a sample $z_{\text{final}} \sim p_1$, corresponding to a plausible future bounding box of the queried object at time τ_f.

We instantiate this conditional flow in latent space with `FlowMaps`, a Transformer-based FM model built on the Conditional Diffusion Transformer (CDiT) of Bar et al. [33]. A VAE first encodes object bounding boxes and semantic labels into a latent target space (Section 4.2). A CFM network then learns the flow in this space, predicting future object placements conditioned on the current scene, queried label, and prediction horizon (Section 4.3). Dynamic scene generation and data collection are described in Section 4.1. Architectural, hyperparameter, solver, and interpolation details are provided in the supplementary material.

4.1 Dynamic Scenes Generation and Data Collection

We used ProcTHOR [8] to generate dynamic indoor environments for training both components of `FlowMaps`. Object movements are driven by predefined human-like routines that produce semantically consistent patterns. We instantiated three representative routines, each capturing a different type of plausible indoor behavior. In Habit #1, the simulated human exhibits location preferences, repeatedly returning to a small set of favored places and therefore spending more time there. In Habit #2, the simulated human follows a balanced routine, distributing time approximately uniformly across the relevant locations in the environment. In Habit #3, the simulated human follows a highly dynamic routine, frequently transitioning between locations and spending only short intervals at each one. For each routine, we generate 2706 training and 918 validation environments. Each scene contains up to 15 dynamic objects moving between semantically compatible receptacles, *e.g.,* a "Fork" can appear on a "Sink" or "DiningTable", but not on a "ShelvingUnit". Each environment is simulated for 4 weeks with hourly resolution $d\tau$, yielding 672 timestamps per scene s, with $s_\tau = [\tau, \mathcal{O}\tau, \mathcal{O}BG]$. This results in over 1.8M training samples per habit. For each habit, we train and evaluate a separate `FlowMaps` model. More information on habits can be found in the appendix.

4.2 Variational Autoencoder

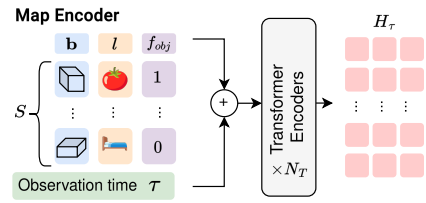
We train a VAE to map individual object tokens into the latent space. Each token consists of a normalized 3D bounding box $\mathbf{b}$ and a semantic label l, and is encoded into a latent code $\mathbf{z} \in \mathbb{R}^{d_z}$. Once trained, the VAE is frozen: its encoder e_ϕ provides the target latents for the CDiT, while its decoder d_ξ maps generated latents back to object predictions. The decoder is trained to predict both geometry and semantics: its geometry head reconstructs the normalized bounding box parameters $\hat{\mathbf{b}}$,

while its semantic head predicts logits over object classes. The network is trained with the standard VAE objective $\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{rec}} + \beta_t \mathcal{L}_{\text{KL}}$, where β_t is linearly annealed during training to prevent posterior collapse [34]. The reconstruction term combines geometric and semantic supervision: $\mathcal{L}_{\text{rec}} = \lambda_{\text{CIoU}} \mathcal{L}_{\text{CIoU}} + \lambda_{L_1} \mathcal{L}_{L_1} + \lambda_{\text{CE}} \mathcal{L}_{\text{CE}}$. Here, $\mathcal{L}_{\text{CIoU}}$ and $\mathcal{L}_{L_1}$ supervise the reconstructed bounding box, while $\mathcal{L}_{\text{CE}}$ supervises the semantic class prediction.

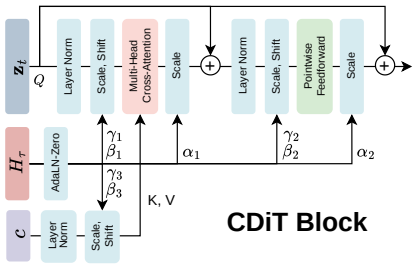
4.3 Latent Flow Matching network

The `FlowMaps` network parametrizes the velocity field u_t^θ that transports a Gaussian latent $\mathbf{z}_0$ to the latent encoding $\mathbf{z}_1 = e_\phi(\mathbf{b}_q, l_q)$ of the queried object's future bounding box. It is composed of: *(i)* a *map encoder* that aggregates the scene context into a sequence of contextualized tokens, and *(ii)* a stack of *CDiT blocks* that iteratively refine the noisy query latent by cross-attending to that context.

Map encoder. At each timestep, the scene is a padded set $\mathcal{M}_\tau$ of at most $N_O + N_{BG}$ object tokens, with maximum length S. Each token represents either a dynamic object or a static background object through a 3D axis-aligned bounding box $\mathbf{b}$, semantic label l, and object-type flag f_{obj}. Following Wald et al. [35], this flag acts as a set-membership indicator, where $f_{obj} = 1$ for dynamic objects and $f_{obj} = 0$ for furniture/background objects. The map encoder (Fig. 1a) maps $\mathcal{M}_\tau$ to context tokens $H_\tau \in \mathbb{R}^{S \times d_h}$ by summing bounding box, class, type, and learned time embeddings, then processing them with N_T pre-norm Transformer encoder layers. Self-attention lets each token incorporate scene-level context and relations to other objects. No positional encoding is used across tokens, making the encoder permutation-invariant: spatial structure is provided only by the bounding box embedding.



(a) Map encoder.



(b) CDiT block.

Figure 1: Overview of the latent FM network components: *(a)* the map encoder, and *(b)* the CDiT block.

CDiT block. Given H_τ, the flow is implemented by N_C CDiT blocks (Fig. 1b) that update the noisy query latent $\mathbf{z}_t \in \mathbb{R}^{d_z}$ by cross-attending to the encoded scene. This CDiT-style design, rather than full-attention DiT [7], reflects the asymmetry of the task: a single query object is transported while the scene provides a variable-length context, avoiding repeated quadratic attention over the full scene at each FM step. Cross-attention and feed-forward layers are modulated with *adaLN-Zero* [7] using a conditioning vector c built from the flow time $t \in [0, 1]$, final timestamp τ_f, and queried label l_q. Since the residual stream contains only one query token, we remove the per-block self-attention of Bar et al. [33]; scene-level reasoning is handled by the map encoder, while the CDiT blocks condition the query trajectory through cross-attention. A final adaLN-modulated linear head maps the refined token to the VAE latent space, producing $u_t^\theta(\mathbf{z}_t \mid H_\tau, l_q, \tau_f)$.

Training and inference. We train the network with the CFM loss of Eq. 1 on a conditional optimal-transport path [36]. Target latents $\mathbf{z}_1$ are obtained by encoding ground-truth future bounding boxes with the frozen VAE encoder e_ϕ and standardizing them with training-set statistics, while source latents $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_z})$ are sampled independently. For $t \sim \mathcal{T}$ on $[0, 1]$, we set $\mathbf{z}_t = a_t \mathbf{z}_1 + b_t \mathbf{z}_0$ and regress u_t^θ to $\dot{\mathbf{z}}_t = \dot{a}_t \mathbf{z}_1 + \dot{b}_t \mathbf{z}_0$ with an L_2 loss. Source-target pairs are matched by an exact mini-batch optimal-transport plan [37, 38], yielding straighter trajectories and fewer integration steps. At inference, given $\mathcal{M}_\tau$ and (l_q, τ_f), one map-encoder pass computes the query-independent H_τ. Since H_τ is query-independent, the same encoder pass is amortized across all dynamic objects and posterior samples. We then integrate $\hat{\mathbf{z}}_t = u_t^\theta(\hat{\mathbf{z}}_t \mid H_\tau, l_q, \tau_f)$ from $\mathbf{z}_0$ to $\hat{\mathbf{z}}_1$ in K fixed steps, de-standardize $\hat{\mathbf{z}}_1$, and decode it with d_ξ into $\hat{\mathbf{b}}_{q,\tau_f}$. Independent $\mathbf{z}_0$ draws produce samples from $p(\mathbf{b}_{q,\tau_f} \mid \mathcal{M}_\tau, l_q, \tau_f)$. The full pipelines are shown in Fig. 2.

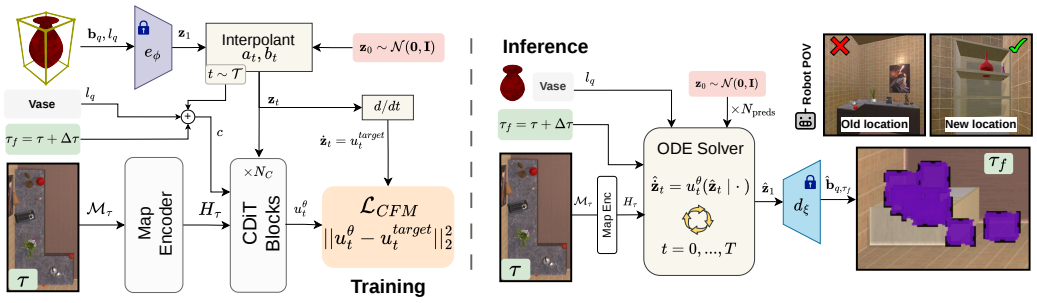


Figure 2: Training (left) and inference (right) pipeline for `FlowMaps`. We train the stack of CDiT blocks to learn the vector field u_t^θ, which is used at inference time to predict $\hat{\mathbf{b}}_{q,\tau_f}$.

5 Experimental Results

Distributional Evaluation. We evaluate `FlowMaps` using the joint accuracy and distributional metrics reported in Table 1. All entries are averaged over the three considered habits.

We report two best-of-K accuracy metrics, following the protocol introduced for multimodal trajectory prediction [39]. For each query, we draw $K{=}50$ samples. *minFDE@K* measures the 3D distance between the ground-truth bounding box center and the closest predicted sample, while *Rec@1* reports the corresponding best-of-K recall within $\delta{=}1m$ on the floor plane. To assess the predicted distribution beyond point accuracy, we also report *coverage* and *density* following Naeem et al. [40]. Coverage is the fraction of ground-truth positions whose k-Nearest Neighbor ball contains at least one generated sample, whereas density is the average number of such balls occupied by each sample. These metrics separate mode coverage from over-concentration, without penalizing valid multimodal predictions. We further compute *TV* and *JS*, respectively the total-variation and Jensen-Shannon divergences between predicted and ground-truth samples discretized as histograms on an xz floor grid with $0.5\,\mathrm{m}$ resolution. These divergences measure how spatial mass is allocated across the scene, independently of best-of-K accuracy.

We compare `FlowMaps` against four baselines. **FreqPrior** estimates a training-set distribution $P(r \mid l_q)$ over receptacle categories r. At test time, given $(\mathcal{M}_\tau, l_q)$, it samples a receptacle category, selects an instance of that category in the scene, and samples a point uniformly from the footprint of its top face. This captures scene-conditional object co-occurrence but ignores temporal structure. **LLMPrior** replaces this empirical distribution with a ranked top-T list of receptacles suggested by an LLM. **EmpiricalMean** is an oracle point predictor that returns the mean of all ground-truth positions of objects with label l_q in the environment, providing a strong upper bound for regression methods with the same $(\mathcal{M}_\tau, l_q)$ conditioning. **MeanFlowMaps** collapses `FlowMaps` samples to their centroid, isolating the effect of multimodality from the conditioning signal. More details on the distributional baselines are provided in the appendix.

`FlowMaps` achieves the best minFDE ($0.342\ m$, 7.6% below the strongest baseline), while nearly saturating mode coverage. It also raises the Naeem density from 0.647 (FreqPrior) to 0.886, a 37% gain, indicating that its samples concentrate more tightly around ground-truth modes. The two single-point baselines are consistently worse on accuracy (minFDE $> 2\ m$, Rec@1 < 0.42); MeanFlowMaps, despite using the same conditioning, is over $7\times$ worse in minFDE, confirming the need for a multimodal generator. FreqPrior matches `FlowMaps` only in best-of-K recall, whereas `FlowMaps` also reduces the error tail, increases density, and lowers both TV and JS, showing that it captures *where* on the plausible surfaces mass concentrates, not just *which* surfaces are plausible.

Object Navigation. We further evaluate `FlowMaps` on ObjNav, our primary downstream application, using a minival split of 25 random validation environments. For each environment, we select 5 query objects and sample 5 (τ, τ_f) pairs per object, obtaining more than 600 ObjNav episodes. An episode is successful if the agent reaches the target within N_{steps} steps, is within distance δ_{dist} of the object's ground-truth position, and observes the object from its point of view. All simulated experiments are conducted in AI2-THOR [41] with ProcTHOR [8] environments, where visibility is

Table 1: Distributional evaluation averaged across the three habits. Density, coverage, TV and JS are distribution-shape metrics and are reported only for methods that emit a non-degenerate sample distribution.

Method	minFDE(m)↓	Rec@1↑	Density↑	Coverage↑	TV↓	JS↓
FreqPrior	0.370	**0.958**	0.647	0.991	0.848	0.514
LLMPrior	0.907	0.780	0.773	0.967	**0.796**	**0.471**
EmpiricalMean	2.040	0.414	—	—	—	—
MeanFlowMaps	2.712	0.260	—	—	—	—
`FlowMaps` (ours)	**0.342**	0.942	**0.886**	**0.999**	0.829	0.482

assessed using ground-truth information. In the real-world lab demo, a VLM is used only to trigger the candidate "*found*" signal, while success is still evaluated using the same three criteria, so false positives do not count as successful episodes.

We compare `FlowMaps` against representative dynamic ObjNav baselines: **TAP-LGX** [21], a zero-shot VLM-based extension of LGX [18]; **OSG** [3] and **CEG** [22], which implement the CP-SAT planning approximation of Rudra et al. [3]; **HOMER** [17] and **SGM** [23], which model dynamic ObjNav as temporal link prediction with GNNs, with SGM additionally exploiting the scene graph hierarchy; and a **Naive LLM** baseline directly prompted to rank future object locations. Since OSG and CEG require object location likelihoods, we follow Wang et al. [22] and evaluate scene-prior (SP) variants based on object-receptacle co-occurrences, as well as LLM-based variants that parse human habit hints into relevant events for posterior estimation. All methods return a ranked list of K candidate future locations; for `FlowMaps`, we obtain this list by running inference N_{preds} times, clustering predictions with DBSCAN [42], and ranking clusters by mass. Additional baseline details are provided in the appendix.

We report Success Rate (SR) and Success weighted by Path Length (SPL) [43], including SR@K, SPL@K, mean SR (mSR), and mean SPL (mSPL) averaged over K. Success at smaller K indicates that the model localizes the object's future position with fewer trials. We also report the mean path length and mean number of steps required to reach the target. Table 2 shows that `FlowMaps` consistently achieves the best mSR and mSPL across all settings, while remaining optimal or competitive on the other metrics. Notably, `FlowMaps`' SR@1 exceeds the SR@5 of some baselines, indicating that its first prediction can be more accurate than several competing top-5 proposal sets. Among the baselines, GNN-based methods perform best on the easier Habit #1 and Habit #2 settings, but are surpassed by zero-shot LLM-based methods on the harder Habit #3. This suggests that harder behavioral patterns benefit more from transferable dynamic exploration than from directly generalizing learned patterns. Overall, `FlowMaps` performs best on all three habits, showing stronger dynamic-pattern learning and cross-environment generalization.

Real-world deployment. We further showcase our proposed approach in a real-world demonstration with a TIAGo robot[1]. The experiment is depicted in Fig. 3: it consists in finding a "CellPhone" that belongs to a person operating in this environment and alternates between the different desks, spending an equal amount of time in each of them (Habit #2). The scene is observed when the phone is at the right desk. A couple of hours pass, and the robot is fetched to find again the phone. Predictions of `FlowMaps` are clustered together and become ranked navigation targets for the agent. Eventually, the phone gets found at second targeted position.

6 Limitations and Future Directions

The current formulation leaves room for further extensions. First, we consider a closed set of 41 object classes and 17 predefined receptacles, which makes `FlowMaps` not directly applicable to open-vocabulary settings. This choice was motivated by our focus on indoor household environments, where relevant object classes and receptacles are relatively constrained. Extending the method to open-vocabulary labels could broaden its applicability to more diverse indoor scenarios, such as offices, laboratories, or factories.

[1] https://pal-robotics.com/robot/tiago/

	Method	SR@1 (%)↑	SR@5 (%)↑	SR@10 (%)↑	mSR (%)↑	SPL@1 (%)↑	SPL@5 (%)↑	SPL@10 (%)↑	mSPL (%)↑	Path (m)↓	Steps↓
Habit #1	Naive LLM	42.13	59.34	63.77	57.44	33.54	38.40	39.03	37.80	12.01	195.1
	TAP-LGX [21]	35.25	57.05	62.62	54.03	28.88	40.03	44.00	43.62	10.62	193.6
	OSG+SP [3]	18.69	56.89	64.10	51.85	15.34	31.46	33.21	29.29	12.99	224.3
	OSG+LLM [3]	42.79	62.46	63.44	60.13	34.65	44.03	44.27	42.90	8.38	131.9
	CEG+SP [22]	22.30	58.03	64.75	52.66	18.16	33.32	34.82	30.98	12.42	212.7
	CEG+LLM [22]	42.95	62.62	63.61	60.29	34.93	44.57	44.81	43.40	**8.00**	**126.3**
	SGM [23]	48.36	69.18	71.80	67.26	38.74	46.52	46.84	45.64	8.82	137.4
	HOMER [17]	46.72	67.54	69.84	64.46	37.31	44.59	44.98	43.62	9.29	145.9
	FlowMaps (ours)	**57.70**	**71.15**	**72.62**	**69.26**	**45.31**	**49.52**	**49.72**	**48.95**	9.01	141.1
Habit #2	Naive LLM	39.02	58.52	60.66	55.46	31.30	37.95	38.21	36.94	8.72	135.2
	TAP-LGX [21]	34.26	54.92	61.64	53.57	28.12	40.73	43.18	39.47	10.98	199.6
	OSG+SP [3]	19.18	57.38	66.07	52.69	15.74	32.03	34.49	29.99	13.37	233.3
	OSG+LLM [3]	36.89	47.38	50.16	46.43	29.85	34.97	35.69	34.40	8.14	129.5
	CEG+SP [22]	22.46	58.36	65.74	53.08	18.31	33.39	35.27	31.17	12.29	212.6
	CEG+LLM [22]	37.38	48.20	50.98	47.25	30.11	35.47	36.17	34.90	8.02	127.5
	SGM [23]	47.54	66.23	68.52	63.85	38.07	44.83	45.14	43.94	7.76	117.6
	HOMER [17]	41.48	**67.38**	**69.18**	62.90	33.59	41.65	41.85	40.37	8.79	135.4
	FlowMaps (ours)	**50.49**	66.56	67.21	**63.92**	**41.18**	**46.51**	**46.59**	**45.72**	**7.55**	**113.5**
Habit #3	Naive LLM	41.15	56.23	61.48	55.46	33.52	38.31	39.21	37.89	9.45	145.1
	TAP-LGX [21]	36.72	57.05	64.10	55.61	29.82	42.30	44.86	40.97	9.37	163.0
	OSG+SP [3]	15.90	56.23	65.25	51.21	12.95	31.00	33.31	28.55	11.92	207.0
	OSG+LLM [3]	38.52	54.26	55.74	52.33	31.72	38.97	39.33	38.04	**7.68**	**115.7**
	CEG+SP [22]	18.52	57.21	64.43	52.21	14.85	31.48	33.30	29.37	11.54	199.2
	CEG+LLM [22]	38.03	54.10	55.74	52.00	31.23	38.59	38.93	37.62	7.91	119.9
	SGM [23]	27.21	62.62	65.41	56.84	22.19	33.64	34.02	31.84	12.24	198.1
	HOMER [17]	38.03	62.95	65.74	58.93	30.29	38.55	38.94	37.25	11.51	186.9
	FlowMaps (ours)	**50.98**	**66.72**	**68.20**	**64.39**	**41.07**	**45.99**	**46.17**	**45.30**	7.90	117.4

Table 2: Results of FlowMaps and the baselines for the dynamic ObjNav episodes. We report metrics grouped by the three distinct habits that induce the objects displacement.

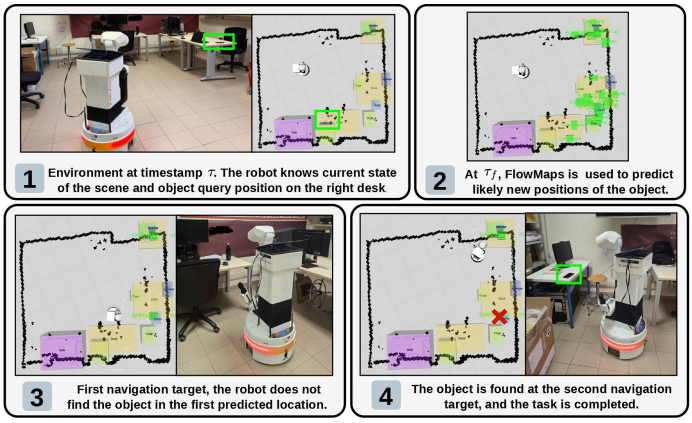


Figure 3: FlowMaps deployed in a real-world environment.

A second limitation concerns the amount of data needed to train the models. In our experiments, ProcTHOR enabled us to generate sufficient training data for the considered setup. However, extending the approach to more general environments would likely require comparable data, possibly without access to the same simulation tools. One direction to mitigate this issue is to reduce reliance on fully offline training. For example, after an appropriate sample-efficiency analysis, a compact offline backbone could be trained and then fine-tuned online using data collected during deployment. Finally, the current evaluation is limited to scenarios with at most three habits. While sufficient to assess the proposed formulation, further analysis is needed to understand how the method scales with the number and diversity of habits. These limitations will be investigated and addressed in future work.

7 Conclusions

In this paper, we introduced FlowMaps, a latent FM framework for modeling multimodal spatio-temporal distributions of dynamic objects in human-inhabited environments. Rather than predicting discrete receptacle-level states, FlowMaps models object relocalization as a distribution over future bounding boxes in continuous 3D space. Conditioned on the current scene, queried object, and prediction horizon, it learns recurring human-driven motion patterns directly from data, without explicit habit or activity labels. By combining a VAE-based latent object representation with a CDiT flow model, FlowMaps remains grounded in object geometry while generalizing to unseen household

layouts. We evaluated both its distributional properties and practical utility in dynamic ObjNav, where predicted future-location distributions generate ranked navigation targets. Across ProcTHOR simulations and real-world robotic deployments, `FlowMaps` captures continuous, multimodal, and cross-environment spatio-temporal structure, improving object search and providing an effective prior for robotic reasoning beyond static scene assumptions.

Acknowledgments

This work has been carried out while Francesco Argenziano was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome. This research was conducted while Francesco Argenziano was enrolled as a visiting researcher at Mila - Quebec AI Institute. The work conducted at the Université de Montréal was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through an NSERC Discovery Grant and by the CIFAR AI Chair program (Liam Paull). Individual support was provided through NSERC Postgraduate Scholarships-Doctoral (PGS D) (Sacha Morin, Charlie Gauthier, Miguel Saavedra-Ruiz). This research was also enabled in part by computational resources provided by Mila (mila.quebec).

Appendix

Table 3: Closed-set semantic vocabularies used for dataset generation.

Pickupable objects $\mathcal{O}_{\text{obj}}$						**Receptacles $\mathcal{O}_{\text{BG}}$**			
ID	Class	ID	Class	ID	Class	ID	Class	ID	Class
0	Apple	14	Bottle	28	Spoon	0	CounterTop	9	Sink
1	Egg	15	Knife	29	Watch	1	Plate	10	Sofa
2	Fork	16	Spatula	30	BaseballBat	2	Pot	11	TVStand
3	Kettle	17	Bread	31	BasketBall	3	ShelvingUnit	12	GarbageCan
4	Ladle	18	ButterKnife	32	DishSponge	4	Chair	13	SideTable
5	Lettuce	19	Candle	33	ToiletPaper	5	DiningTable	14	Desk
6	Pencil	20	CellPhone	34	TissueBox	6	Dresser	15	CoffeeTable
7	Potato	21	Newspaper	35	TeddyBear	7	ArmChair	16	Box
8	SaltShaker	22	Vase	36	SoapBar	8	Bed		
9	SoapBottle	23	AlarmClock	37	PaperTowelRoll				
10	SprayBottle	24	KeyChain	38	DeskLamp				
11	Tomato	25	Laptop	39	TennisRacket				
12	WineBottle	26	Pen	40	Cloth				
13	Book	27	RemoteControl						

A Dataset and environments

Environment classes. We generated the environments using ProcTHOR and adopted the closed set of semantic labels available in the simulator for dynamic pickupable objects and receptacles. This resulted in a vocabulary $\mathcal{O}_{\text{obj}}$ of 41 pickupable object classes and a vocabulary of $\mathcal{O}_{\text{BG}}$ 17 receptacle classes. The former contains the dynamic objects whose locations may change over time, while the latter contains the static receptacles and furniture elements that define the nvironment layout and provide support surfaces for object placement. The complete list can be observed in Table 3.

Scene layouts. The generated dataset contains indoor ProcTHOR scenes with diverse spatial layouts and room configurations. Scenes range from compact single-room environments to larger multi-room households, with different floor plans, furniture arrangements, and numbers of receptacle instances. This variability affects both the geometric structure of the environment and the set of valid support surfaces available for object placement. A sample of 30 validation environments can be observed in Fig. 5

Object-receptacle compatibility and habit-conditioned schedules. To avoid arbitrary or semantically invalid placements, we use ProcTHOR's pickupable and receptacle weighting files to determine which pickupable object classes can be assigned to which receptacle classes. These metadata define a class-level compatibility prior between objects and support surfaces, ensuring that each object is placed only on semantically plausible receptacles. For example, kitchen objects such as Fork or Tomato may be assigned to counters, sinks, plates, or dining tables, while unrelated receptacles are excluded from their candidate sets. For each object class $o \in \mathcal{O}_{\text{obj}}$, we define the compatible receptacle set as $\mathcal{R}(o) \subseteq \mathcal{O}_{\text{BG}}$. Then, for each object instance in a scene, we select candidate receptacle instances whose class belongs to $\mathcal{R}(o)$.

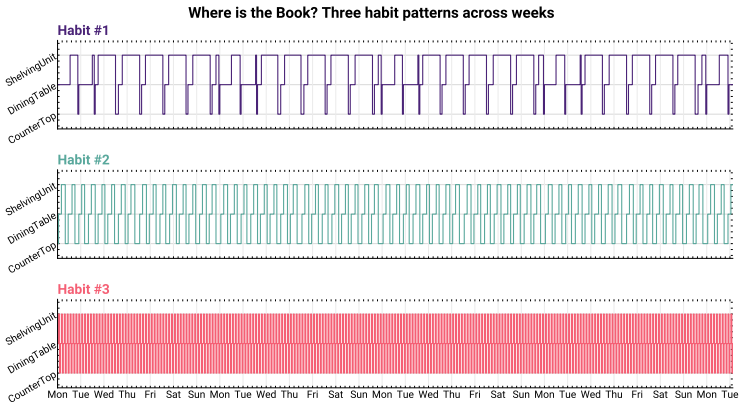


Figure 4: Different spatio-temporal behaviors of the same object ("Book") under different human habits in a ProcTHOR scene.

Figure 5: An overview of 30 ProcTHOR environments coming from the validation split.

Each dynamic scene is generated by assigning every pickupable object a habit-conditioned trajectory over this candidate set. The compatibility prior constrains which receptacles are valid, while the habit determines the temporal structure of the object's receptacle sequence. We consider the three habit families described in the main text. Habit#1 captures a preference-driven routine, where the simulated human spends more time in a few favored locations. Habit#2 represents a balanced rou-

Table 4: Tokenization and bounding box encoder hyperparameters.

Parameter	Value
Sequence length S	32
Hidden width d	256
Center normalization range $[c_{\min}, c_{\max}]$	$[-0.5, 34.0]$ m
Log-size offset ε_s	10^{-4}
Bounding box branches	4 translation/size $\times$ direction/magnitude branches
Bounding box MLP depth	4 layers, GELU
Label/flag MLP depth	2 layers, SiLU
Linear init, bounding box branches and label MLPs	$\mathcal{N}(0, 0.02^2)$, bias 0
LayerNorm init	gain 1, bias 0

tine, with time distributed roughly evenly across relevant locations. Habit#3 models a more dynamic routine, characterized by frequent transitions and short stays. An example of such rouines can be observed in Fig. 4.

Continuous placement by potential-field sampling. The habit schedule specifies the target receptacle for each object at each timestamp, but not a single deterministic 3D placement. To obtain continuous object locations, we sample placements from a potential field defined over the support surface of the selected receptacle. The sampled surface point determines the continuous position of the object on the receptacle, and the object is then teleported to the corresponding pose in the simulator. This separates the discrete semantic component of the data generation process from the continuous geometric one: the habit determines which receptacle an object moves to, while the potential field determines where on that receptacle surface the object is placed. This procedure produces trajectories in which objects follow semantically consistent, habit-conditioned patterns while still exhibiting continuous spatial variation. As a result, the model is not trained to predict only discrete object-receptacle relations, but rather full future bounding boxes in continuous 3D space.

B Implementation Details

B.1 Tokenization and encoding strategies

Bounding box normalization. Because ProcTHOR generates scenes with varying layouts, we normalize all bounding box center coordinates to $[0, 1]$ using fixed global bounds, c_{min} and c_{max}, computed from the extrema observed in the training set. Since object sizes vary substantially across semantic categories, for example between an "Egg" and a "Bed", we map bounding box extents to log-space and normalize them using the observed log-size range in the training data. Scenes are then tokenized to a fixed sequence length $S = 32$, as no environment exceeded this number of receptacles plus dynamic objects.

Embeddings. Each normalized bounding box is embedded through four separate branches, corresponding to translation magnitude, translation direction, size magnitude, and size direction. Each branch is a 4-layer Multi Layer Perceptron (MLP), implemented as two stacked Linear $\rightarrow$ GELU $\rightarrow$ Linear blocks with hidden width $d = 256$. Translation- and size-group sums are individually layer-normalized, summed, and passed through a final layer norm to produce the per-token geometric embedding. Semantic labels and the dynamic/background flag are embedded using learned tables of width d, each followed by a 2-layer Linear $\rightarrow$ SiLU $\rightarrow$ Linear MLP. Receptacle and dynamic-object classes use *separate* embedding tables to reflect their different cardinalities and priors. Hyperparameters are summarized in Table 4.

B.2 Variational Autoencoder

A general overview of the VAE can be observed in Fig. 6. The encoder e_ϕ and decoder d_ξ are residual MLPs with hidden width $d_{\text{vae}} = 256$, depth $L_{\text{vae}} = 3$, and an MLP expansion ratio of 4. Each residual block is a Linear $\rightarrow$ LayerNorm $\rightarrow$ SiLU $\rightarrow$ Dropout $\rightarrow$ Linear $\rightarrow$ LayerNorm $\rightarrow$ SiLU $\rightarrow$ Dropout

Table 5: VAE hyperparameters.

Parameter	Value
Latent dimension d_z	32
Hidden width d_{vae}	256
Depth L_{vae}	3
MLP expansion ratio	4
Dropout	0.1
Linear init	Xavier-uniform, bias 0
Iterations	30,000
Batch size	256
Optimizer	AdamW
Learning rate	10^{-4}
(β_1, β_2)	$(0.95, 0.999)$
ε	10^{-8}
Weight decay	10^{-6}
LR schedule	cosine, 500 linear-warmup steps
λ_{CIoU}	1.0
λ_{L_1}	5.0
λ_{CE}	1.0
Final KL weight β	10^{-2}
KL warm-up steps T_{kl}	10,000, linear from 0 to β

stack with a skip connection. The encoder input is the bounding box embedding summed with the class-label embedding described in Section B.1; its final linear layer projects to $2d_z$ and is split into $(\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2)$. The decoder mirrors the encoder and terminates in two heads: a 6-dimensional geometry head with sigmoid activation, so outputs stay in $[0, 1]^6$, and a $|\mathcal{O}_{\text{obj}}|$-way class-logit head. The CIoU term is computed in metric space after inverting the log-size mapping, so that the area/volume penalty reflects actual overlap rather than overlap in the normalised representation. After training, per-dimension latent statistics $(\boldsymbol{\mu}_{\mathcal{D}}, \boldsymbol{\sigma}_{\mathcal{D}})$ are computed once on the training split and cached on disk. The FM stage operates entirely on standardized latents $(\mathbf{z} - \boldsymbol{\mu}_{\mathcal{D}})/\boldsymbol{\sigma}_{\mathcal{D}}$, with the inverse transform applied to predicted latents before decoding. All other architectural and optimization hyperparameters are reported in Table 5.

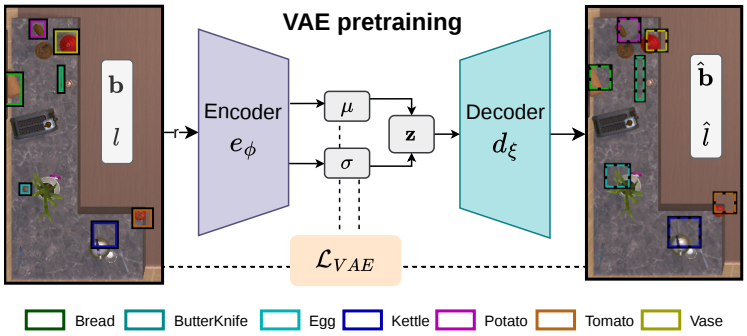


Figure 6: VAE used to encode and decode object tokens in a ProcTHOR environment. Dotted bounding boxes mean decoded objects.

B.3 Latent Flow Matching Network

Map encoder. The map-encoder layers are pre-norm Transformer Encoder modules with GELU activations. The dynamic/background flag is implemented as a 2-entry learned parameter, with one d-dimensional vector per type, added on top of the bounding box, class, and observation time embeddings before the first layer.

CDiT block. The adaLN-Zero projection is a single SiLU $\to$ Linear$(d, 8d)$ MLP whose output is split into eight chunks $(\gamma_1, \beta_1, \alpha_1, \gamma_2, \beta_2, \alpha_2, \gamma_3, \beta_3)$. The first triple modulates the query token and gates the cross-attention residual. The second triple modulates and gates the feed-forward sublayer. The last pair modulates the context tokens, namely keys and values, before the cross-attention sublayer. The feed-forward sublayer is a two-layer Linear $\to$ GELU $\to$ Linear MLP with the same

Table 6: Latent flow-matching network hyperparameters.

Parameter	Value
Hidden width d	256
Attention heads H	8
MLP expansion ratio	4
Map encoder depth N_T	8
CDiT block depth N_C	8
Dropout	0.1
Latent input embedder	2-layer MLP, $d_z \to d$, SiLU
Flow time embedder	sinusoidal, 256, plus 2-layer MLP
τ, τ_f embedders	separate $T_{\max} \times d$ tables
Query label embedder	2-layer MLP, separate table
Conditioning vector c	$\mathrm{emb}(t) + \mathrm{emb}(\tau_f) + \mathrm{emb}(l_q)$
Init: adaLN-Zero and final gate	zero
Init: other linear layers	Xavier-uniform
Init: embedding tables, class/time MLPs	$\mathcal{N}(0, 0.02^2)$

expansion ratio 4 as the map encoder. Cross-attention uses H heads with biases on the key/value projections.

Conditioning embeddings. The flow time $t \in [0, 1]$ is encoded with a 256-dimensional sinusoidal frequency embedding followed by a 2-layer Linear $\to$ SiLU $\to$ Linear MLP. The final timestamp τ_f and the observation time τ use *separate* learned tables of size $T_{\max} \times d$. The queried object label l_q uses its own 2-layer-MLP class embedder, distinct from the map-encoder object table, so that the query and map tokens are not tied. The conditioning vector fed to each CDiT block is the sum of the embeddings of t, τ_f, and l_q. The noisy latent $\mathbf{z}_t$ is lifted from $d_z = 32$ to $d = 256$ by a 2-layer Linear $\to$ SiLU $\to$ Linear input embedder.

Initialization. All adaLN-Zero projections and the gate projection of the final output head are zero-initialized so that the network starts as an identity map on the noisy latent [7]. Other linear layers are initialized with Xavier-uniform. Embedding tables and the projection MLPs of the class and timestep embedders use $\mathcal{N}(0, 0.02^2)$. Architectural hyperparameters are reported in Table 6.

B.4 FlowMaps Training and Sampling

We adopt the CondOT path [36] with $\sigma_t = 0$, which reduces the conditional probability path to the straight-line interpolation $\mathbf{z}_t = (1-t)\mathbf{z}_0 + t\mathbf{z}_1$ with constant target velocity $u_t^{\text{target}} = \mathbf{z}_1 - \mathbf{z}_0$. Within each mini-batch, source and target samples are paired by an exact Hungarian solver on the squared-Euclidean cost $\|\mathbf{z}_0 - \mathbf{z}_1\|_2^2$. Only $\mathbf{z}_0$ is permuted, leaving $\mathbf{z}_1$ and every conditioning variable in their original order so that semantic alignment with (M_τ, l_q, τ_f) is preserved. We sample $t = \mathrm{sigmoid}(s)$ with $s \sim \mathcal{N}(0, 1)$, giving a logit-normal flow-time distribution. This concentrates supervision on the middle of the trajectory, where the target velocity has its largest variance. We also experimented with sampling t from a uniform distribution $U[0, 1]$ and from a Beta distribution, but empirically found the logit-normal schedule to yield the best results. During inference, we integrate with a fixed-step Euler solver on a uniform grid over $[0, 1]$, with a step of 0.05. Smaller steps and different intregration methods (*e.g.,* midpoint) produced no measurable improvement. The map encoder is run once per scene and its output reused across all dynamic objects in the scene and across the N_{pred} posterior samples drawn for each query. Training and sampling hyperparameters are summarized in Table 7.

Table 7: Flow-matching training and sampling hyperparameters.

Parameter	Value
Iterations	30,000
Batch size	1024
Optimizer	AdamW
Learning rate	10^{-4}
(β_1, β_2)	$(0.95, 0.999)$
Weight decay	10^{-6}
LR schedule	cosine, 500 linear-warmup steps
Validation interval	every 2,500 iterations
Path	CondOT [36], $\sigma = 0$
Coupling	exact mini-batch OT, Hungarian
Flow-time sampler	logit-normal, $s \sim \mathcal{N}(0, 1)$
Loss	squared-L_2 CFM regression
EMA decay	0.9999
EMA inverse-gamma / power	1.0 / 0.75
EMA warm-up	none
ODE solver, inference	euler
Step size	0.05

C Experiments

C.1 Distributional Evaluation: Experimental Details

We evaluate on a shared minival split of 25 validation environments, identical across the three habits described in the main paper. For each scene with T hourly timestamps we select n_τ evenly-spaced observation times $\tau \in [0, T-2]$ and, for each τ, up to 10 evenly-spaced query times $\tau_f \in (\tau, T-1]$, giving at most 30 (τ, τ_f) pairs per (env, l_q). For each pair we draw $K{=}50$ samples from the predictor. Per-pair, per-instance *minFDE@K* and *Rec@1* are then averaged over all (τ, τ_f) pairs and all instances of class l_q in the environment. In contrast, the distributional shape metrics are computed once per (env, l_q) using pooled point clouds. The predicted cloud is obtained by concatenating the K sampled centers across all pairs, yielding $M \approx 1500$ points. The ground-truth cloud is the union of the bounding box centers of every instance of class l_q at every evaluated τ_f.

Regarding the metrics, *minFDE* uses the full 3D Euclidean distance to the ground-truth bounding box center, while *Rec@1* uses xz-only distance with $\delta{=}1m$. *Density* and *Coverage* follow Naeem et al. [40] with $k_{\text{NN}}{=}5$, augmented with a $0.5m$ radius floor that prevents degenerate k-NN radii on classes whose ground-truth positions coincide across timestamps (*e.g.,* static objects on the same receptacle). *TV* and *JS* are computed on an xz floor grid of resolution $0.5\,\text{m}$, with JS reported in nats (bounded by $\ln 2 \approx 0.693$). We rely on sample-based histogram divergences rather than the change-of-variables likelihood obtainable from the flow itself for two distinct reasons. First, the inverse-ODE likelihood is defined only for `FlowMaps`, and even there only in the VAE latent space the flow operates on; the VAE decoder is not invertible, so a likelihood in 3D position units would require an importance-weighted estimate that introduces its own variance and bias. The baselines fare no better: FreqPrior and LLMPrior have closed-form densities, but those are two-dimensional delta measures concentrated on receptacle top surfaces, while EmpiricalMean and Mean`FlowMaps` are point masses. In every, case the pointwise log-likelihood at a generic 3D ground-truth coordinate is either $-\infty$ or ill-defined, and the only way to make the numbers comparable is to fit a separate density estimator (*e.g.,* kernel density estimation) on each method's samples, which introduces additional hyperparameters and confounds the comparison. Second, even within `FlowMaps` the inverse-ODE likelihood scores how much density the model assigns at the ground-truth coordinates, whereas the downstream use of `FlowMaps` consumes *samples* drawn from the model rather than its evaluated density: a metric defined on the predicted samples is therefore more directly aligned with the actual object of comparison. TV and JS satisfy both requirements simultaneously:

they are computable from any baseline's samples without further fitting, and they measure how the predicted mass is *spatially allocated* across the scene rather than how dense it is at the reference point.

FreqPrior. This baseline represents a class-conditional categorical over receptacle categories. From the training split of the current habit we estimate, for each pickupable class l_q and receptacle category r, the empirical frequency of l_q resting on instances of r, and store the resulting table $P(r \mid l_q)$ alongside the data. At test time, for each of the K samples we *(i)* restrict $P(\cdot \mid l_q)$ to receptacle categories present in M_τ and renormalize, *(ii)* sample a category r, emph(iii) sample a concrete receptacle of that category uniformly, and *(iv)* sample a point uniformly inside its footprint polygon. The prior is time-invariant and the scene receptacles are read at τ and reused for any τ_f, so this baseline has no notion of time.

LLMPrior. LLMPrior uses the same point-sampling pipeline as FreqPrior, but the distribution over receptacle *instances* is replaced by a ranked top-T list suggested from an LLM. We use `llama3.1:8b` as our LLM, with T=5, sampling temperature 0, and a rank-softmax temperature of 1.0, so the rank-i receptacle receives mass proportional to $\exp(-i)$. One LLM call is issued per $(\text{env}, l_q, \tau, \tau_f)$: the returned indeces induce the receptacle distribution, and the K predicted points are obtained by drawing receptacles from it and sampling uniformly inside the chosen footprint at its top face. The system prompt remains unchanged across queries. For each query, we generate the user prompt by filling the bold-bracketed placeholders in the box below. These placeholders specify the observation time τ, the scene state M_τ, the queried class l_q, the receptacle that currently contains the target, the elapsed time $\tau_f - \tau$, the query time τ_f, and the top-T size. The scene state is written as a bullet list over receptacles, with each line reporting the pickupable categories currently on that receptacle, or "empty" when none are present. Timestamps are formatted as `Week W, Weekday HH:00` starting from `Week 1, Monday 00:00`, and elapsed time as `X day(s) and Y hour(s)`.

System Prompt: You are an agent that has to find an object that may have moved from its previous location. You will be given the layout of receptacles in a scene and the object's previous location. Reason about where the object is most likely to be now.

User Prompt: You are an agent that has to find an object that could have moved from its previous location. You know that on **[**τ**]** the environment has the following layout: **[**M_τ **receptacle list]**. Also, on **[**τ**]**, the **[**l_q**]** is on **[current receptacle of** l_q**]**. **[**$\tau_f - \tau$**]** later, it is now **[**τ_f**]**. From the list of receptacles above, enumerate the top **[**T**]** most likely receptacles where the **[**l_q**]** can be found on **[**τ_f**]**, ordered from most to least likely. Reply with one receptacle id per line, exactly as written in the list above. No commentary, no numbering, no extra text.

When the queried object is not assigned to any receptacle at τ, the "**[**l_q**]** is on **[current receptacle of** l_q**]**" clause is replaced by "the **[**l_q**]** is not on any of the listed receptacles".

EmpiricalMean. This baseline represents an oracle point predictor: for each (env, l_q), it returns the centroid of every ground-truth bounding-box center of class l_q pooled across the scan's timestamps, replicated K times. The predictor consults the evaluation set and is independent of τ and τ_f; since this centroid minimizes L_2 error against the pooled ground-truth cloud, it upper-bounds any deterministic regression baseline conditioned on at least (M_τ, l_q). Its distribution-shape entries in the main results table are not reported since these metrics are not informative for a predictor that emits K identical samples.

Mean`FlowMaps`. This baseline is a wrapper around the trained `FlowMaps` predictor that, per query, draws K_{int}=50 samples internally, returns their arithmetic mean, and replicates that point K times in the output. The conditioning (M_τ, l_q, τ_f) is inherited from `FlowMaps`; the only operation removed is multimodality, which isolates its contribution to the metric gains. Its distribution-shape entries are not reported for the same reason as EmpiricalMean.

C.2 Object navigation: experimental details

We evaluate on the same minival split of 25 validation environments used in the distributional protocol, with the agent embodied in AI2-THOR through ProcTHOR scene assets. For every environment we sample 5 target objects, and for each target we draw 5 (τ, τ_f) pairs (the observation timestamp at which the agent perceives the scene, and the future timestamp at which we ask it to retrieve the object), yielding 600+ episodes per habit. Each method receives, at the start of an episode, the per-timestep data that backs the scene up to τ (the observation history of receptacle-object assignments) and is asked to produce a ranked list of K candidate future receptacles for the target at τ_f. The agent then navigates the AI2-THOR controller toward those candidates in order, querying `GetShortestPathToPoint` for the planner-relevant distances, with a per-episode budget of 1000 environment steps. An episode is successful at depth K if any of the first K candidates places the agent within $\delta_{\text{dist}} = 1.25m$s of the target and the target is observed from the agent's point of view. In simulator runs, observation is assessed using AI2-THOR ground-truth visibility. In the real-robot demonstration, the VLM is used only to trigger a candidate target observation; a trial is counted as successful only when this trigger occurs at a valid target location, so VLM false positives are not counted as successful detections.

For `FlowMaps`, the ranked candidate list is built by drawing N_{preds}=25 samples per query from the trained flow, clustering them with DBSCAN (ε=1.0 m, $\min$ =2) and ranking clusters by mass. Ties are broken by the cluster centroid that minimises the distance from the current agent pose. Across all baselines we report Success Rate, SPL, and their per-K variants exactly as in the main results table.

TAP-LGX. We follow the formulation of [21], which extends the zero-shot exploration of LGX [18] with a rolling *transit memory* of past observations. At each LGX step the agent performs a 360° scan, the visible objects are fed into a VLM together with the current memory, the VLM returns a single object name, and the agent advances a fixed distance toward it; failures and target sightings are appended to the memory. We reproduce the published exploration loop in full: the LGX step-distance is $2m$, the per-episode budget is 30 LGX steps, and the memory horizon is the last 30 entries; on lock-on the agent navigates directly to the detected target. We deviate from the original setup in three points. First, we substitute the VLM with the same `llama3.1:8b` used elsewhere in our evaluation, constraining its output to the visible-object set with a JSON schema so the language model behaves as a discrete chooser instead of a free-form generator.Second, we replace the RGB-frame YOLO detections used in the original paper with ground-truth visibility queries from AI2-THOR, filtered to the LGX vocabulary. This is a substantial strengthening of the perception input, since it removes detector noise and gives the system the same perception-noise floor used by the other baselines in the table. Third, we provide input-parity to the other basilines, we pre-seed the rolling memory with the full per-receptacle scene state at every past timestep $t \leq \tau$, and grows during exploration with one entry per LGX step recording the agent's pose, the list of visible objects of the current scan, and the heading the LLM chose. The system and user prompts are shown in the box below: the system prompt renders the rolling memory verbatim, the user prompt is issued once per LGX step to elicit a single visible-object choice, and the bold-bracketed placeholders are filled per query with the target class l_q, the current memory contents, and the set of objects detected at the current scan.

System Prompt: I am a smart robot trying to find a **[l_q]** in my house. The target sometimes moves between receptacles over time; the lines below record where I have seen it or the scene before, and what I observed as I walked around. Use this to reason about where it likely is now.
Memory:
[rolling memory: one line per past entry, e.g. "[step s] near (x, z) heading h deg observed [obj list] $\rightarrow$ went toward a" or "[prior t] target on r"; pre-seeded scene-state lines in TAP-LGX]

User Prompt: I want to find a **[l_q]** in my house. Which object from **[visible objects at the current scan]** should I go towards? Reply in ONE word.

CEG. The Cost-Effective Greedy planner of Wang et al. [22] can be viewed as a multiplicative simplification of the original OSG additive blend described below. It ranks unvisited receptacles by the ratio p_i/d_i, where p_i is the estimated probability that receptacle i currently contains the target, and d_i is the navigation distance from the agent to that receptacle. At each step, the agent visits the highest-ranked receptacle. If the target is not found, the corresponding probability mass is set to zero, the remaining masses are renormalized, and the procedure continues until the target is found, all receptacles have been visited, or the per-episode receptacle budget is exhausted. We set this budget to 10, matching the typical scene size. Navigation distances are computed using AI2-THOR's `GetShortestPathToPoint`. To avoid numerical artifacts when receptacles are collocated or nearly collocated, we use a $0.25m$ epsilon in the denominator of p_i/d_i, matching the ProcTHOR grid resolution. We evaluate two variants of the probability estimate p_i. The *scene-prior* (SP) variant uses the per-class receptacle co-occurrence table $P(r \mid l_q)$, which is the same table used by FreqPrior in the distributional evaluation. The *LLM* variant instead queries `llama3.1:8b` for an unnormalized probability for each receptacle. The LLM estimator prompt, shown in the box below, follows the structure of the paper, but replaces the "simulated resident" human-activity hint with one of three habit-specific lines.

- Habit #1: ``This [l_q] has a few favorite places. Most often on [r_1] ([p_1]%), sometimes [r_2] ([p_2]%), and occasionally [r_3] ([p_3]%).''
- Habit #2: ``This [l_q] doesn't have a favorite spot. It is equally likely to be on any receptacle it can sit on.''
- Habit #3: ``This [l_q] moves between receptacles quickly without dwelling. Past placements aren't strong predictors of where it is now. Recently seen at [r_1, r_2, r_3].''

where r_i and p_i are filled from the target's empirical placement history in the environment. This grounds the LLM estimator in the dataset's residence model in the same way the original paper paper used the simulated resident to elicit human knowledge.

System Prompt: You are an expert assistant that estimates where household objects are likely to be found. Given a scene's receptacle list and the object's previous location, you must output a probability for every receptacle, indicating how likely the object currently rests on it. Probabilities may be unnormalized but must be non-negative.

User Prompt: Behavioral context: **[habit-specific hint, see CEG description above]**.
You are an agent that has to find an object that could have moved from its previous location. On **[τ]** the environment has the following layout: **[M_τ receptacle list]**. Also, on **[τ]**, the **[l_q]** is on **[current receptacle of l_q]**. **[$\tau_f - \tau$]** later, it is now **[τ_f]**. Estimate the probability that the **[l_q]** is on each of the receptacles listed above on **[τ_f]**. Probabilities may be unnormalized but must be non-negative. Reply with one line per receptacle in the exact form: `<receptacle_id>\t<probability>`. Use every receptacle id from the list, no extra commentary or numbering.

OSG. The One-Step Greedy planner of [3] uses the original additive-blend score from which CEG can be seen as a simplified variant. It ranks receptacles according to $\frac{\alpha_p}{d_i} + (1-\alpha_p)p_i$, where d_i is the navigation distance to receptacle i and p_i is the estimated probability that it contains the target. The coefficient α_p controls the trade-off between cost and reward: $\alpha_p = 1$ yields purely distance-based selection, while $\alpha_p = 0$ yields purely probability-based selection. We set $\alpha_p = 0.5$, following the default used in [22]. All remaining components are kept identical to the CEG setting described above. In particular, we use the same two probability estimators, namely the scene-prior table $P(r \mid l_q)$ and the LLM-based variant; the same online visit-fail-zero-renormalize procedure; the same AI2-THOR `GetShortestPathToPoint` distance computation; the same $0.25m$ denominator guard; and the same per-episode receptacle budget of 10. The LLM variant also uses the prompt shown in the previous box.

HOMER. We adopt the spatio-temporal object-tracking model of [17]. The model represents each scene as a heterogeneous graph whose nodes correspond to rooms, furniture, and objects, and whose edges encode containment and spatial proximity. Given this graph and the observation history, a

Graph Translator Network predicts, for each pickupable object, a probability distribution over its possible receptacles at the next timestep. We train the network on our training split using the official implementation and training protocol. However, instead of fitting one model per household, we train a single cross-environment model. This setting is aligned with the generalization regime evaluated by `FlowMaps`: the same checkpoint is applied to all minival environments, without any per-house fine-tuning. At test time, we run the network on the environment scene graph and the observation history up to time τ. We then apply a softmax over destination receptacle nodes and extract the row corresponding to the queried target object. This row defines the predicted receptacle distribution at the future time τ_f. The top-$K = 10$ receptacles under this distribution are returned, in ranked order, as navigation goals, preserving the one-shot prediction semantics of the original method.

SGM. We use the Scene Graph Memory (SGM) predictor of [23]. SGM represents the environment as a hierarchical scene graph, with nodes organized as HOUSE $\rightarrow$ ROOM $\rightarrow$ FURNITURE $\rightarrow$ OBJECT, and applies a HEAT neural edge predictor over this structure. The graph is augmented with temporal node and edge counters, which summarize the observation history up to time τ. Given a queried object, the predictor scores hypothetical edges between that object and each candidate receptacle. These scores are converted to sigmoid probabilities and used to rank receptacles directly. We take the top-$K = 10$ ranked receptacles as the navigation-goal sequence. Our implementation follows the published SGM architecture, featurizer, prior, and temporal-counter pipeline. The only departure from the original per-household training setup is that we train a single cross-environment checkpoint. This matches the generalization setting evaluated by `FlowMaps`: the same model is applied to every minival environment, with no per-house fine-tuning.

Naive LLM. This approach consists in a direct prompting baseline that asks the LLM to rank the receptacles by likelihood of currently hosting the target. The model and the prompting template are identical to those of the LLMPrior baseline used in the distributional evaluation, with the single difference that the requested top-T is raised from 5 to 10 to match the ranked-list size of the other ObjNav baselines. We refer to the distributional appendix for the verbatim system and user prompts.

C.3 Ablation Studies

We additionally report a full suite of ablation studies to display how each components of our architecture affect the final results. In particular, we consider the following ablation cases.

A first family probes the role of semantic identity, that is, whether the model benefits from knowing what the entities in the problem are rather than only where they sit. In the *no-query-class* case we drop the class of the query object from the conditioning vector, forcing the model to place an object without knowing its category. In the *no-scene-class* case we remove the per-token class embeddings from the scene map, so that furniture and object tokens are described purely by their bounding boxes, with no semantic label. In the *no-type-emb* case we remove the coarser type embedding that distinguishes furniture tokens from object tokens, testing whether even this binary distinction within the scene carries useful signal.

A second family examines two architectural and encoding choices. In the *no-scene-encoder* case we remove the Transformer encoder that performs self-attention over the scene tokens before they are consumed by the diffusion backbone, letting the cross-attention layers reason over the raw scene representation directly. In the *linear-bbox-embedding* case we replace our structured three-dimensional bounding-box embedding, which decomposes each box into separate direction and scale components processed by dedicated MLPs, with a single linear projection, in order to assess whether this structured decomposition is worth its additional complexity.

The ablations are reported in Table 8 and Table 9. Overall, the ablations show that semantic information is the dominant factor. Removing the query class causes the largest degradation, especially in ObjNav metrics, indicating that the model needs the target object category to generate placements that are useful for downstream navigation. Removing scene-level class information also substantially hurts both distributional quality and navigation performance, while removing only the coarse object/furniture type embedding has a milder but still consistent effect. The architectural ablations

Method	minFDE(m) ↓	Rec@1↑	Density↑	Coverage↑	TV↓	JS↓
`FlowMaps` (full)	**0.342**	**0.942**	**0.886**	**0.999**	**0.829**	**0.482**
no-query-class	0.438	0.930	0.438	0.997	0.922	0.580
no-scene-class	0.400	0.932	0.636	0.998	0.894	0.544
no-type-emb	0.377	0.937	0.674	0.997	0.885	0.533
no-scene-encoder	0.372	0.934	0.805	0.995	0.844	0.498
linear-bbox-emb	0.376	0.934	0.755	0.995	0.865	0.514

Table 8: Distributional metrics ablations. Results are averaged over the three habits.

Method	SR@1↑	SR@5↑	SR@10↑	mSR↑	SPL@1↑	SPL@5↑	SPL@10↑	mSPL↑
`FlowMaps` (full)	**53.06**	**68.14**	**69.34**	**65.86**	**42.52**	**47.34**	**47.50**	**46.66**
no-query-class	19.18	47.38	50.82	43.33	15.15	24.30	24.79	23.02
no-scene-class	39.02	59.02	59.51	55.52	31.31	37.83	37.90	36.90
no-type-emb	47.21	63.61	64.59	60.79	38.03	43.42	43.61	42.65
no-scene-encoder	47.21	64.59	65.25	61.51	37.92	43.47	43.54	42.62
linear-bbox-emb	47.05	61.97	63.11	59.67	37.58	42.75	42.90	42.00

Table 9: ObjNav ablations. Results are averaged over the three habits.

are less severe, but still meaningful: removing the scene encoder or replacing the structured bounding box embedding with a linear projection reduces performance across most metrics. This suggests that both contextual reasoning over scene tokens and the structured geometric encoding contribute to the final model, although their impact is smaller than that of semantic conditioning.

Failure analysis. We further analyze how navigation episodes terminate by decomposing each run into successful discoveries at different ranked modes and two failure cases: mode exhaustion, where all proposed targets are visited without observing the object, and step-budget failure, where the agent runs out of steps before completing the ranked search. Fig. 7 reports this decomposition for `FlowMaps`, HOMER, and SGM across the three habits.

Across all habits, `FlowMaps` concentrates a larger fraction of successful episodes in the first predicted mode. In Habit #1, `FlowMaps` finds the target at the first mode in 57.7% of episodes, compared with 46.7% for HOMER and 48.4% for SGM. The same trend holds in Habit #2, where `FlowMaps` reaches 49.7% first-mode success, and in Habit #3, where the gap becomes largest: 51.0% for `FlowMaps`, compared with 38.0% for HOMER and 27.2% for SGM. This indicates that the continuous multimodal distribution learned by `FlowMaps` is not only useful for covering plausible future locations, but also for ranking them: when the method succeeds, it more often does so without requiring the agent to exhaust later proposals.

The failure modes reveal a complementary trend. `FlowMaps` consistently produces fewer step-budget failures than the graph-based baselines. For example, in Habit #1 only 10.5% of `FlowMaps` episodes exceed the step budget, compared with 20.5% for HOMER and 18.7% for SGM. In Habit #3, the same failure accounts for 17.4% of `FlowMaps` episodes, versus 24.1% for both HOMER and SGM. This suggests that `FlowMaps` proposes targets that are spatially more efficient for the downstream navigation policy, reducing failures caused by long or unproductive search trajectories. At the same time, `FlowMaps` exhibits a larger fraction of mode-exhaustion failures. This effect is visible in all three habits, with 16.9%, 16.4%, and 14.4% exhausted episodes for Habits #1, Habit #2, and Habit #3, respectively. By contrast, HOMER and SGM generally have lower exhaustion rates, but compensate with more late-mode successes and more step-budget failures. This points to a distinct error profile: `FlowMaps` tends to make sharper and more useful early predictions, but when its posterior misses the correct region, additional ranked modes are less likely to recover the object. In contrast, HOMER and SGM spread successful detections over later modes, which can improve recovery in some cases but increases the cost of search. Overall, the Sankey analysis shows that `FlowMaps`' advantage is primarily driven by earlier target localization and fewer navigation-induced failures. The remaining errors are more often due to the predicted distribution missing the correct region than to inefficient traversal. This suggests that future improvements should focus on increasing tail-mode coverage while preserving the strong first-mode ranking behavior, for exam-

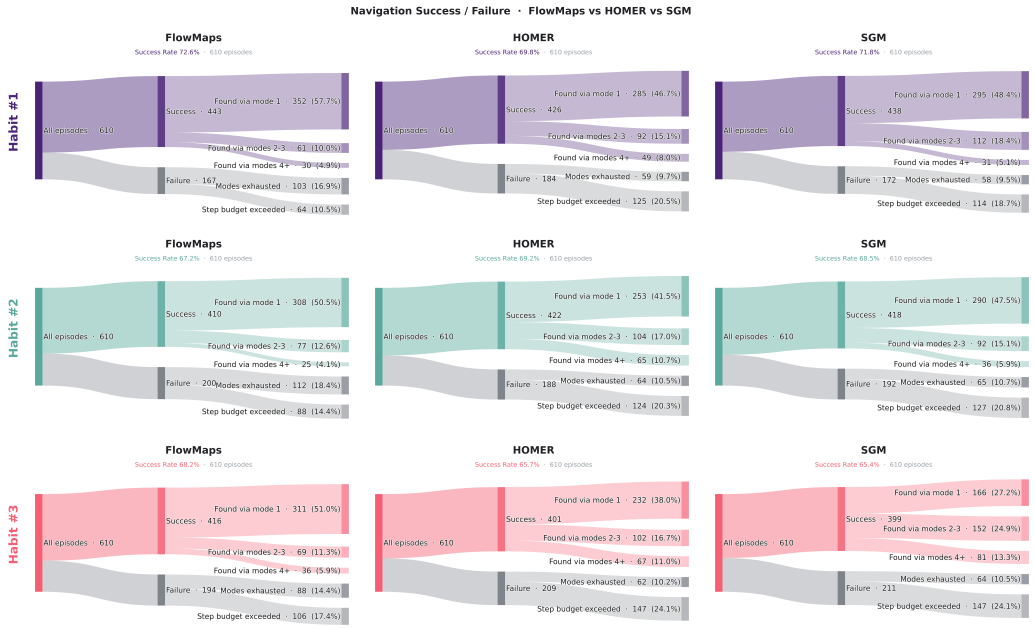


Figure 7: Success and failure decomposition for ObjNav episodes across habits. `FlowMaps` produces more first-mode successes and fewer step-budget failures, while its remaining failures are mostly due to exhausting the proposed modes.

ple by encouraging more diverse samples before clustering or by explicitly calibrating cluster mass during target ranking.

C.4 Additional Results

Object Navigation Examples. Figs. 8-15 collect representative successful episodes across different habits, environments and objects. Every row reads left to right as one episode. The first panel shows the raw predictive distribution: all $N_{\text{preds}}=25$ samples drawn from the flow for the queried object l_q at τ_f, projected onto the top-down map. The second panel overlays the DBSCAN clusters ranked by mass (numbered #k) together with the ground-truth box of l_q both at the observation time τ (dashed) and at the query time τ_f (solid): the offset between the two makes explicit that retrieval requires anticipating where the object moved, not recalling where it was last seen. The third panel draws the trajectory the agent actually executed, colored by attempt, as it visits the ranked modes in order from its start pose. The fourth panel is the egocentric view at the moment the target is confirmed. Two patterns recur. For low-depth successes the highest-mass mode lands directly on the τ_f ground truth and the agent reaches it in a near-straight path, confirming that `FlowMaps` concentrates probability on the correct future receptacle even when it is far from the last observed location. For higher-depth successes the trajectory visibly re-routes between modes: the agent rules out the leading clusters and falls back through the ranked list, which is exactly the behavior the mass-based ranking is meant to support when the future placement is genuinely multimodal.

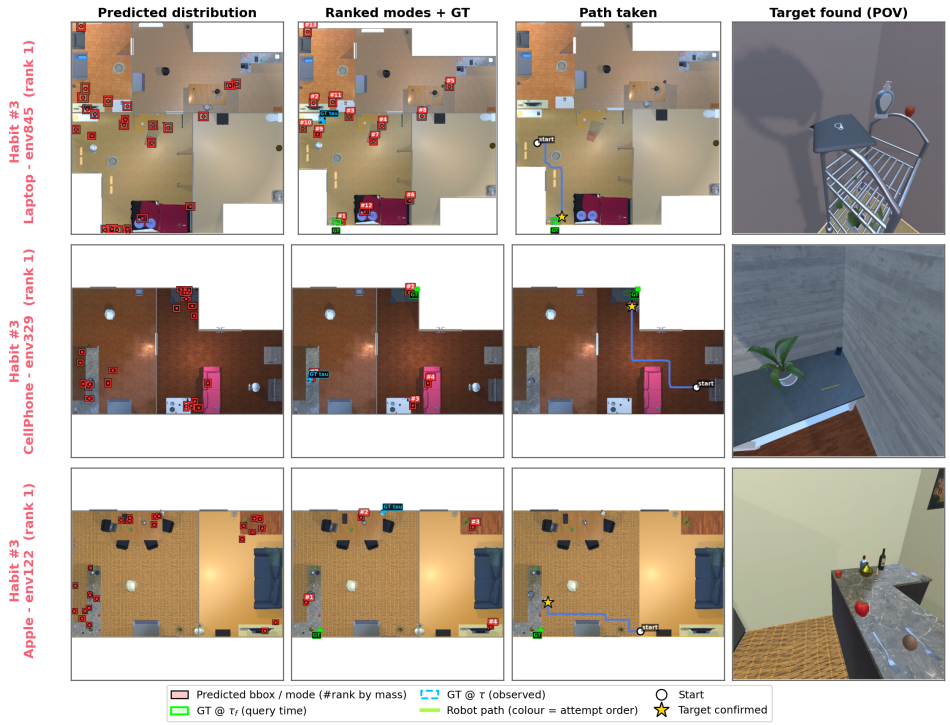


Figure 8: Representative successful ObjNav episodes with `FlowMaps`, set 1. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

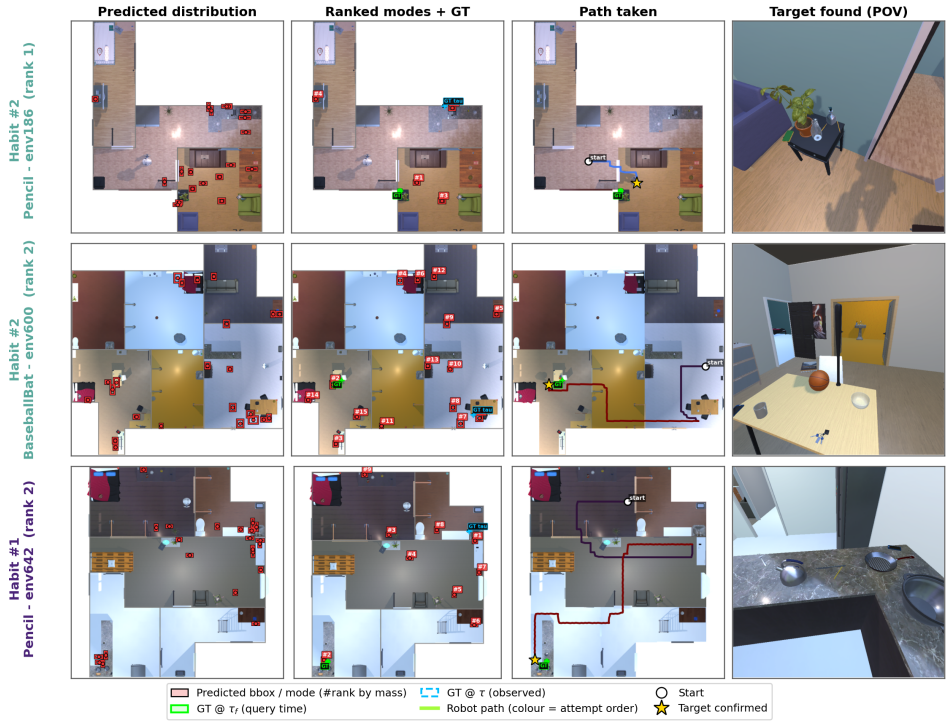


Figure 9: Representative successful ObjNav episodes with `FlowMaps`, set 2. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

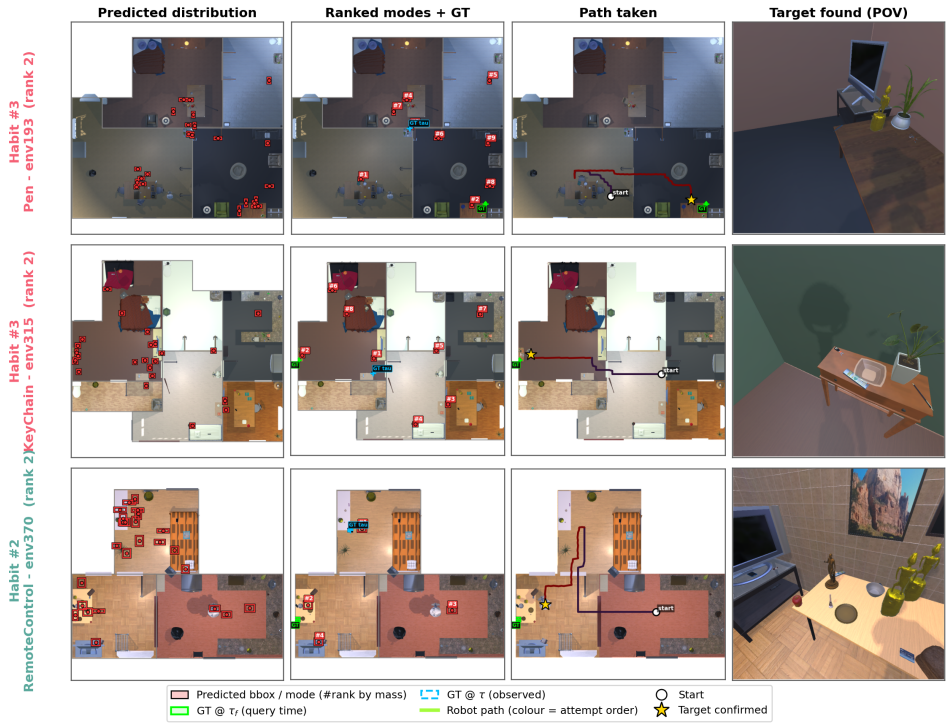


Figure 10: Representative successful ObjNav episodes with `FlowMaps`, set 3. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

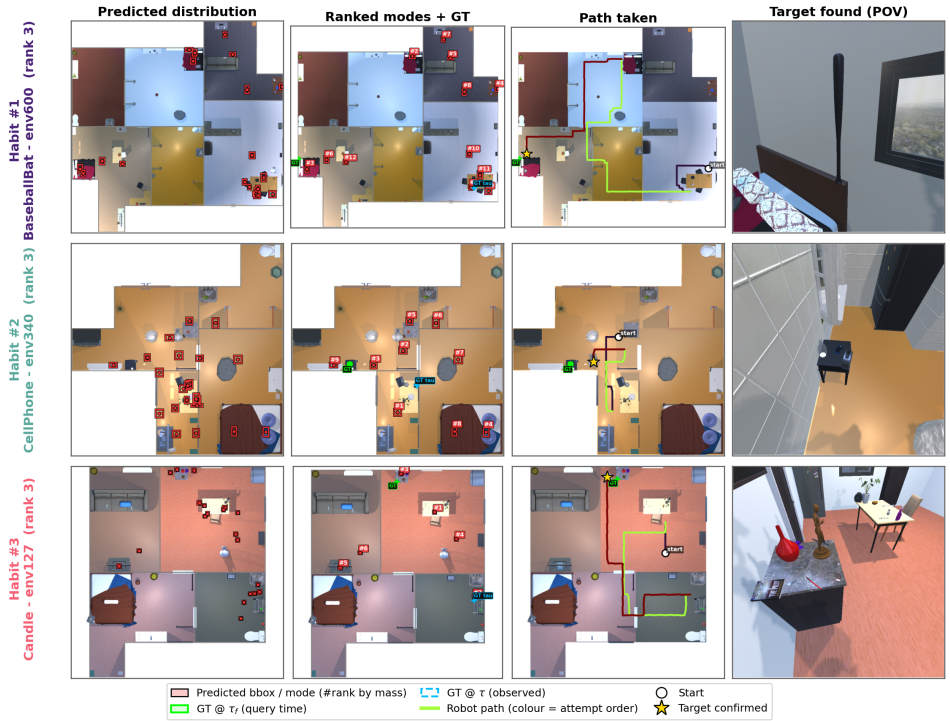


Figure 11: Representative successful ObjNav episodes with `FlowMaps`, set 4. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

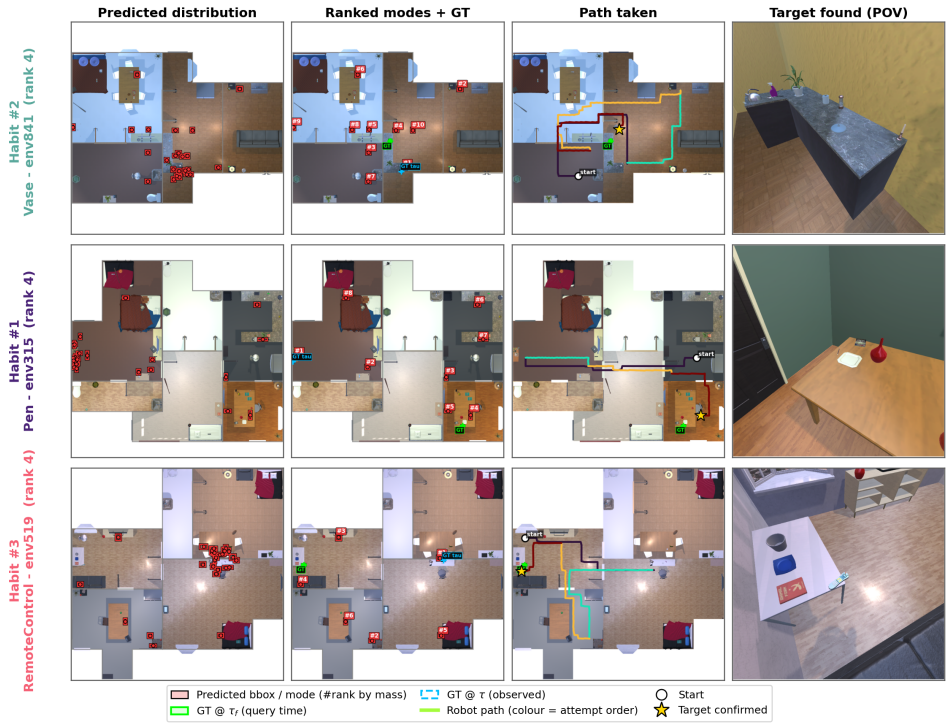


Figure 12: Representative successful ObjNav episodes with `FlowMaps`, set 5. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

Figure 13: Representative successful ObjNav episodes with `FlowMaps`, set 6. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

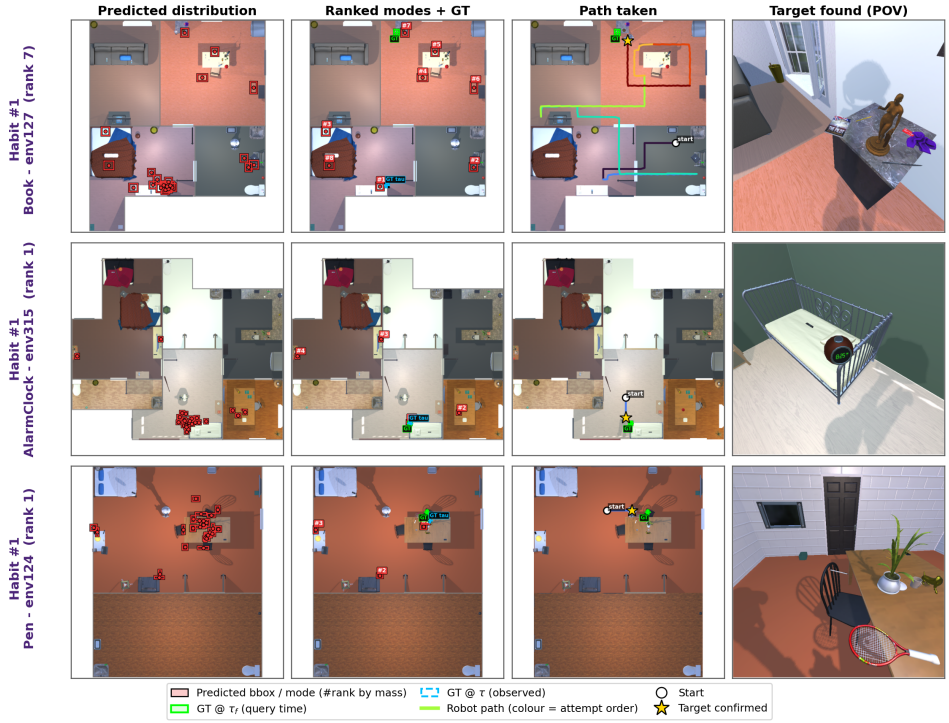


Figure 14: Representative successful ObjNav episodes with `FlowMaps`, set 7. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

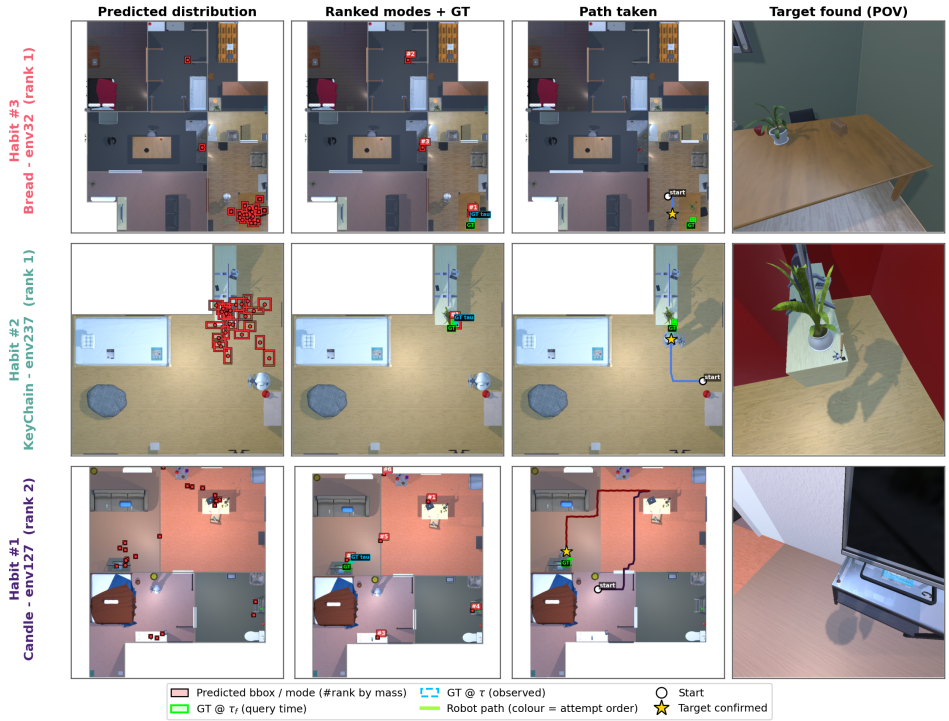


Figure 15: Representative successful ObjNav episodes with `FlowMaps`, set 8. Each row shows the predicted samples, ranked clusters, executed trajectory, and final target confirmation view.

References

[1] G. Mois and J. M. Beer. The role of healthcare robotics in providing support to older adults: a socio-ecological perspective. *Current Geriatrics Reports*, 9(2):82–89, 2020.

[2] R. J. López-Sastre, M. Baptista-Ríos, F. J. Acevedo-Rodríguez, S. Pacheco-da Costa, S. Maldonado-Bascón, and S. Lafuente-Arroyo. A low-cost assistive robot for children with neurodevelopmental disorders to aid in daily living activities. *International Journal of Environmental Research and Public Health*, 18(8):3974, 2021.

[3] 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 (ICRA)*, pages 5645–5652, 2023. doi:10.1109/ICRA48891.2023.10160473.

[4] N. F. Troje. Retrieving information from human movement patterns. *Understanding events: From perception to action*, 4:308–334, 2008.

[5] L. Schmid, J. Delmerico, J. L. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8018–8024. IEEE, 2022.

[6] V. Yugay, T. Kersten, L. Carlone, T. Gevers, M. R. Oswald, and L. Schmid. Gaussian mapping for evolving scenes. *arXiv preprint arXiv:2506.06909*, 2025.

[7] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[8] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award.

[9] K. Li and M. Q.-H. Meng. Personalizing a service robot by learning human habits from behavioral footprints. *Engineering*, 1(1):079–084, 2015.

[10] B. Irfan, A. Ramachandran, S. Spaulding, D. F. Glas, I. Leite, and K. L. Koay. Personalization in long-term human-robot interaction. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 685–686. IEEE, 2019.

[11] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *2012 IEEE international conference on robotics and automation*, pages 842–849. IEEE, 2012.

[12] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38 (1):14–29, 2015.

[13] J. Baraglia, M. Cakmak, Y. Nagai, R. P. Rao, and M. Asada. Efficient human-robot collaboration: when should a robot take initiative? *The International Journal of Robotics Research*, 36 (5-7):563–579, 2017.

[14] G. Hoffman and C. Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 1–8, 2007.

[15] S. Buyukgoz, J. Grosinger, M. Chetouani, and A. Saffiotti. Two ways to make your robot proactive: Reasoning about human intentions or reasoning about possible futures. *Frontiers in Robotics and AI*, 9:929267, 2022.

[16] M. K. van Den Broek and T. B. Moeslund. What is proactive human-robot interaction?-a review of a progressive field and its definitions. *ACM Transactions on Human-Robot Interaction*, 13(4):1–30, 2024.

[17] M. Patel and S. Chernova. Proactive robot assistance via spatio-temporal object modeling. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 881–891. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/patel23a.html.

[18] V. S. Dorbala, J. F. Mullen, and D. Manocha. Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation. *IEEE Robotics and Automation Letters*, 9(5): 4083–4090, 2023.

[19] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 464–474, 2024.

[20] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. ieee, 2017.

[21] V. S. Dorbala, B. Patel, A. S. Bedi, and D. Manocha. Personalized embodied navigation for portable object finding, 2026. URL https://arxiv.org/abs/2403.09905.

[22] 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.

[23] 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*, pages 17976–17993. PMLR, 2023.

[24] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[25] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[26] Z. Hou, T. Zhang, Y. Xiong, H. Pu, C. Zhao, R. Tong, Y. Qiao, J. Dai, and Y. Chen. Diffusion transformer policy. *arXiv preprint arXiv:2410.15959*, 2024.

[27] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. In *Conference on Robot Learning*, 2025.

[28] F. Zhang and M. Gienger. Affordance-based robot manipulation with flow matching. *arXiv preprint arXiv:2409.01083*, 2024.

[29] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[30] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[31] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Q. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

[32] P. Holderrieth and E. Erives. Introduction to flow matching and diffusion models, 2026. URL https://diffusion.csail.mit.edu/.

[33] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15791–15801, 2025.

[34] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 10–21, 2016.

[35] J. Wald, H. Dhamo, N. Navab, and F. Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020.

[36] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[37] A. Tong, K. FATRAS, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

[38] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28100–28127. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/pooladian23a.html.

[39] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.

[40] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. Reliable fidelity and diversity metrics for generative models. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7176–7185. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/naeem20a.html.

[41] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[42] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[43] 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 preprint arXiv:1807.06757*, 2018.