EgoEnv: Human-centric environment representations from egocentric video

Tushar Nagarajan[2], **Santhosh Kumar Ramakrishnan**[1], **Ruta Desai**[2],
James Hillis[2], **Kristen Grauman**[1,2]
[1]University of Texas at Austin, [2]FAIR, Meta

Abstract

First-person video highlights a camera-wearer's activities *in the context of their persistent environment*. However, current video understanding approaches reason over visual features from short video clips that are detached from the underlying physical space and capture only what is immediately visible. To facilitate human-centric environment understanding, we present an approach that links egocentric video and the environment by learning representations that are predictive of the camera-wearer's (potentially unseen) local surroundings. We train such models using videos from agents in simulated 3D environments where the environment is fully observable, and test them on human-captured real-world videos from unseen environments. On two human-centric video tasks, we show that models equipped with our environment-aware features consistently outperform their counterparts with traditional clip features. Moreover, despite being trained exclusively on simulated videos, our approach successfully handles real-world videos from HouseTours and Ego4D, and achieves state-of-the-art results on the Ego4D NLQ challenge. Project page: https://vision.cs.utexas.edu/projects/ego-env/

1 Introduction

Egocentric video offers a unique view into human activities through the eyes of a camera-wearer. Understanding this type of video is core to building augmented reality (AR) applications that can provide context-relevant assistance to humans based on their activity. Ego-video is thus the subject of several recent datasets and benchmarks that are driving new research [47, 75, 13, 26].

A key feature of the egocentric setting is the tight coupling of a camera-wearer and their persistent physical environment, i.e., a person's mental model of their surroundings informs their actions. This mental model is important, for example, to reach for a cabinet door out of view, to re-visit the couch to search for a misplaced phone or to visit spaces configured to support certain activities. This raises an important need for human-centric environment understanding — to learn representations from video that capture the camera-wearer's activities *in the context of their environment*. Such representations would encode the human-environment link, and allow models to jointly reason about both (e.g., to answer "what did the person cut near the sink?"). See Fig. 1.

Despite its importance, there has been only limited work on learning human-centric environment representations. Current video models segment a video into short clips (1-2s long) and then aggregate clip features over time (e.g., with recurrent, graph or transformer-based networks) for tasks like action forecasting [24, 26, 57, 25], temporal action localization [95, 50, 51, 96], episodic memory [26, 15], and movie understanding [90]. Critically, the clip features encode what is immediately visible in a short time window, and their aggregation over *time* does not equate to linking them in *physical space*. Other approaches use explicit camera pose information (e.g., from SLAM) to localize the camera-wearer, but not its relation to surrounding objects (e.g., forecasting [58, 68, 27]) or to group activities by location, but do not learn representations for agent video (e.g., affordance prediction [67, 57]).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

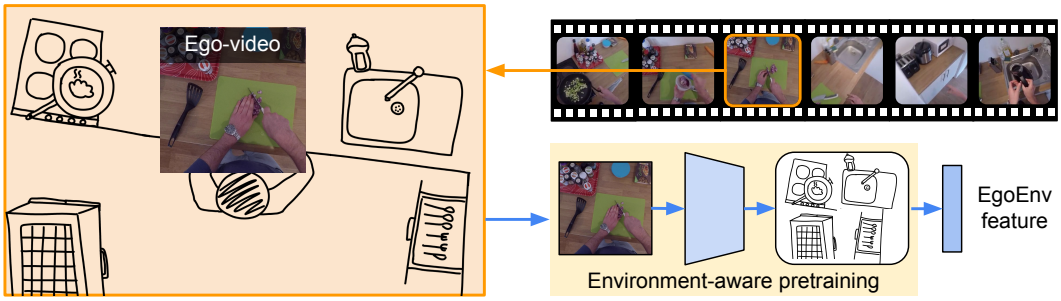


Figure 1: Main idea. Video models trained on short egocentric clips capture a narrow, instantaneous view of human activity (e.g., cutting an onion at the counter) without considering the broader context to which the activity is tied (e.g., the pan to the left to cook the onions, the fridge further behind to store leftovers). We propose to ground video in its underlying 3D environment by learning representations that are predictive of their surroundings, thereby enhancing standard clip-based models with complementary *environment* information.

To address these shortcomings, we propose to learn environment-aware video representations that encode the surrounding physical space. Specifically, we define the *local environment state* at each time-step of an egocentric video as the set of objects (and their rough distance) in front, to the left, right, and behind the camera-wearer. See Fig. 2. We use this state as supervision to train a transformer-based video encoder model that aggregates visual information across a video to build an *environment memory*, which can be queried to predict the local state at any point in the video.

The local state captures the rough layout of objects relative to the camera-wearer. It is important for understanding physical space — it provides a semantic signal to localize the camera-wearer (e.g., in the living room, from the arrangement of couches, lamps and tables) — as well as human behavior, since people move towards layouts that support activities (e.g., stove-top areas, dressers). Predicting the local state thus involves capturing the natural statistics of object layouts across different homes and then translating contexts across environments to reason about new ones. Once trained, given an observation from a new video, our model produces a drop-in EgoEnv feature which encodes *environment* information to complement the *action* information in existing video clip features.

An important practical question is how to supervise such a representation. Sourcing local state labels requires agent and object positions and omnidirectional visibility at each time step. This is challenging as egocentric videos only offer sparse coverage of the environment. Furthermore, they are prone to object detection, tracking, and SLAM failures. Therefore, for training we turn to videos generated by agents in simulation. This allows us to sample diverse, large-scale trajectories to cover the environment, while also providing ground-truth local state. Once trained with simulated video, we apply our models to *real-world* videos from new, unseen environments.

We demonstrate our EgoEnv approach on two video tasks where joint reasoning of both human action and the underlying physical space is required: (1) inferring the room category that the camera wearer is physically in as they move through their environment, and (2) localizing the answer to a natural language query in an egocentric video. These tasks support many potential applications, including AR systems that can offer context-relevant assistance.

We are the first to demonstrate the value of 3D simulation data for real-world ego-video understanding. Our experiments show that by capitalizing on both geometric and semantic cues in our proposed "local environment state" task, we can leverage video walkthroughs from *simulated agents* in HM3D scenes [64] to ultimately enable downstream human-centric environment models on *real-world* videos from HouseTours [7] and Ego4D [26]. Furthermore, models equipped with our EgoEnv features outperform both popular scene classification [97] and natural language video localization models [95], and achieve state-of-the-art results on the Ego4D NLQ challenge leaderboard.

2 Related work

Video understanding in 3D environments. Prior work encodes short video clips [4, 85, 60, 33] or temporally aggregates them for additional context [89, 24, 95, 50, 51, 96, 90]. However, these methods treat the video as a temporal sequence and fail to capture the spatial context from the underlying persistent environment. For egocentric video, prior work has used structure from motion (SfM) to map people and objects for trajectory [58] and activity forecasting [27] and action grounding

in 3D [67, 14]. These approaches localize the camera-wearer but do not learn representations for the camera-wearer's surroundings. The model of [52] associates features to voxel maps to localize actions; however, they require a pre-computed 3D scan of the environment at both training and inference. Prior work groups clips by rough spatial location as topological graphs [57] or activity threads [61], but they stop short of learning representations using these groups. In contrast, we explicitly learn features that relate clips based on their spatial layout, for each step of an ego-video.

Video representation learning. Traditional video understanding methods learn representations by training models on large, manually curated video action recognition datasets [4, 54, 26]. Recent self-supervised learning (SSL) approaches eliminate the supervision requirement by leveraging implicit temporal signals [88, 55, 81, 40, 31, 84, 86, 87, 66, 23]. In contrast, we learn features that encode the local spatial-state of the environment (as opposed to temporal signals). Further, we show how to leverage state information that is readily accessible in simulation, but not in video datasets (i.e., locations and semantic classes of objects surrounding the agent) for training.

Environment features for embodied AI. In embodied AI, pose-estimates are used to build maps [11, 8, 62], as edge features in graphs [9, 7], as spatial embeddings for episodic memories [20], to project features to a grid map [28, 36, 5] or to learn environment features for visual navigation [65]. However, these approaches are explored solely in simulation and typically require accurate pose-estimates or smooth action spaces, and thus are not directly applicable to egocentric videos. Research on world-models [18, 29, 30] and unseen panorama reconstruction [41, 76, 44] *hallucinate* the effect of agent actions to aid decision-making in simulation. In contrast, we aim to learn environment features for an egocentric camera-wearer to aid real-world video understanding.

Learning from simulated data. Prior work has proposed cost-effective ways to generate large-scale synthetic image datasets for various vision tasks [77, 79, 16, 37, 19, 70]. In robotics, simulation environments have been developed to quickly and safely train policies, with the eventual goal of transferring them to real world applications [43, 80, 45, 6, 91, 72, 64, 78]. The resulting *sim-to-real* problem, where models must adapt to changes in simulator and real-world domains, is an active area of research for robot navigation [79, 42, 71, 3, 1]. However, simulated data for video understanding is much less explored. Prior work has synthesized data for human body pose estimation [12, 82, 93, 17], trajectory forecasting [48], and action recognition [69]. Rather than model human behavior, our approach is the first to directly capture the environment surrounding the camera-wearer for real-world video understanding tasks.

3 Approach

Our goal is to learn EgoEnv representations that encode the local surroundings of the camera-wearer. Such a representation would implicitly maintain a semantic memory of surrounding objects beyond what is immediately visible, and coupled with a standard video feature, would allow models to jointly reason about activities and the underlying physical space. Directly appending the camera pose with each video frame may capture the local state; however noise in pose estimated from ego-video with quick head motions and characteristic blur limits its utility (see Supp. for experiments).

Instead, we introduce an approach that leverages simulated environments where perfect state information is available to train models that can link visual information to the physical surroundings. To this end, we first define the local state prediction task in simulation (Sec. 3.1). Next, we introduce our transformer-based architecture that predicts the local state in videos (Sec. 3.2). Finally, we show how our model trained in simulation generates *environment features* for real-world egocentric video frames (Sec. 3.3).

3.1 Local environment state

We require a model that is aware of not just what is immediately visible in a single frame, but also of the camera-wearer's surroundings. We therefore define the *local environment state* of the camera-wearer as the set of objects in each relative direction — i.e., what objects are to the front, left, right or behind the camera-wearer, together with their rough distance from the camera-wearer — and train a model to predict this state. Our definition of local state takes inspiration from cognitive science [32, 46], and offers supervision signals that are both geometric (relative object locations) and semantic (semantic object labels), which we observe leads to strong representations.

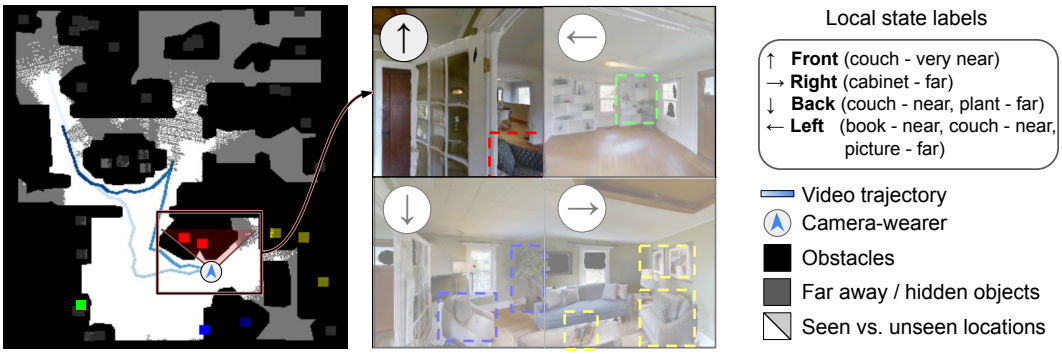


Figure 2: Local environment state task. Left panel: Top-down environment view showing the camera-wearer path (blue gradient line) and nearby objects (colored boxes). **Middle panel:** Egocentric view in each direction. Only the forward view (top left) is observed. Remaining views are shown for clarity. **Task:** Given an egocentric video of a person in their environment, the model must predict the set of objects (and their rough distance) in front, to the left, right, and behind the camera-wearer at each time-step (right panel). The model only sees the forward ego-view (middle panel, top left) and does not have access to the top-down map. Note that not all parts of the environment are seen during a walkthrough (white vs. grey regions on map) — models must link seen observations based on their shared space, as well as anticipate unseen surroundings based on statistics of training environments. Best viewed in color.

More formally, let $\mathcal{O}$ be a set of object classes. For a frame f from a video trajectory in an environment, the local state is a tuple (y_o, y_r). y_o is a $4 \times |\mathcal{O}|$ dimensional matrix which represents instances of each object class in the four cardinal directions relative to the camera-wearer. y_r is a matrix of the same size containing the distance of the objects in y_o from the camera-wearer, discretized into 5 distance ranges between $0.25 - 5.0m$[1]. For direction i and object class j, the labels are:

$$y_o[i,j] = \begin{cases} 1 & \text{if } d(p_a, p_j) < \delta \wedge \theta(p_a, p_j) = i \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

$$y_r[i,j] = \bar{d}(p_a, p_j) \quad \text{if } y_o[i,j] = 1, \quad (2)$$

where p_a, p_j are the poses of the camera-wearer and object o_j respectively, $d(p_a, p_j)$ is the euclidean distance between them, δ is a distance threshold for nearby objects (we set $\delta = 5.0$m, beyond which visible objects are small), $\theta(p_a, p_j) \in [0...3]$ is the discretized angle of the object relative to the agent's heading (forward, right, behind, left), and $\bar{d}(p_a, p_j) \in [0...4]$ is discretized distance. See Fig. 2 and Supp. for empirical analysis of related alternatives, e.g., predicting just object presence or image features.

We then train a model to predict the local state of the target video frame, conditioned on the video trajectory. Once trained, the model can relate what is visible in a frame to the camera-wearer's possibly *hidden* surroundings to produce environment-aware features.

Since supervision for camera-wearer pose and every object's location is non-trivial for egocentric videos, where camera localization and tracking is error prone, we leverage videos in simulated environments for training, as presented next.

3.2 Environment-aware pretraining in simulation

To source local state labels, we generate a dataset of video walkthroughs of agents in simulated 3D environments where agent and object poses are accessible at all times (see Sec. 4). We train our model in two stages described below.

3.2.1 Pose embedding learning

While ground truth camera pose is available from the simulator at training time, a model trained to rely on it will fail on real-world egocentric video at test time, where pose estimates are noisy. With the goal of handling arbitrary indoor egocentric video, we instead explore representations that implicitly encode coarse pose information. See Supp. for pose-related experiments.

[1]For object classes with multiple instances, we select the nearest one.

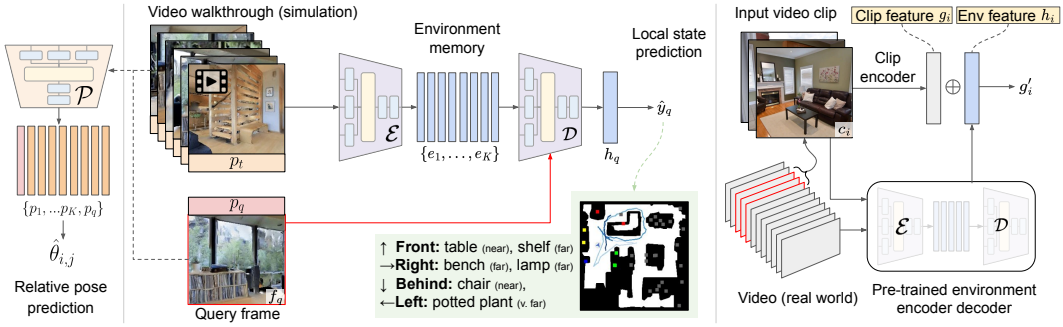


Figure 3: Model framework. Left: Our model first learns pose embeddings by predicting discretized relative pose between observations from simulated video walkthroughs (Sec. 3.2.1). **Center:** Next, it encodes observations and their pose embeddings into an *environment memory* that is trained to predict the local environment state for a query frame (Sec. 3.2.2). **Right:** Once trained, our model builds and queries the environment memory for any time-point of interest in a real-world video, to generate an environment feature for downstream video tasks in disjoint and novel scenes (Sec. 3.3). $\oplus$ = concatenation.

Specifically, for a sequence of RGB frames $\{f_t\}_{t=1}^T$ and camera poses $\{\theta_t\}_{t=1}^T$, we generate *pose embeddings* $\{p_t\}_{t=1}^T = \mathcal{P}(f_1, ..., f_T)$ using a transformer encoder network. These embeddings are used to predict the relative pose between each observation pair using a bilinear layer

$$\hat{\theta}_{i,j} = p_i^T V_p p_j + W_p^T (p_j - p_i) + b_p, \tag{3}$$

where $\hat{\theta}_{i,j}$ is the predicted relative pose and V_p, W_p, b_p are trainable parameters. We discretize the relative pose into 12 angles and 4 distance ranges to provide an approximate yet robust pose estimate. The network is trained to minimize cross-entropy between the predicted and the target relative pose labels for all observation pairs $\sum_{i,j} \mathcal{L}_{ce}(\hat{\theta}_{i,j}, \theta_{i,j})$. The trained pose embeddings p_t encode information to help relate video observations based on their location and orientation in the environment. Note that once trained, pose embeddings are inferred directly from video frame sequences — ground truth pose is only required for training.

3.2.2 Local state pretraining

Next, we train a model to embed visual information from a video walkthrough into an environment memory, which can then be queried to infer the local state corresponding to a given video frame from the same video. We implement this model as a transformer encoder-decoder model.

Specifically, for a video walkthrough $\mathcal{V}$ with RGB frames $\{f_t\}_{t=1}^T$, and a query frame f_q, we predict the local state $y_q = (y_o, y_r)$ as follows. First, pose embeddings $\{p_t\}_{t=1}^T$ are generated for the video and query frames following Sec. 3.2.1. Then, each frame is encoded jointly with the pose embedding using a linear transform $\mathcal{M}_p$.

$$x_t = \mathcal{M}_p([f_t; p_t]). \tag{4}$$

Next, we uniformly sample K video frames to construct an environment memory using a transformer encoder $\mathcal{E}$, which updates frame representations using self-attention:

$$\{e_1, ..., e_K\} = \mathcal{E}(x_1, ..., x_K). \tag{5}$$

The resulting memory represents features for each time-step that contain propagated information from all other time-steps. Compared to prior work [24, 95, 50, 51, 96], our encoder has the ability to relate observations based on not just visual characteristics and their temporal ordering, but also their relative spatial layout in the environment. A transformer decoder $\mathcal{D}$ then attends over the memory using query x_q to produce the output EgoEnv representation h_q:

$$h_q = \mathcal{D}(\{e_1, ..., e_K\}, x_q), \tag{6}$$

which is finally used to predict the local state using two linear classifiers $\mathcal{M}_o$ and $\mathcal{M}_r$ for object class and distance predictions respectively. The network is trained to minimize the combination of losses over the predicted and the target state labels for each direction:

$$\mathcal{L}(h_q, y_o, y_r) = \mathcal{L}_{bce}(\mathcal{M}_o(h_q), y_o) + \lambda \mathcal{L}_{ce}(\mathcal{M}_r(h_q), y_r),$$

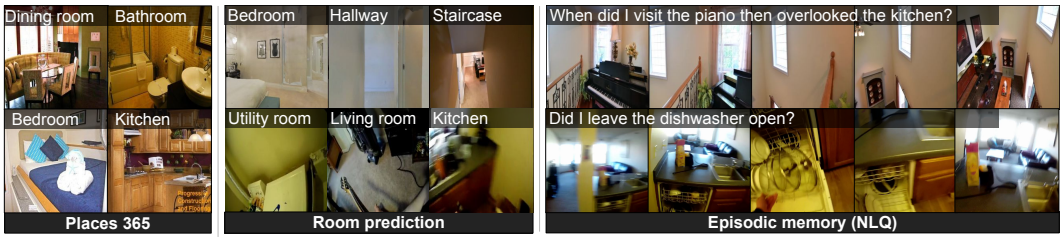


Figure 4: Scene understanding in third-person photos vs. human-centric environment understanding. Left: Well-framed, canonical images from Places365 are considerably different from scene content observed in egocentric video. **Center and right:** Real egocentric video streams from HouseTours (top) and Ego4D (bottom) illustrating the value in modeling the underlying environment, rather than just what is visible in short clips. For example, the person does not explicitly look at the staircase while walking down it (center, top row); the spatial relation between the person, piano, and kitchen is important to answer the question (right, top row).

where $\mathcal{L}_{bce}$ and $\mathcal{L}_{ce}$ refer to binary cross-entropy and cross-entropy losses respectively. We set $\lambda = 0.1$ which balances the contributions of each loss function based on validation experiments. The distance loss is computed only for objects that are in the local state ($y_o = 1$). See Fig. 3 (left) and Supp. for more architecture details.

Learning to predict the local state involves aggregating information about observed objects across time, as well as anticipating unseen objects based on learned priors from the layout of objects in other scenes (e.g., TVs are usually in front of couches; kitchens have particular arrangements of sinks, refrigerators, and stove-tops).

Once trained, given a video in a new environment and a time-point of interest, our model constructs an environment memory, predicting the local state based on information aggregated throughout the test video. Importantly, h_q — the EgoEnv feature — contains valuable information about the camera-wearer's surroundings, offering environment features to complement traditional video features (e.g., for a person watching TV, also encode the couch they are sitting on, the lamp nearby).

3.3 Environment-memory for video understanding with real videos

Next, we leverage our environment-memory model for real-world video tasks. A video understanding task defines a mapping from a sequence of video clips $\{c_1, ..., c_N\}$ from longer video $\mathcal{V}$ to a task label. We consider two tasks: (1) **RoomPred**: where the model must classify which room r_t the camera-wearer is in (e.g., living room, kitchen) at time t in the video, and (2) **NLQ**: natural language queries, an episodic memory task popularized recently in Ego4D [26] where the model must identify the temporal window (t_s, t_e) in the video that answers an environment-centric query q specified in natural language. See Fig. 4. Both tasks entail human-centric spatial reasoning from video.

Current models produce clip features that encode only what is immediately visible. This is sufficient for short-horizon tasks (e.g., action recognition), but as we will show, falls short on the tasks above that require extra reasoning about the agent's surroundings. Our environment-memory model addresses this by enhancing standard clip features with context from the camera-wearer's surroundings.

To do this, for each input clip in $c_i \in \mathcal{V}$, we select the center frame f_i of the clip as the query frame. Following Sec. 3.2, we uniformly sample K frames from the video around the query frame, encode them along with their pose embeddings (Eqn. 4), and build an environment memory using our environment encoder $\mathcal{E}$ (Eqn. 5). Finally, we use our decoder $\mathcal{D}$ to produce the output feature. This results in set of output EgoEnv features, one per input clip.

$$h_i = \mathcal{D}(\left\{e_1^i, ..., e_K^i\right\}, x_i). \tag{7}$$

Each environment feature then enhances the original clip feature as follows

$$g_i' = W_{\mathcal{E}}^T [g_i; h_i] + b_{\mathcal{E}}, \tag{8}$$

where g_i is the original clip feature for clip c_i (e.g., ResNet, SlowFast, EgoVLP) and $W_{\mathcal{E}}, b_{\mathcal{E}}$ are linear transform parameters. See Fig. 3 (right).

The new clip features $\{g_1', ..., g_N'\}$ consolidate features from what is directly visible in a short video clip and features of the (potentially unseen) space surrounding the camera-wearer. Put simply, our approach implicitly widens the field of view for tasks that reason about short video clips by providing a way to access features of their surroundings in a persistent, geometrically consistent manner.

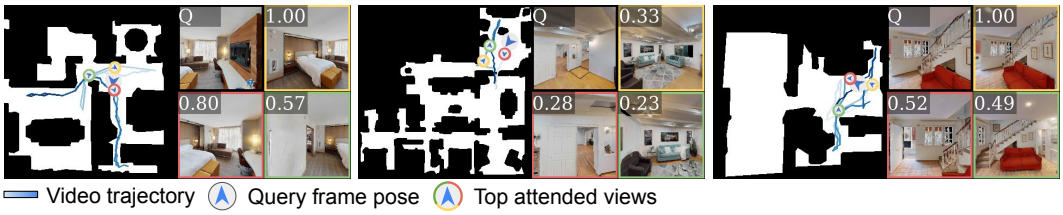


Figure 5: Visualized attention weights. Our model learns to attend to views that help solve the local state prediction task. The query frame Q and top-3 attended views (colored boxes), their positions along the trajectory (colored circles), and their associated attention scores are shown. See Supp. for more examples.

Critically, we use the exact same EgoEnv representations to tackle both video understanding tasks. This is a departure from traditional sim-to-real approaches where a task-specific dataset needs to be carefully designed for every downstream task, which may be impractical. That said, given that today's 3D assets (Matterport3D, HM3D, etc.) focus on indoor spaces, our model is best suited to videos in indoor environments. We discuss the sim-to-real gap in detail in Supp.

4 Experiments

We evaluate how our EgoEnv features learned in simulation benefit real-world video understanding.

Simulator environments For training, we use the Habitat simulator [72] with photo-realistic HM3D [64] scenes to generate simulated video walkthroughs. We generate ~15k walkthroughs from 900 HM3D scenes, each 512 steps long, taken by a shortest-path agent that navigates to randomly sampled goal locations (move forward, turn right/left 30°). For each time-step, we obtain the ground-truth local state from the simulator required in Sec. 3.1 (i.e., object labels and relative pose). For object labels, we map instance predictions across $|\mathcal{O}| = 23$ categories to the 3D scenes using a pretrained instance segmentation model [21] trained on the subset of scenes with semantic labels. Though the walkthroughs involve discrete actions, they share characteristics with real-world video (cameras at head-level; views covering the environment) making them suitable for transfer. See Supp. for examples and details.

Video datasets We evaluate our models on three egocentric video sources. (1) **HouseTours [7]** contains 119 hours of real-world video footage of house tours from YouTube. We use ~32 hours of video from 886 houses where the camera can be localized and create data splits based on houses. (2) **Ego4D [26]** contains 3k hours of real-world video of people performing daily activities. We use all videos annotated for the NLQ benchmark and apply the provided data splits, which yields 1,259 unseen scenes. (3) **Matterport3D (MP3D) [6]** contains simulated video walkthroughs from 90 photo-realistic 3D scenes. We use 146 long video walkthroughs and standard data splits [5].

These datasets provide an ideal test-bed for our approach. On the one hand, both HouseTours and Ego4D are *real-world* video datasets allowing us to test generalization to both real-world visuals as well as natural human activity across diverse, cluttered environments in unseen houses. On the other hand, MP3D offers novel scenes with distinct visual characteristics and object distributions compared to HM3D, allowing us to test our model's robustness to domain shift in a controlled *simulated* video setting. Note that none of the datasets have a 1-1 alignment in object taxonomy with HM3D, meaning our downstream tasks require generalization to both unseen environments and unseen objects.

We collect crowd-sourced labels for each task, which we will publicly share. For ROOMPRED these are room category labels from 21 classes (e.g., living room, kitchen) for each time-step on HouseTours and Ego4D. For NLQ these are natural language queries and corresponding response tracks in the video. On HouseTours, we crowd-source queries (e.g., "where did I last see my phone in the kitchen", "when did I first visit the bathtub") and on Ego4D, we use the official NLQ benchmark annotations, which require reasoning over actions, objects, and locations (e.g., "what tool did I pick up from the table", "where did I hang the pink cloth"). On MP3D, we source all labels directly from the simulator (9 room categories, and automatically generated NLQ queries from simulator object labels and locations). See Supp. for data collection details and Fig. 4 for examples.

Experiment setup For pre-training, we use 2048-D ImageNet-pretrained ResNet50 [35] features for each video frame. Our encoder-decoder models $\mathcal{P}, \mathcal{E}, \mathcal{D}$ are 2-layer transformers [83] with hidden dimension 128. $K = 64$ frames are sampled from each video to populate the memory. We train

models for 2.5k epochs and select the model with the lowest validation loss. For RoomPred, we generate a single EgoEnv feature aligned with the query clip. For NLQ we generate one feature per input clip. Full architecture and training details are in Supp.

Baselines For RoomPred we use a popular scene recognition model PlacesCNN [97] as the baseline model. For NLQ we use the state-of-the-art moment localization model VSLNet [94, 49, 53]. Within these two frameworks, we compare the following approaches to enhance clip representations: **FrameFeat** uses a pretrained ResNet50 [35] model to generate a frame feature corresponding to each clip. **ObjFeat** trains an object detector on all available simulated HM3D data and generates backbone features for each clip. We use the QueryInst model [21]. **MAE [34]** trains a state-of-the-art self-supervised learning approach to reconstruct masked patches of walkthrough video frames. **EgoTopo [57]** trains a graph convolutional network (GCN) over the video graph built following [57]. **EPC [65]** trains an environment memory model to predict masked *zone* features conditioned on pose. **TRF (scratch)** trains a scene-memory transformer model [20] that shares our model architecture but is randomly initialized and fine-tuned for the task.

These baselines represent various strategies to incorporate environment information into clip representations ranging from frame features (MAE, FrameFeat, ObjFeat), to topological graph-based features (EgoTopo), to pose-based features (EPC). Note that ObjFeat, MAE and EPC all pre-train on the same walkthrough videos as our approach. ObjFeat further benefits from ground-truth object labels from the simulator. Features from these approaches augment the input clip representations following Eqn. 8 — baseline architectures remain unchanged. Note that EPC requires privileged information—ground-truth camera poses at inference time—whereas our model does not.

4.1 Pose embedding and local state pretraining

We begin by evaluating the pose embedding network trained to predict relative pose discretized into 12 angles and 4 distance ranges. On the validation set, the model achieves accuracies of 48.4% on relative distance prediction and 34.4% on relative orientation prediction. Note that this task is challenging — models must predict relative pose for all possible pairs of observations in a trajectory using their visual features alone — however the goal is to generate pose encodings, not to output perfect pose. Next, we evaluate how well our model can infer the local state, reporting average precision (AP) in each direction. Given a *forward* view, objects can be reliably recognized (37.8 AP) compared to naively outputting the distribution of objects seen at training (5.4 AP). Moreover, our approach can link views in the video trajectory to also infer and anticipate objects in other directions, i.e., to the right, left and behind (21.5, 24.9 and 20.2 AP respectively) given the forward view. We visualize the attention weights learned by our model to link relevant observations to the query in Fig. 5. Our model learns to select informative views for the task beyond just temporally adjacent frames or views with high visual overlap. For example, in the first image, the view with highest attention score (1.0) looks at the bed directly *to the left* (yellow box), allowing our model to benefit from information beyond its field of view.

4.2 EgoEnv features for room prediction

Next, we evaluate our method on predicting what room the camera-wearer visits in the video. All models have access to the full video, but inference is at at each time-step. The PlacesCNN model is a 2-layer MLP classifier trained on features from a Places365 [97] scene classification model. Features are max-pooled across a clip of $N = 8$ frames around the time-step of interest for additional context, as a single frame may be uninformative (e.g., facing a wall). We generate an environment feature aligned with the center of the clip.

Fig. 6 shows the results. We report top-1 accuracy as a function of dataset difficulty, measured by the prediction entropy of the Places365 model trained on canonical scene images. Instances are "hard" (high entropy) where frame-level information is insufficient for predicting the room type. Accuracy on the full dataset is at the far-right of the plots. All models perform better on HouseTours compared to MP3D and Ego4D since the house tours were captured explicitly to provide informative views of each room. Despite having access to all additional pre-training videos and labels as supervision, frame-level features from ObjFeat and MAE prove to be insufficient for environment-level reasoning. All methods except ours perform worse with the introduction of hard instances where the surrounding environment-context is important (left to right). This is a key result: despite training our models entirely in simulation, and with videos from a set of disjoint environments, our EgoEnv features are useful for downstream tasks on real-world videos.

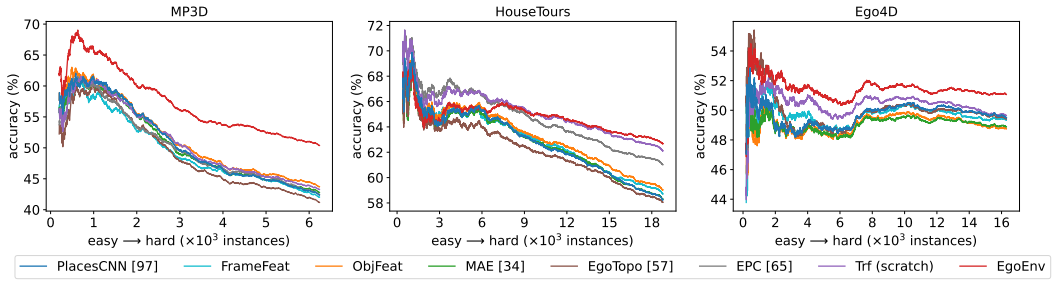


Figure 6: RoomPred accuracy by instance difficulty. Our method outperforms baselines, especially on *hard* instances (smaller performance drop along curve). EPC requires pose at inference, which is unavailable in Ego4D. Results are aggregated across three runs. See Supp. for averaged results with error bars.

	MP3D [6]			HouseTours [7]			Ego4D‡ [26]		
Rank1@m →	@0.3	@0.5	AVG	@0.3	@0.5	AVG	@0.3	@0.5	AVG
VSLNet [94]	33.69	22.83	28.26	42.94	27.68	35.31	5.45	3.12	4.29
FrameFeat	35.23	24.57	29.90	48.45	32.06	40.25	5.58	3.28	4.43
ObjFeat	37.20	26.33	31.76	47.74	32.49	40.11	5.76	3.43	4.59
MAE [34]	35.11	24.13	29.62	44.49	27.82	36.16	5.65	3.02	4.34
EgoTopo [57]	36.10	25.06	30.58	43.36	27.97	35.66	5.45	3.19	4.32
EPC† [65]	36.85	**27.64**	32.24	43.22	27.82	35.52	-	-	-
Trf (scratch)	34.18	24.65	29.42	40.54	22.32	31.43	5.25	3.12	4.19
EgoEnv	**38.18**	26.85	**32.52**	**51.98**	**34.18**	**43.08**	**6.04**	**3.51**	**4.77**

Table 1: NLQ results. All baselines build on VSLNet [95] with alternate features. †Privileged access to pose at inference, unavailable to our model, and absent in Ego4D. ‡Validation split. See Table 2 for test set results.

4.3 EgoEnv features for episodic memory

Next we evaluate on localizing the responses to natural language queries in egocentric video. We use VSLNet [94] and provide it with $N = 128$ clips sampled uniformly from the full video to generate predictions. We use SlowFast [22] clip features and generate environment features aligned with each input clip. We use the benchmark-provided metric of *Rank n@m*, which measures temporal localization accuracy [26].

Table 1 shows the NLQ results. Similar to RoomPred, instances are harder in MP3D than in HouseTours as MP3D's shortest-path agents produce moments that are quick transitions between objects and locations, and contain only short glimpses of them. The global, video-level information from EgoTopo improves performance slightly on all datasets. Strong image-level supervision (object labels in ObjFeat and FrameFeat) results in the largest improvements; MAE, which has access to the same data but trains self-supervised representations, does not show strong improvements. EPC performs well on MP3D, where accurate pose is available from the simulator, but not on HouseTours with only noisy pose estimates (see Supp.). Our EgoEnv approach performs the best overall, outperforming even EPC, which (unlike EgoEnv) has access to ground-truth pose at inference.

Finally, Table 2 shows official eval server results for the Ego4D NLQ benchmark. We augment the baseline [63] with our EgoEnv features. We achieve state-of-the-art results, ranking 1st on the public leaderboard at the time of submission, and currently ranked 3rd. Note that Ego4D contains in-the-wild videos of natural human activity in diverse scenes (e.g., workshops, gardens) compared to the simulated walkthrough videos in pretraining. Due to this *sim-to-real* gap, our approach performs even better on instances aligned with training videos (navigation in indoor homes) as our Supp. experiments show. Our leading results on the full challenge set for this major benchmark demonstrates the value of our environment-centric feature learning approach, despite this gap.

4.4 Analysis of sim-to-real gap

Next, we discuss our approach in the context of the sim-to-real gap. Ego4D videos are in-the-wild, capture natural human actions and object-interactions, and take place in diverse scenes. These scenes may be significantly different from the simulated environments used for pre-training (navigation in indoor houses). In Table 3, we show results on the subset of videos that are aligned with the training environments on the Ego4D NLQ validation set. We select these using the scenario labels provided in Ego4D including indoor home scenarios (e.g., listening to music, household management)

RANK 1@M →	@0.3	@0.5	AVG
CONE [39]	15.26	9.24	12.25
BADGERS@UW-MAD. [56]	15.71	9.57	12.64
INTERNVIDEO [10]	16.46	10.06	13.26
NAQ [63]	21.70	13.64	17.67
EGOENV	**23.28**	**14.36**	**18.82**

Table 2: Ego4D NLQ challenge results. Our model obtains the best results against published approaches, and ranks 3rd among concurrent, unpublished work [38, 74].

and navigation-heavy scenarios (walking indoors and outdoors) while excluding outdoor activities (e.g., golfing, outdoor cooking). See Supp. for the full list of scenarios. Our approach shows healthier improvements across both sets of scenarios, highlighting the effect of the sim-to-real gap. However, despite this gap, our approach is still able to outperform other approaches over all scenarios, demonstrating the value of our environment-centric feature learning approach.

RANK K@M →	R1@0.3	R1@0.5	AVG
NAQ (ALL) [63]	24.12	15.04	19.58
EGOENV (ALL)	**25.37**	**15.33**	**20.35**
NAQ (INDOOR) [63]	28.91	17.97	23.44
EGOENV (INDOOR)	**31.22**	**19.09**	**25.16**
NAQ (NAV) [63]	23.58	15.49	19.53
EGOENV (NAV)	**26.16**	**16.01**	**21.08**

Table 3: Ego4D NLQ validation set results on aligned scenes. Our approach performs better on the subset of videos that are aligned with our approach's simulated training environments (navigation, indoor houses).

4.5 Ablation experiments

Next, we discuss some important ablations of our model design. We present full details of these experiments in Supp E and F, but we discuss the main conclusions here.

Importance of pose information: We measure the effect of pose embeddings on our local state prediction task. Our models show small improvements in predicting objects in the forward view (+0.7 mAP) where scene information is directly visible, but large for other views that need to be inferred: mAP improvements of +2.2 (right), +0.9 (behind) and +1.0 (left). Further, we directly embed ground-truth pose as part of the input and see benefits on both tasks on MP3D, but not on HouseTours, due to noise in extracted pose (compared to simulator-provided pose in MP3D).

Alternate pretraining task formulations: Next, we investigate alternate pretraining objectives instead of local state prediction including variants that only predict the object categories in each cardinal direction, but not the distances or that directly predicts the image features in each cardinal direction, among others. We find that our approach that requires predicting both object labels, orientations as well as rough distances offers a balance of both cues during pretraining, translating to strong downstream performance.

Hyperparameter ablations: We vary window size, memory size and the loss weighting term. We find that small window sizes are sufficient for localizing the room category for ROOMPRED, while larger windows are required for NLQ; and that the model is not very sensitive to the other parameters.

5 Conclusion

We proposed a framework to learn environment-aware representations in simulation and transfer them to video understanding tasks on challenging real-world datasets. Our approach outperforms state-of-the-art representations for predicting visited rooms and retrieving important moments from natural language queries, despite a significant sim-to-real gap. Despite its strengths, there are several opportunities for future work to improve our model further. These include incorporating 3D information into the local state task (currently defined in a 2D, top-down map), generating more *human-like* simulated videos and integrating more explicit approaches to tackle the sim-to-real gap.

Acknowledgements Thanks to Fu-Jen Chu and Jiabo Hu for help collecting the HouseTours annotations. UT Austin is supported in part by IFML NSF AI institute. KG is paid as a research scientist at Meta.

References

[1] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *CoRL*, 2021. 3

[2] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. In *arXiv preprint arXiv:2006.13171*, 2020. 4

[3] Roberto Bigazzi, Federico Landi, Marcella Cornia, Silvia Cascianelli, Lorenzo Baraldi, and Rita Cucchiara. Out of the box: embodied navigation in the real world. In *CAIP*, 2021. 3

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 2, 3

[5] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views. In *AAAI*, 2021. 3, 7

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. MatterPort3D dataset license available at: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. 3, 7, 9, 11

[7] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In *NeurIPS*, 2020. 2, 3, 7, 9, 11

[8] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 3

[9] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 3

[10] Guo Chen, Sen Xing, Zhe Chen, Yi Wang, Kunchang Li, Yizhuo Li, Yi Liu, Jiahao Wang, Yin-Dong Zheng, Bingkun Huang, et al. Internvideo-ego4d: A pack of champion solutions to ego4d challenges. *arXiv preprint arXiv:2211.09529*, 2022. 10

[11] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *ICLR*, 2019. 3

[12] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3DV*, 2016. 3

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: collection, pipeline and challenges for epic-kitchens-100. In *IJCV*, 2022. 1

[14] Dima Damen, Teesid Leelasawassuk, and Walterio Mayol-Cuevas. You-do, i-learn: Egocentric unsupervised discovery of objects and their modes of interaction towards video-based guidance. In *CVIU*, 2016. 3

[15] Samyak Datta, Sameer Dharur, Vincent Cartillier, Ruta Desai, Mukul Khanna, Dhruv Batra, and Devi Parikh. Episodic memory question answering. In *CVPR*, 2022. 1

[16] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. In *arXiv preprint arXiv:1911.01911*, 2019. 3

[17] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. In *NeurIPS*, 2019. 3

[18] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *ICLR*, 2017. 3

[19] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, 2021. 3

[20] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, 2019. 3, 8

[21] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *ICCV*, 2021. 7, 8, 9

[22] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 9, 12

[23] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *CVPR*, 2021. 3

[24] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019. 1, 2, 5

[25] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *ICCV*, 2021. 1

[26] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen,

Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the World in 3,000 Hours of Egocentric Video. In *CVPR*, 2022. 1, 2, 3, 6, 7, 9, 4, 5, 10, 12

[27] Jiaqi Guan, Ye Yuan, Kris M Kitani, and Nicholas Rhinehart. Generative hybrid representations for activity forecasting with no-regret learning. In *CVPR*, 2020. 1, 2

[28] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 3

[29] David Ha and Jürgen Schmidhuber. World models. *NeurIPS*, 2018. 3

[30] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2019. 3

[31] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *CVPR Workshops*, 2019. 3

[32] Xue Han, Patrick Byrne, Michael Kahana, and Suzanna Becker. When do objects become landmarks? a vr study of the effect of task relevance on spatial memory. In *PloS one*, 2012. 3

[33] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018. 2

[34] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 8, 9

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7, 8

[36] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *CVPR*, 2018. 3

[37] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *ECCV Workshops*, 2018. 3

[38] Zhijian Hou, Lei Ji, Difei Gao, Wanjun Zhong, Kun Yan, Chao Li, Wing-Kwong Chan, Chong-Wah Ngo, Nan Duan, and Mike Zheng Shou. Groundnlq@ ego4d natural language queries challenge 2023. *arXiv preprint arXiv:2306.15255*, 2023. 10, 2

[39] Zhijian Hou, Wanjun Zhong, Lei Ji, Difei Gao, Kun Yan, Wing-Kwong Chan, Chong-Wah Ngo, Zheng Shou, and Nan Duan. An efficient coarse-to-fine alignment framework@ ego4d natural language queries challenge 2022. *arXiv preprint arXiv:2211.08776*, 2022. 10

[40] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 3

[41] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018. 3, 12

[42] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? In *RA-L*, 2020. 3

[43] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, 2004. 3

[44] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *ICCV*, 2021. 3, 12

[45] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. In *arXiv preprint arXiv:1712.05474*, 2017. 3

[46] Eric Krokos, Catherine Plaisant, and Amitabh Varshney. Virtual memory palaces: immersion aids recall. In *Virtual reality*, 2019. 3

[47] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *CVPR*, 2015. 1

[48] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Simaug: Learning robust representations from simulation for trajectory prediction. In *ECCV*, 2020. 3

[49] Kevin Qinghong Lin, Alex Jinpeng Wang, Mattia Soldan, Michael Wray, Rui Yan, Eric Zhongcong Xu, Difei Gao, Rongcheng Tu, Wenzhe Zhao, Weijie Kong, et al. Egocentric video-language pretraining@ ego4d challenge 2022. In *arXiv preprint arXiv:2207.01622*, 2022. 8, 10, 12

[50] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *ICCV*, 2019. 1, 2, 5

[51] Daizong Liu, Xiaoye Qu, Jianfeng Dong, Pan Zhou, Yu Cheng, Wei Wei, Zichuan Xu, and Yulai Xie. Context-aware biaffine localizing network for temporal sentence grounding. In *CVPR*, 2021. 1, 2, 5

[52] Miao Liu, Lingni Ma, Kiran Somasundaram, Yin Li, Kristen Grauman, James M Rehg, and Chao Li. Egocentric activity recognition and localization on a 3d map. In *ECCV*, 2022. 3

[53] Naiyuan Liu, Xiaohan Wang, Xiaobo Li, Yi Yang, and Yueting Zhuang. Reler@ zju-alibaba submission to the ego4d natural language queries challenge 2022. In *arXiv preprint arXiv:2207.00383*, 2022. 8, 10, 12

[54] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019. 3

[55] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 3

[56] Sicheng Mo, Fangzhou Mu, and Yin Li. A simple transformer-based model for ego4d natural language queries challenge. *arXiv preprint arXiv:2211.08704*, 2022. 10

[57] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *CVPR*, 2020. 1, 3, 8, 9

[58] Hyun Soo Park, Jyh-Jing Hwang, Yedong Niu, and Jianbo Shi. Egocentric future localization. In *CVPR*, 2016. 1, 2

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 2019. 8

[60] Fiora Pirri, Lorenzo Mauro, Edoardo Alati, Valsamis Ntouskos, Mahdieh Izadpanahkakhk, and Elham Omrani. Anticipation and next action forecasting in video: an end-to-end model with memory. In *arXiv preprint arXiv:1901.03728*, 2019. 2

[61] Will Price, Carl Vondrick, and Dima Damen. Unweavenet: Unweaving activity stories. In *CVPR*, 2022. 3

[62] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*, 2020. 3

[63] Santhosh Kumar Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Naq: Leveraging narrations as queries to supervise episodic memory. *CVPR*, 2023. 9, 10, 12

[64] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *NeurIPS Datasets and Benchmarks Track*, 2021. 2, 3, 7

[65] Santhosh K Ramakrishnan, Tushar Nagarajan, Ziad Al-Halah, and Kristen Grauman. Environment predictive coding for embodied agents. In *ICLR*, 2022. 3, 8, 9, 11

[66] Adria Recasens, Pauline Luc, Jean-Baptiste Alayrac, Luyu Wang, Florian Strub, Corentin Tallec, Mateusz Malinowski, Viorica Pătrăucean, Florent Altché, Michal Valko, et al. Broaden your views for self-supervised video learning. In *ICCV*, 2021. 3

[67] Nicholas Rhinehart and Kris M Kitani. Learning action maps of large environments via first-person vision. In *CVPR*, 2016. 1, 3

[68] Nicholas Rhinehart and Kris M Kitani. First-person activity forecasting with online inverse reinforcement learning. In *ICCV*, 2017. 1

[69] Cesar Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez. Procedural generation of videos to train deep action recognition networks. In *CVPR*, 2017. 3

[70] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 3

[71] Marco Rosano, Antonino Furnari, Luigi Gulino, and Giovanni Maria Farinella. On embodied visual navigation in real environments through habitat. In *ICPR*, 2021. 3

[72] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 3, 7

[73] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 11

[74] Jiayi Shao, Xiaohan Wang, Ruijie Quan, and Yi Yang. Action sensitivity learning for the ego4d episodic memory challenge 2023. *arXiv preprint arXiv:2306.09172*, 2023. 10, 2

[75] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. In *arXiv preprint arXiv:1804.09626*, 2018. 1

[76] Shuran Song, Andy Zeng, Angel X Chang, Manolis Savva, Silvio Savarese, and Thomas Funkhouser. Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. In *CVPR*, 2018. 3, 12

[77] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*, 2015. 3

[78] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 3

[79] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 3

[80] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012. 3

[81] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In *arXiv preprint arXiv:1807.03748*, 2018. 3

[82] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017. 3

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 7, 8

[84] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016. 3

[85] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2

[86] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017. 3

[87] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 3

[88] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *CVPR*, 2018. 3

[89] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019. 2

[90] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In *CVPR*, 2021. 1, 2

[91] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. Gibson dataset license agreement available at `http://svl.stanford.edu/gibson2/assets/GDS_agreement.pdf`. 3, 14

[92] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 10

[93] Ye Yuan and Kris Kitani. 3d ego-pose estimation via imitation learning. In *ECCV*, 2018. 3

[94] Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. Span-based localizing network for natural language video localization. In *ACL*, 2020. 8, 9, 11, 12

[95] Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. Learning 2d temporal adjacent networks for moment localization with natural language. In *AAAI*, 2020. 1, 2, 5, 9

[96] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization. In *ICCV*, 2021. 1, 2, 5

[97] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. In *TPAMI*, 2017. 2, 8, 9

Supplementary material

This section contains supplementary material to support the main paper text. The contents include:

A Videos illustrating our approach

We include our supplementary video on our project page https://vision.cs.utexas.edu/projects/ego-env/. The video illustrates the local state prediction task (Sec. 3.2), downstream video tasks (Sec. 3.3) and our main results.

In the first part of the video, we demonstrate the local state prediction task from Sec. 3.2 of the main paper. The video shows the first-person view of the camera-wearer (left panel). The right panel shows the top-down view of the environment with the agent trajectory (blue gradient) and nearby objects (colored squares). Note that models only see the egocentric view — the top-down map is for illustration only. Given a simulated video walkthrough and a query time-stamp, the model must predict the direction and rough distance of each object near it. Correct, missing and false positive predictions are shown for each direction (left panel). Correct predictions on the top-down map are highlighted in cyan.

In the second part of the video, we show examples of the two downstream video tasks from Sec. 3.3 of the main paper, on MP3D, HouseTours and Ego4D. In the ROOMPRED examples, the model must predict which room the camera-wearer is in from a short clip. As mentioned in Sec. 4.2, the clips show quick motions and often contain ambiguous views making it hard to predict room labels directly using traditional scene recognition methods. For example, the "staircase" is not visible as the person descends it (Ego4D, bottom right video). In the NLQ examples, the model must predict the moment in time that answers a particular environment-centric query. The video examples show how this requires reasoning about the camera-wearer's surroundings. For example, in the HouseTours clip ("when did I visit the sink in the bathroom?") the sink is only seen briefly in the video but the response requires the window of time that the camera-wearer was physically near it (within arms reach) regardless of visibility. In the Ego4D clip ("Did I leave the drawer open"), the model must know where the drawer is relative to the camera-wearer and link their actions to this physical location in order to respond.

Rank	Participant team	r@1, IoU=0.3 (↑)	r@1, IoU=0.5 (↑)	Mean r@1 (↑)	r@5, IoU=0.3 (↑)	r@5, IoU=0.5 (↑)	Last submission at	Meta Attributes
1	GroundNLQ	25.67	18.18	21.93	42.06	29.80	5 months ago	View
2	asl_nlq	24.13	15.46	19.79	34.37	23.18	5 months ago	View
3	ego-env	23.28	14.36	18.82	27.25	17.58	5 months ago	View
4	mzs (SnAG)	21.90	15.43	18.67	38.29	27.05	5 months ago	View
5	VioletPanda (finetuned+vgit+git)	22.03	14.11	18.07	26.50	17.81	5 months ago	View
6	Host_90322_Team (NaQ++ ReLER) B	21.70	13.64	17.67	25.12	16.33	7 months ago	View

Figure 1: Snapshot of the Ego4D NLQ leaderboard on 10/26/23. Leaderboard can be found on the challenge page. Note that both GroundNLQ [38] and asl_nlq [74] are concurrent works that were added to the leaderboard after the submission of this paper.

B Sim-to-real gap in Ego4D

As mentioned in Sec. 4.4 of the main paper, our model is affected by the type and diversity of pretraining data — videos of simulated agents walking around a house — limiting its generalization to unconstrained real-world video. Similarly, our approach is limited by simulator functionality — HM3D scenes support a small set of objects, which may not overlap with real-world environments, and Habitat does not support fine-grained object-interactions (e.g., chopping vegetables). As a result, we find that our approach works well on videos that are consistent with pretraining (i.e., indoor home scenarios; videos with lots of walking and less object interaction) but contributes less on out-of-distribution scenes and activities (e.g., golfing, outdoor cooking). The full list of scenarios is in Table 1.

Our results on the benchmark challenge (test set) in Table 2 corroborate this result. Note that our method was the top-ranked approach at the time of submission. Since then, other unpublished methods have been submitted to the leaderboard. We expect that future advancements in simulator capabilities (e.g., human motion models for agents, fine grained object interaction simulation) will help address this class of limitations. Fig. 1 shows a snapshot of the leaderboard as of 10/23/23.

Indoor	Visiting exhibition, On a screen (phone/laptop), Listening to music, Household management - caring for kids, Talking on the phone, Watching tv, Talking to colleagues, Electronics (hobbyist circuitry board kind, not electrical repair), Practicing a musical instrument, Eating, Cooking, Making coffee, Playing board games, Working at desk, Working out at home, Reading books, Playing games / video games, Hosting a party
Navigation	Roller skating, Bus, Walking on street, Indoor Navigation (walking), Walking the dog / pet, Car - commuting, road trip, Grocery shopping

Table 1: List of scenarios aligned with training environments.

C Data collection and annotation details.

We present data collection and annotation details for simulated walkthroughs, and our two downstream tasks (RoomPred and NLQ) for all three datasets (MP3D, HouseTours, Ego4D).

C.1 Walkthrough generation details

As mentioned in Sec. 4 (simulators) of the main paper, we generate simulated walkthroughs in HM3D [64] scenes to train our models. Given an environment, we first cluster all navigable points using KMeans, selecting between 4-64 clusters depending on the environment size. With each cluster

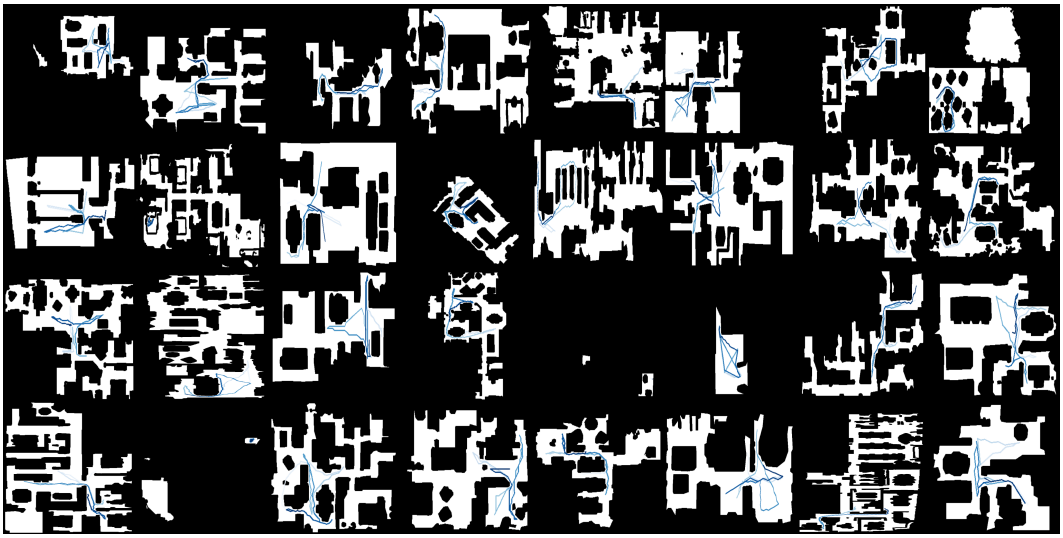


Figure 2: Walkthrough examples in HM3D. The blue gradient represents the trajectory from start (white) to end (blue). Black and white regions represent obstacles and free space respectively.

	SCENES	ROOMPRED	NLQ
HM3D [64]	800 / 100 / –	–	–
Matterport3D [6]	57 / 6 / 21	1337 / 173 / 536	8380 / 837 / 3452
HouseTours [7]	570 / 135 / 181	4512 / 1107 / 1593	2009 / 462 / 706
Ego4D [26]	998 / 328 / 333	3604 / 1596 / 1434	11291 / 3874 /4004

Table 2: Dataset train / val / test splits. Splits based on scenes and instances per downstream task are shown. HM3D is used only for pretraining (Sec. 3.2).

centroid as a starting location, we sample 8-16 nearest goal locations, shuffle them (to allow re-visitation), and make an agent visit the goals in sequence. We use a shortest-path planning agent that uses the underlying navigation graph to reach goals in the fewest number of steps. We collect a dataset of ~15k episodes, each of 512 steps, of our agents visiting such goal sequences for experiments in Sec. 3.2. Fig. 2 shows a random sample of walkthroughs.

Note that the walkthroughs are generated in environments where objects are not moved, however a large part of real-world environments are in fact static. This includes static scene elements like doors, windows, counter tops, staircases, and most objects that are typically not moved like refrigerators, beds, couches, TV sets. Encoding these objects and scene elements can thus still provide value for human-centric environment understanding, even when some objects may have moved around.

C.2 Data collection and annotation details

As mentioned in Sec. 4 (simulators) of the main paper, we collect labels for each video dataset. For Matterport3D, we directly use the ground truth information available through the simulator to extract labels. For HouseTours and Ego4D, we crowd-source annotations where required. We describe the data collection process and present data statistics for each dataset and task. The resulting dataset splits can be seen in Table 2.

C.2.1 Annotation requirements

ROOMPRED For this task, room labels are required at each time-step of the video. The 9 room categories used in Matterport3D are in Table 3 (left). These categories are pre-defined in the simulator. The 21 room categories used in HouseTours and Ego4D are in Table 3 (right). These categories were generated manually from a combination of Matterport3D room categories and a relevant subset of Places365 categories corresponding to indoor scenes.

NLQ For this task, natural language queries and corresponding moment boundaries (start and end times) are required. For HouseTours, we define 7 query templates where o refers to objects and r

Matterport3D

hallway	bathroom	bedroom
office	kitchen	living room
lounge	dining room	family room

HouseTours / Ego4D

attic	balcony	basement	bathroom	bedroom
closet	corridor/hallway	dining room	driveway	front door/entrance
garage/shed	gym	kitchen	lawn/yard/garden	living room
office/home office	porch	recreation room (billiards room/play room)		
staircase	storage/laundry/utility room	swimming pool		

Table 3: Room taxonomies for Matterport3D, HouseTours and Ego4D

Template	Example	Description
see o	Where did I first see the remote control?	Objects must be visible for the moment duration
see o in r	When did I see the mirror in the bathroom?	Object must be physically inside the room
see o_1 then o_2	Where did I see a table then a chair?	Objects can either be seen together or in quick succession
visit r_1 then r_2	When did I walk from the living room to the kitchen?	Start = when the person begins to leave; End = they are fully inside the kitchen.
visit o/r	When did I last visit the couch?; When did I last visit the bedroom?	Visit = physically near an object (within arms reach) or physically inside a room
visit o_1 then o_2	When did I visit the lamp then the couch?	Same as see o_1 then o_2, but using the "visit" criteria above
visit o in r	When did I visit the mirror in the bathroom?	Same as see o_1 in o_2, but using the "visit" criteria above

Table 4: Query templates and examples for the NLQ task on MP3D and HouseTours.

refers to rooms: "see o", "see o in r", "see o_1 then o_2", "visit r_1 then r_2", "visit o/r", "visit o_1 then o_2", "visit o in r". Each template captures a type of question that requires a different mechanism of reasoning. "see" queries require reasoning about what is immediately visible; "visit" queries require an understanding of where the camera-wearer is in the environment and what objects are nearby (within arms reach); "see/visit o_1 then o_2" and "see/visit o in r" require both spatial and temporal reasoning. Natural language queries follow from these templates. For example, for the "visit r_1 then r_2" template, a natural language query may be "When did I first walk from the kitchen to the bathroom?". The list of query templates, examples and descriptions can be seen in Table 4. The task definition follows prior work [26] but is adapted for the datasets used, and contains more environment-centric queries. For Ego4D, we use the existing NLQ benchmark task annotations.

Our video in Sec. A shows examples of both tasks to complement the image examples from Fig. 4 of the main paper. The video highlights the stark contrast between prediction in static images (third-person photos) which contains well-framed images that are easy to recognize, and egocentric video which is much more challenging. In this setting, video is tied to quick ego-motion as the camera-wearer moves around the environment and objects are seen only briefly (or not at all) in non-canonical viewpoints.

C.2.2 Matterport3D annotations

For ROOMPRED, navigable location are mapped to room categories using information from the simulator. We map agent positions for each video frame to these categories. For NLQ we use room labels and extracted object positions to generate queries from the 7 templates above. We define objects as "seen" if they occupy at least 5% of the pixels in a given frame. We define objects as "visited" if the agent is < 1.0m from the object, regardless of its visibility, following embodied navigation protocol (ObjectNav [2]). Rooms are "visited" using the position $\rightarrow$ room category mapping above. We generate queries for each template by tracking objects and rooms that are seen and visited over time. If there are multiple visitations to an object or room, we ensure unique responses to queries by adapting them to consider only the first (or last) visit.

Fig. 3 shows annotation statistics for both tasks on Matterport3D. Trajectories are fixed length (512 steps) with 7 room transitions on average. Both visits and moments are short, making it difficult to localize the response to the natural language query, and providing little extra context to recognize rooms. See our video in Sec. A for examples.

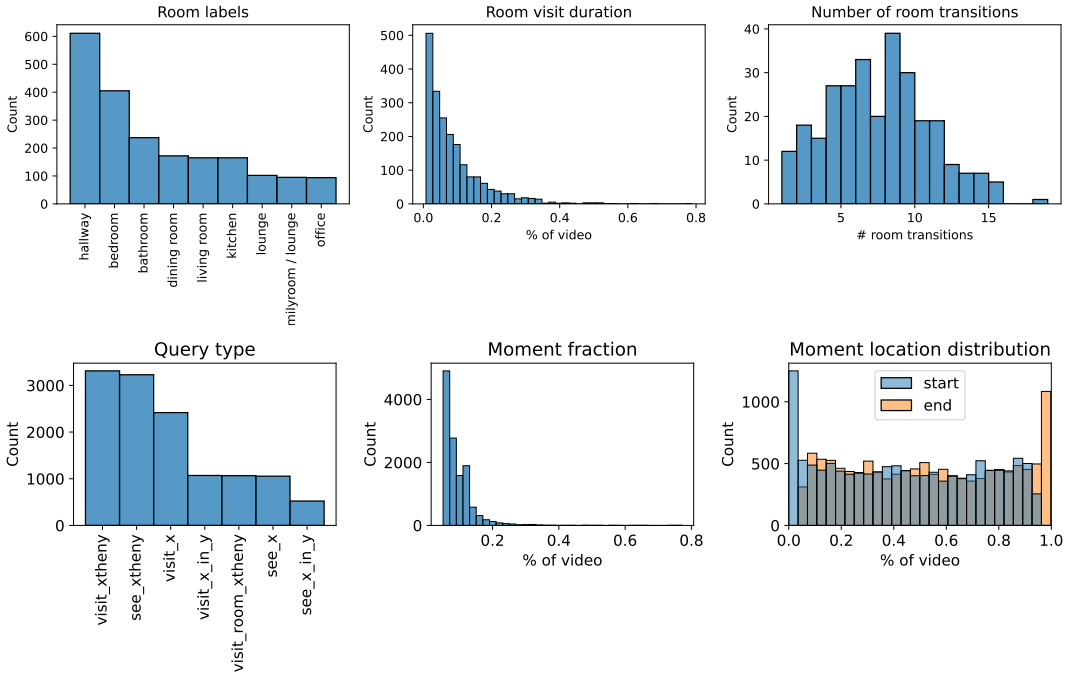


Figure 3: Annotation statistics for Matterport3D. Top panel: ROOMPRED data distribution. (left) distribution of room categories; (center) length of each room visit relative to the full video; (right) number of room transitions in each video. **Bottom panel:** EPISODIC MEMORY RETRIEVAL data distribution. (left) distribution of query types; (center) length of each annotated moment relative to the full video; (right) distribution of start and end times of annotated moments.

C.2.3 HouseTours annotations

We crowd-source annotations for real-world videos from HouseTours. For ROOMPRED, we ask annotators to watch a video and mark the start and end time of each "visit" to a room. They must then label each visit with one of the 21 room categories (from Table 3, right). An illustration of the annotation interface is shown in Fig. 6.

For NLQ, we ask annotators to identify an interesting moment (e.g., where a person sees a salient object, moves from one room to another, visits an important object) which serves as the answer to a query. The moment is specified by a start and end time, while the query is specified as natural language text generated following one of the 7 template classes from Table 4. An illustration of the annotation interface is shown in Fig. 7.

Fig. 4 shows annotation statistics for both tasks on HouseTours. In general, trajectories are relatively shorter than Matterport3D, though they involve a similar number of room transitions (6 on average). They share similar challenges with short moments. See our video in Sec. A for examples.

C.2.4 Ego4D annotations

Following the same procedure as HouseTours, we crowd-source room visit labels on Ego4D videos (see Fig. 5). Ego4D videos are much longer (30 mins on average). As a consequence, visits are naturally a much smaller fraction of the overall video and the number of room transitions are much higher (16 on average). The distribution of room categories is similar to HouseTours. Annotation statistics for NLQ can be found in Section F.4 of the supplementary material of the Ego4D paper [26].

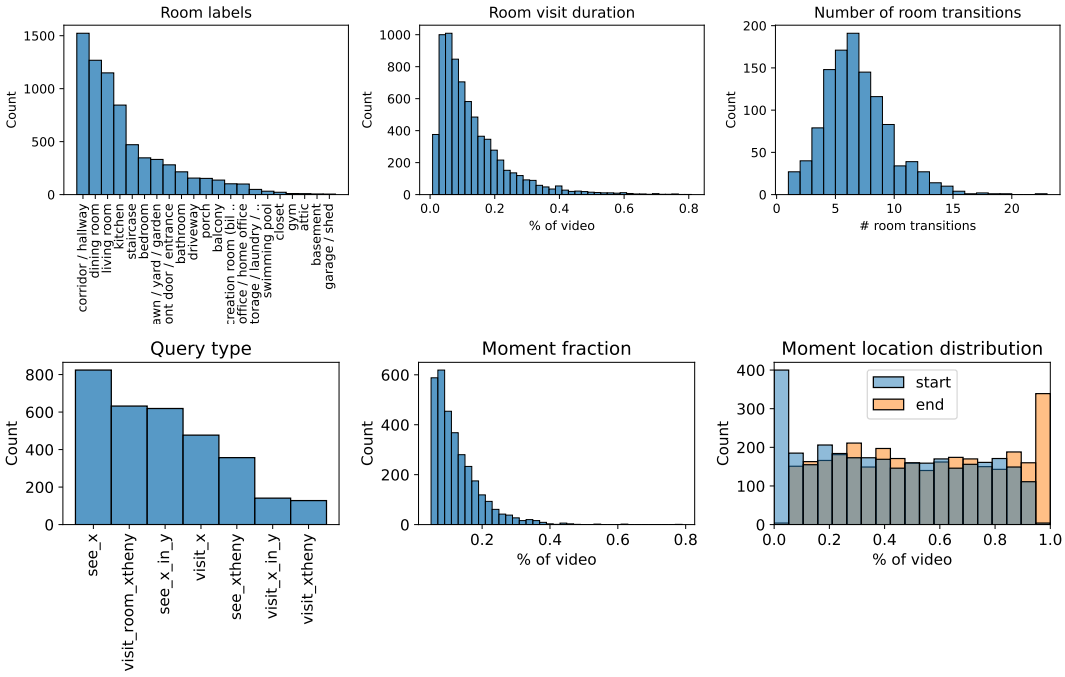


Figure 4: Annotation statistics for HouseTours. Top panel: ROOMPRED data distribution. (left) distribution of room categories; (center) length of each room visit relative to the full video; (right) number of room transitions in each video. **Bottom panel:** EPISODIC MEMORY RETRIEVAL data distribution. (left) distribution of query types; (center) length of each annotated moment relative to the full video; (right) distribution of start and end times of annotated moments.

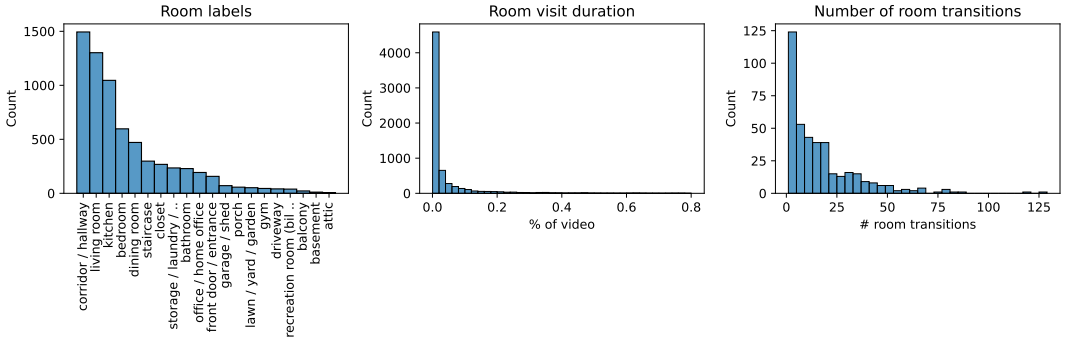


Figure 5: Annotation statistics for Ego4D. ROOMPRED data distribution. (left) distribution of room categories; (center) length of each room visit relative to the full video; (right) number of room transitions in each video. We do not collect annotations for NLQ on Ego4D. We use annotations from the official Ego4D benchmark challenge (Section F.4. in the Ego4D paper [26]).

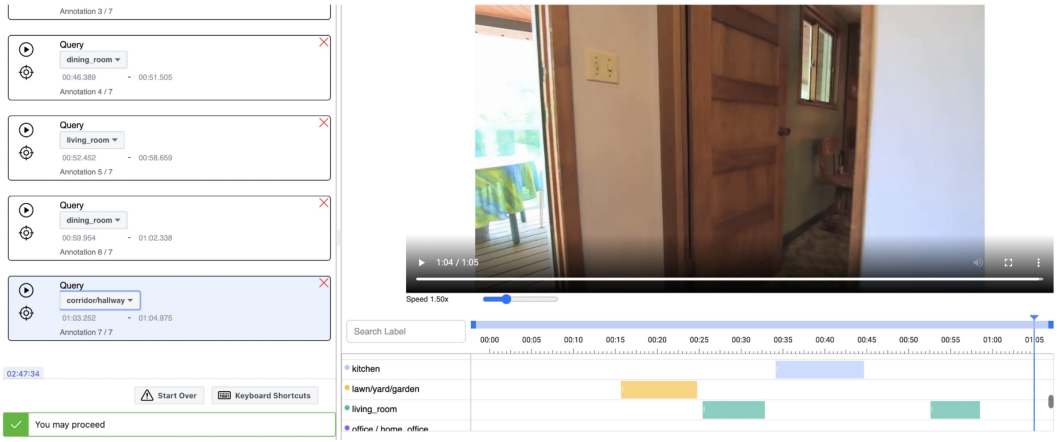


Figure 6: Annotation interface for collecting ROOMPRED labels. Annotators must densely segment room visits (start and end times) and associate a class label to each of them.

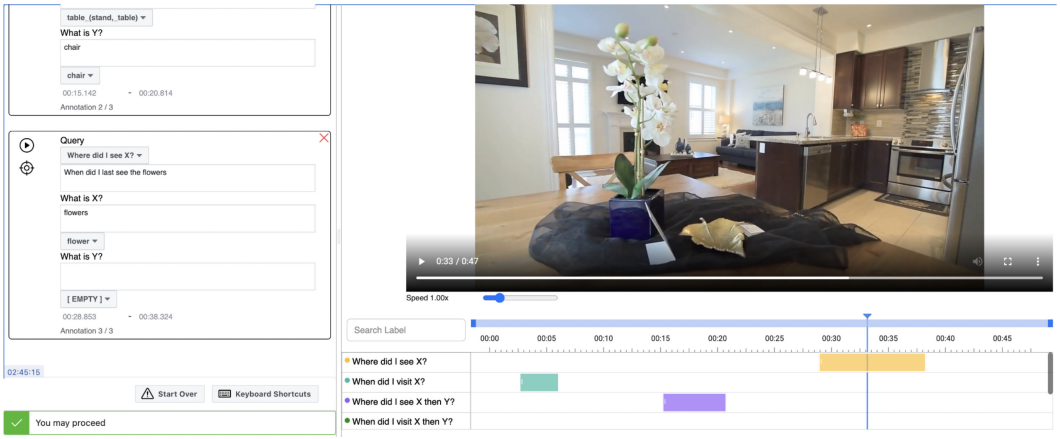


Figure 7: Annotation interface for collecting NLQ labels. Annotators must identify an interesting moment in time (start and end time) and associate a query template, a natural language query and object/room labels that fill the query template slots.

chair	table	picture	cabinet
cushion	sofa	bed	chest_of_drawers
plant	sink	toilet	stool
towel	tv_monitor	shower	bathtub
counter	fireplace	shelving	seating
furniture	appliances	clothes	

Table 5: HM3D object taxonomy

D Implementation and training details

We present architecture and training details for all approaches.

D.1 Architecture and training details (pretraining)

We present additional details for our model in Sec. 3 of the main paper, as well as the baselines in Sec. 4 (baselines). The full list of hyperparameters are in Table 6.

Pose embedder $\mathcal{P}$, environment encoder $\mathcal{E}$ and decoder $\mathcal{D}$. We build on the transformer [83] architecture for our model. The inputs $f \in \mathbb{R}^{2048}$ are features from an ImageNet-pretrained ResNet-50 . These are encoded into 128-dimension vectors using a visual encoder (2-layer MLP).

First, 128-D pose embeddings are generated following Eqn. 3. These pose embeddings are concatenated with the visual embedding, and then transformed back to a 128-dimension vector using $\mathcal{M}_p$. Sin-cosine position embeddings are added to this input following [83] resulting in the transformer input $\{x_1, ..., x_N\}$ in Eqn. 4. Note that a separate visual encoder generates input features for the encoder $\mathcal{E}$ and the pose embedder $\mathcal{P}$.

The encoder $\mathcal{E}$ performs multi-headed self-attention using these inputs, with 2 layers, 8 heads and hidden dimension 128. The decoder $\mathcal{D}$ is a 2-layer transformer with hidden dimension 128 which attends to the outputs of $\mathcal{E}$ to generate the output representation. We predict relative directions for 23 object classes in HM3D (see Table 5). We use the transformer implementation from PyTorch [59].

Our architecture is similar to prior embodied navigation approaches [65, 20] but does not require pose (and instead uses pose embeddings), includes the environment decoder that queries the memory based on pose embeddings and visual content, and is trained using our proposed learning objective in Sec. 3.2. See Table 6 (left).

EPC architecture details. EPC [65] is a transformer encoder-decoder model that masks out frames from physical locations and predicts the features of these masked zones given a query pose. We generate graphs to train this baseline following their approach.

Specifically, for every video frame $\{f_1, ..., f_T\}$, we compute the geometric viewpoint overlap with every other frame. The viewpoint overlap $\psi(f_i, f_j)$ is calculated by projecting pixels from the frames to 3D point-clouds using camera intrinsics, agent pose and depth measurements, and measuring the percentage of shared points across frames. We use $\psi(f_i, f_j)$ as our distance metric to cluster all frames in video into zones using hierarchical agglomerative clustering. We set the distance threshold to 0.8 as larger values result in too few zones.

We sample 4 unseen zones per instance in a batch of which one is "positive" and three are "negatives" for contrastive learning. We collect negatives from all instances in a batch during training. The network is trained using noise-contrastive estimation following [65]. See Table 6 (center).

EgoTopo architecture details. EgoTopo [57] is an approach to translate egocentric video frames into a topological graph, where each node contains a list of clips that correspond to a physical location, and edges correspond to rough spatial layout.

For MP3D and HouseTours, ee directly use available pose information to determine whether two frames belong to the same zone or not (i.e., we do not train a retrieval network to approximate this). This amounts to a clustering of the trajectory based on pose (position and heading) and represents an enhanced version of EgoTopo that benefits from pose data. We use affinity propagation to cluster frames into nodes. To compute edges, we calculate the distance between the centroid of each node and

Transformer architecture	
Input dim	2048
Hidden size	128
Pose emb dim	128
Walkthrough length	512
Memory size (K)	64
# attention heads	8
# encoder layers	2
# decoder layers	2

EPC parameters	
# Unseen zones	4
Clustering threshold	0.8
EgoTopo parameters	
Affinity prop damping	0.5-0.9
GCN hidden dim	128
# GCN layers	2

Optimization parameters	
Max epochs	2500
Learning rate	1e-4
Weight decay	2e-5
Batch size	512

Table 6: Architecture and training hyperparameters for pretraining (Sec. 3.2)

assign an edge if this distance is < 3.0m to be consistent with Eqn. 2. For Ego4D, pose information is not available. We fall back to clustering visual frames based on ImageNet pre-trained ResNet-50 frame features.

Node features are calculated as the average of features assigned to that node. We use a 2-layer graph convolutional neural network (GCN) to aggregate features across nodes, and then average them to form a single video encoding following [57]. See Table 6 (center).

MAE architecture details. We use all frames from our generated walkthroughs to train an MAE VIT-large model with 16×16 patch size. We use the authors existing code and train a model for 200 epochs.

ObjFeat architecture details. We train a QueryInst instance segmentation model [21] with a R-50-FPN backbone, using a dataset sampled from the 100 HM3D scenes with ground-truth semantic object annotations. We use the implementation in the MMDetection package.

Training details. As mentioned in Sec. 4 (experiment setup) of the main paper, we train our models for 2500 epochs using the Adam optimizer with learning rate $1e-4$. We sample $K = 64$ frames to construct our memory. At training, we sample frames from the video but randomly offset frame indices to train robust models. During inference, we uniformly sample frames. We select the model with the lowest validation loss to evaluate downstream. See Table 6 (right) for all optimization hyperparameters.

D.2 Architecture and training details (downstream)

We present additional details for the approaches in Sec. 3.3 of the main paper. The full list of hyperparameters are in Table 7.

Room prediction models. As mentioned in Sec. 4.2 of the main paper, for our PLACESCNN baseline model, we build a classifier on top of features from the wide ResNet-18 model from [97]. We use the authors existing code and their provided pretrained models to initialize the model. The classifier head is a 2-layer MLP with hidden dimension 512. The backbone is frozen and only the classifier is fine-tuned. $N = 8$ frames around the target frame are used to provide additional context. The features are max-pooled before classification.

For our models, we use the target frame as the query to produce a single environment-feature, which is then concatenated with all $N = 8$ frames and aggregated following Eqn. 8. This new, enhanced input is fed into the baseline model as described above.

We train all models for 80 epochs with learning rate $1e-4$ using the Adam optimizer. The full list of hyperparameters are in Table 7 (left)

Episodic memory retrieval models. As mentioned in Sec. 4.3 of the main paper, we build on the VSLNet model from [94]. We use an existing implementation based on the authors original code. Visual inputs are encoded as $N = 128$ clip features, created by adaptive average pooling of SlowFast-R50 clip features. Note that for MP3D we use ResNet-50 features as the walkthroughs contain discrete agent steps, rather than smooth video frames.

RoomPred	
# input frames	8
Hidden size	512
# layers	2
Optimization parameters	
Max epochs	80
Learning rate	1e-4
Weight decay	2e-5
Batch size	512

NLQ	
# Input clips	128
Hidden size	128
Highlight lambda	5.0
Extend boundary %	0.1
# heads	8
# layers	4
Dropout rate	0.2
Optimization parameters	
Max epochs	200
Learning rate	1e-4
Weight decay	1e-2
Batch size	64

Table 7: Architecture and training hyperparameters for RoomPred and NLQ (Sec. 3.3)

For our models, we select the center frame of each of the $N = 128$ inputs and use them as query frames to produce $N = 128$ environment features. Each pair of input feature and environment feature is aggregated following Eqn. 8, and then input to the VSLNet model described above. Note that we perform this aggregation *after* the video affine and feature encoding layers in VSLNet.

We train all models for 200 epochs with a learning rate of $1e-4$ using the Adam optimizer. The full list of hyperparameters are in Table 7 (right).

For Ego4D experiments, we use the hyperparameters described in the respective benchmark whitepapers [26, 49, 53] and only aggregate precomputed environment features as described above.

E Supplementary experiments and analysis.

E.1 Experiments with extra pretraining data (Ego4D videos vs. simulator walkthroughs)

In Sec. 4 (baselines), all approaches have access to the same set of simulator-generated walkthrough videos for pre-training for apples-to-apples comparisons. In this section, we test the effect of directly performing self-supervised pretraining on Ego4D videos instead. In Table 8 we compare our MAE baseline trained on walkthrough videos with the same architecture trained on Ego4D frames [92] using the author provided pretrained model.

Rank1@m →	@0.3	@0.5	avg
MAE (Walkthrough)	5.65	3.02	4.34
MAE (Ego4D)	5.65	3.27	4.47
EgoEnv	**6.04**	**3.51**	**4.77**

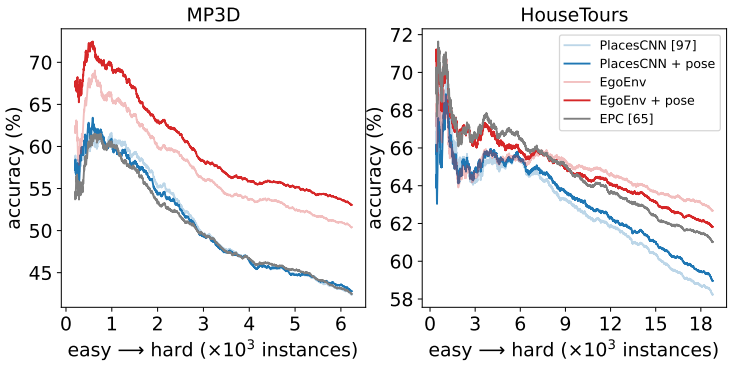
Table 8: NLQ results for Ego4D with MAE trained on Ego4D.

E.2 Additional experiments with pose embeddings and ground-truth pose

As noted in Sec. 3.2.1 ground truth pose may be utilized directly by our model, however it is not easily available in real-world egocentric video, which makes our use of inferred pose embeddings a strength.

E.2.1 Importance of pose embeddings

As noted in Sec. 3.2.1 of the main paper, we train pose embeddings to help relate observations by their visual content as well as relative orientation of capture. We measure the effect of pose embeddings on our local state prediction task. Our models see improvements in predicting objects in each of the cardinal directions (26.1 vs. 24.9 mAP) as well as improved object distance prediction (59% vs 58%). This improvement is small for predicting objects in the forward view (+0.7 mAP) where scene



	MP3D [6]			HouseTours [7]		
RANK1@M →	@0.3	@0.5	AVG	@0.3	@0.5	AVG
VSLNET [94]	33.69	22.83	28.26	42.94	27.68	35.31
VSLNET + POSE	34.68	23.70	29.19	43.22	25.56	34.39
EPC [65]	36.85	**27.64**	32.24	43.22	27.82	35.52
EGOENV	38.18	26.85	32.52	**51.98**	**34.18**	**43.08**
EGOENV + POSE	**38.51**	27.30	**32.91**	46.47	30.08	38.28

Figure 8: Task performance using ground-truth pose. ROOMPRED (top) and NLQ (bottom). Our approach (w/ and w/o pose) outperforms baselines. Noisy pose in HouseTours degrades performance. Ego4D videos do not have associated pose.

information is directly visible, but large for other views that need to be inferred: mAP improvements of +2.2 (right), +0.9 (behind) and +1.0 (left).

E.2.2 Importance of ground truth pose

In this section, we explore using pose estimates obtained directly from the simulator or using off-the-shelf structure for motion methods. Note that the existing method EPC in Fig. 6 and Table 1 already uses this ground-truth pose information, which other models (including ours) do not have access to.

Extracting pose information. For HouseTours videos, we run COLMAP [73], a structure from motion framework to compute camera-pose information associated with each frame of the video. For this, we first extract frames from each video at 2fps. Then we run COLMAP using a precomputed vocabulary tree file (flickr100K_words32K). The resulting trajectories are inherently noisy due to the approximate nature of the SfM pipeline and the absence of true camera parameters for such in-the-wild video. We post-process them by removing erroneous pose values (ones that cause jumps in pose atypical to smooth motion).

Overall, COLMAP successfully localizes ~32 hours of video from 886 houses out of the original 119 hours available in [7]. Some examples of video trajectories are shown in Fig. 9. Comparing these to the simulated trajectories in Fig. 2, we see smoother trajectories overall, but with unrealistic jumps in localizations and loop-closure failures. Note that we visualize only the trajectory, not obstacles in the environment, as we do not have access to occupancy maps for the real-world videos.

Note that these videos are tours of indoor spaces with the intention of visually covering a large area. As a result, the camera-wearer moves slowly and smoothly to show parts of the house. Despite this, only a fraction of trajectories can be localized highlighting the difficulty of estimating pose from monocular video. In contrast, the same procedure fails to localize the camera-wearer in Ego4D videos due to rapid head motions and motion blur. On MP3D, pose is available directly from the simulator.

Performance with ground-truth pose. In Fig. 8, we investigate the role of pose information for ROOMPRED (top) and NLQ (bottom), by directly embedding ground-truth pose as part of the input to the baseline and our model. We find that our model can benefit from pose on MP3D, but falls short on HouseTours, due to noise in extracted pose (compared to simulator-provided pose in MP3D). However, our approach (with and without pose) outperforms EPC, which explicitly leverages pose both at train and test time.

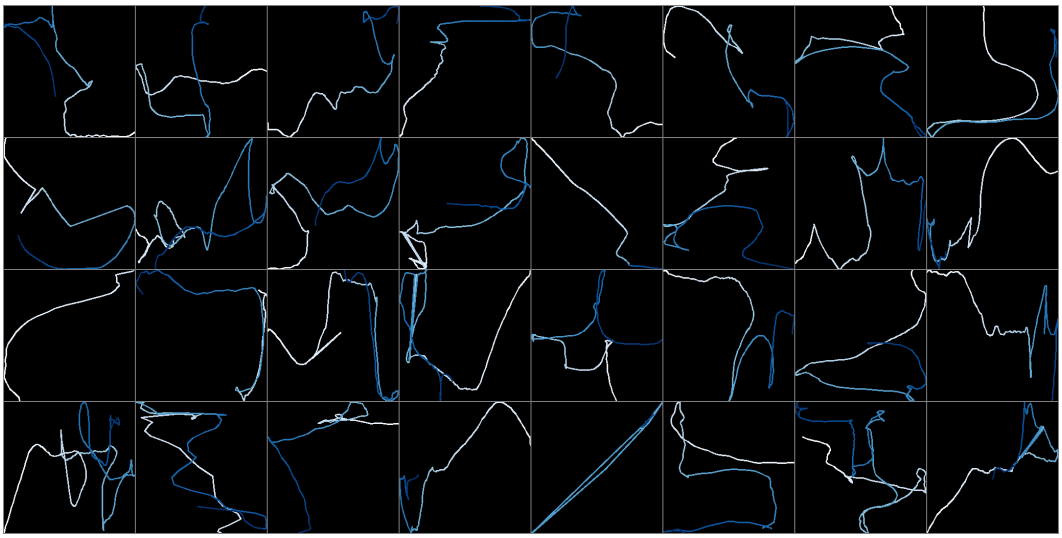


Figure 9: Camera-pose for HouseTours from COLMAP. The blue gradient represents the trajectory from start (white) to end (blue).

	Ego4D [26]		
RANK1@M →	@0.3	@0.5	AVG
VSLNET [94]	5.45	3.12	4.29
+ EGOENV	**6.04**	**3.51**	**4.77**
EGOVLP [49]	**10.53**	5.96	8.25
+ EGOENV	10.51	**6.71**	**8.61**
RELER [53]	10.79	**6.74**	8.77
+ EGOENV	**11.10**	6.56	**8.83**
RELER*	13.68	8.23	10.96
+ EGOENV	**14.40**	**8.54**	**11.47**
NAQ [63]	24.12	15.04	19.58
+ EGOENV	**25.37**	**15.33**	**20.35**

Table 9: EgoEnv features with alternate models. Results on the Ego4D NLQ validation set. RELER* combines EgoVLP [49] features with the model from [53].

E.3 EgoEnv integrated into other baseline approaches.

In Table 1 in the main paper, we report results on Ego4D using a single architecture and feature combination (VSLNet [94] with SlowFast [22] features). In Table 9 we show results with EgoEnv features integrated into other architectures. Our features consistently improve performance across all architectures, highlighting the complementary environment-level information encoded through our approach.

E.4 Alternate pretraining task formulations.

In Sec. 3.2.2 in the main paper, we introduced our local state prediction task that is used to pretrain our video encoders on simulated walkthrough videos. We investigate alternate pretraining objectives to validate our task choice. We compare against the following:

- **CARDINALOBJ** is a variant of our local state tasks where we predict only the object categories in each cardinal direction, but not the distances.
- **POSEEMBED** predicts the relative pose (discretized position and orientation) between every pair of observations in a walkthrough video. This is a sub-component of our full model (Sec. 3.2.1).
- **PANOFEAT** directly predicts the image features in each cardinal direction, inspired by prior work on panorama completion [41, 76, 44].

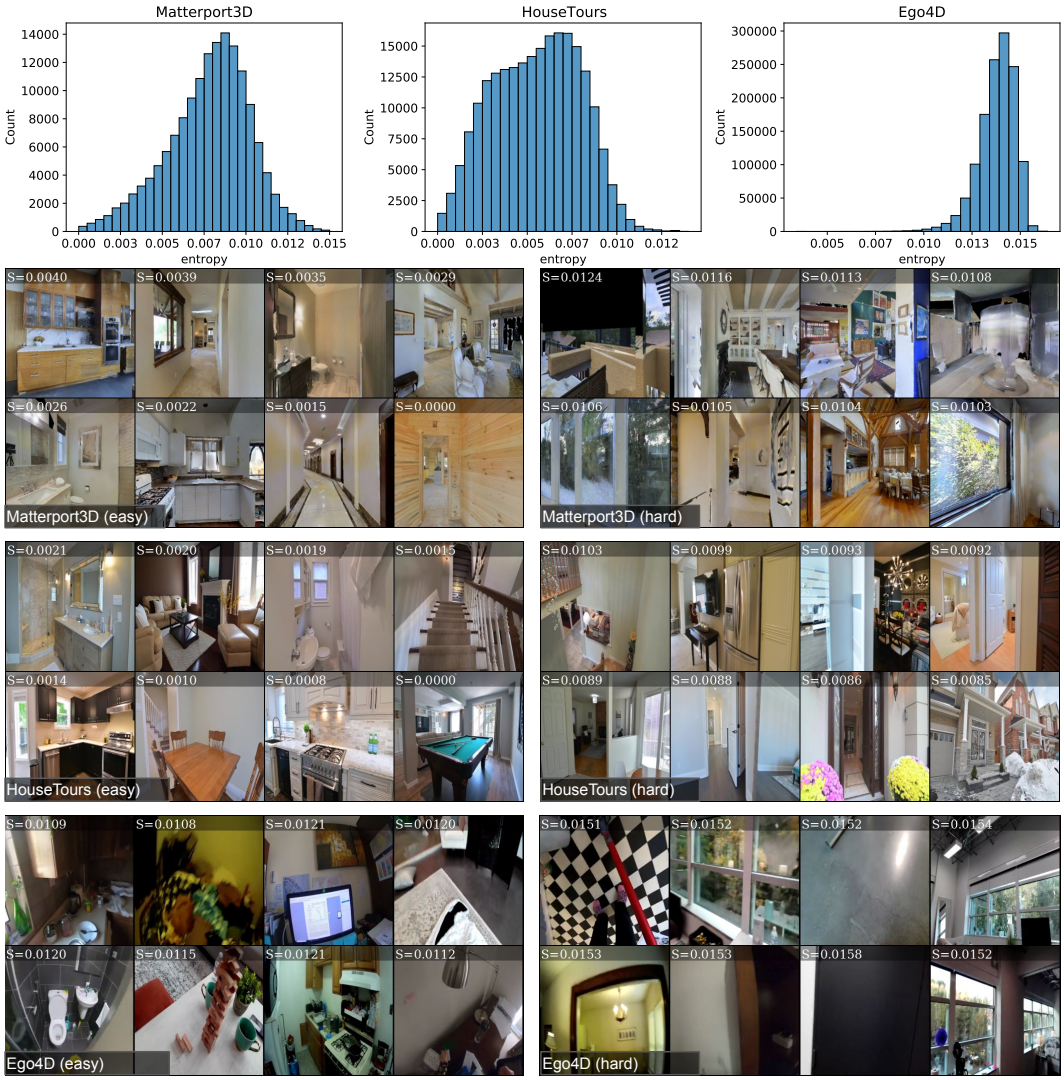


Figure 10: Illustration of easy vs. hard instances for all datasets. Top panel: Distribution of entropy scores for ROOMPRED instances. Ego4D instances are skewed towards hard instances due to the egocentric viewpoint and rapid camera and scene motion. **Bottom panel:** Following Sec. 4.2, we sort instances as by their entropy score S. We show samples from the top and bottom 10% instances.

- **PANOCONTRAST** uses noise contrastive estimation (NCE) to train a model to predict image features in each cardinal direction. For positives, we use the true image feature in the corresponding direction direction. For negatives, we use image features from the other 3 cardinal directions, as well as trajectory images from different scenes.

Each of these objectives explicitly encodes a combination of semantic and geometric information. For example, PANOFEAT and PANOCONTRAST encodes primarily semantic information as they require reconstruction of image features. POSEEMBED encodes primarily geometric information to predict the relative pose between observation pairs. CARDINALOBJ encodes semantics from object categories and weak geometric information from their relative orientations. Table 10 highlights the performance of these variants on both downstream tasks. Our approach that requires predicting both object labels, orientations as well as rough distances offers a balance of both cues during pretraining, translating to strong downstream performance.

	RoomPred	
	MP3D	HT
PoseEmbed	38.73	59.72
CardinalObj	49.04	61.32
PanoFeat	48.19	62.78
PanoContrast	47.28	**62.97**
Ours	**50.40**	62.68

	NLQ	
	MP3D	HT
PoseEmbed	28.46	38.56
CardinalObj	29.32	39.34
PanoFeat	31.34	38.06
PanoContrast	**33.22**	38.56
Ours	32.51	**43.08**

Table 10: Alternate pretraining tasks. Our local state prediction task offers a good balance of semantic and geometric cues that lead to features with strong downstream performance. For RoomPred (left) we report accuracy (%). For NLQ (right) we report mean Rank1@(0.3, 0.5).

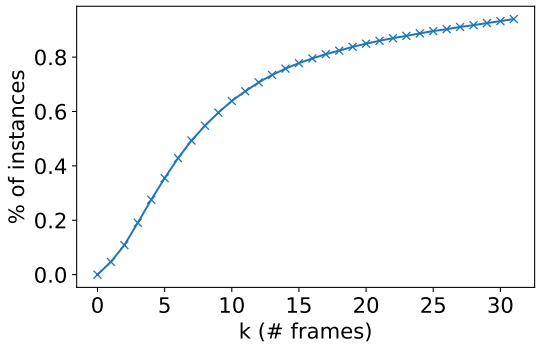


Figure 11: Percentage of training instances that involve "rare" objects. The x-axis sets a threshold for what is considered rare. For example, 5% of training instances involve anticipating completely unseen object instances ($k < 1$).

E.5 Memory vs. anticipation during pretraining

As mentioned in Sec. 3.2 of the main paper, our pretraining task involves elements of both aggregating information about relevant views spread across the walkthrough, as well as anticipating objects that are rarely (or never seen). We quantify this statement in Fig. 11 where we show the percentage of training instances where objects are rarely seen, for different definitions of rarity. For example, 23% of training instances involve predicting objects that appear in only $k < 4$ frames. 5% of training instances involve anticipating completely unseen object instances ($k < 1$).

E.6 Task-specific pre-training in simulation

As mentioned in Sec. 3.3 of the main paper, our goal is to train task-agnostic representations using videos from simulated agents. The end result is a single model that can generate features for multiple tasks (in our experiments, RoomPred and NLQ). This is different from traditional sim-to-real approaches where a new dataset needs to be collected for every downstream task, and a separate model has to be trained on it. Such a dataset needs to be well balanced and carefully designed to match the downstream task. Moreover, as tasks are added, new datasets per task need to be created which may be impractical, especially when they require data beyond the simulator's capability (e.g., simulating human motion, hand-object interaction).

To investigate further, for the RoomPred task, we generate a dataset in simulation that maps trajectory frames to room labels[2]. The room categories are estimated directly from the simulator by matching each frame to the nearest navigable point in an annotated room region. We train a ResNet18 model to predict the room category (a six-way classification) and then use features from this model following baselines in Sec. 4 (baselines). The new baseline benefits from representations learned for the *same task* — room prediction — and on the same volume of simulated training data.

On MP3D this performs better than the PlacesCNN baseline (43.3 vs. 42.4%) but is weaker than our model (50.4%). On HouseTours, it performs worse than PlacesCNN (57.9 vs. 58.2%) and our approach (62.7%). The low performance may be attributed to the small label space (only

[2]We use trajectories from Gibson [91] scenes as HM3D region annotations are not provided.

six categories), resulting in features that are not discriminative for the large-scale, diverse data downstream. More generally, the task-specific approach is more susceptible to failures due to the sim-to-real gap, which manifests as a lower performance on real-world video frames from HouseTours.

F Ablations experiments and additional visualizations.

We present ablation experiments for various model design choices and additional experiments to supplement the discussion in Sec. 4.5 in the main paper.

F.1 ROOMPRED results with error bars

In Fig. 6 of the main paper, we report results over three runs, by aggregating predictions across all three runs, and then sorting them by difficulty (Sec. 4.2). In Table 11, we show results with standard error *averaged over the three runs* to highlight the variance across approaches. Environment-centric approaches (EPC, TRF, EGOENV) perform better than other baselines, despite higher variance in accuracy. Our approach is consistently the best amongst these.

	MP3D	HouseTours	Ego4D
PLACESCNN	42.39 ± 0.15	58.24 ± 0.02	49.50 ± 0.20
FRAMEFEAT	42.04 ± 0.08	58.70 ± 0.10	49.34 ± 0.12
OBJFEAT	43.72 ± 0.06	59.02 ± 0.12	48.74 ± 0.11
MAE	42.79 ± 0.25	58.30 ± 0.08	48.87 ± 0.08
EGOTOPO	41.19 ± 0.57	58.05 ± 0.15	49.42 ± 0.05
EPC	42.48 ± 1.12	61.02 ± 0.20	–
TRF (SCRATCH)	43.27 ± 0.40	62.12 ± 0.14	49.65 ± 0.64
EGOENV	**50.40 ± 1.29**	**62.68 ± 0.19**	**51.07 ± 0.65**

Table 11: ROOMPRED results with error bars across three training runs.

F.2 Ablation studies.

We perform ablation experiments for several model design choices listed in Sec. 4 (experiment setup) of the main paper. All ablation experiments are performed on on validation data splits of MP3D and HouseTours.

Window size W. The window size controls the density of sampled frames for building our environment memory. Larger windows imply temporally separated frame inputs. Our results are in Fig. 12 (left column). We find that $W = 64$ is sufficient for localizing the room category for ROOMPRED, while a $W = 256$ is best for NLQ which requires reasoning over longer horizons.

Memory size K. The memory size controls the number of frames sampled from a window of size W for building our environment memory. Our results in Fig. 12 (middle column) show the sensitivity of our model to this parameter. On both tasks and datasets, we see only marginal improvements with higher memory sizes, though $K = 64$ results in the best performance.

Loss weight term λ. λ controls the weighting between the object and distance prediction term in the local state prediction loss in Sec. 3.2.2. Our results in Fig. 12 (right column) show that $\lambda = 0.1$ results in the best balance between the two semantic and geometric terms.

F.3 Additional attention visualizations

We present additional examples visualizing the learned attention values in our transformer decoder model in Fig. 13 to supplement Fig. 5 of the main paper. Our model learns to attend to diverse views that are not simply based on temporal adjacency or visual overlap — they capture the surroundings of the camera-wearer.

F.4 Easy vs. Hard instances in the ROOMPRED task

As mentioned in Sec. 4.2 of the main paper, we sort instances for evaluation by difficulty based on the prediction entropy of a pre-trained scene classifier model. In Fig. 10 (top), we show the distribution

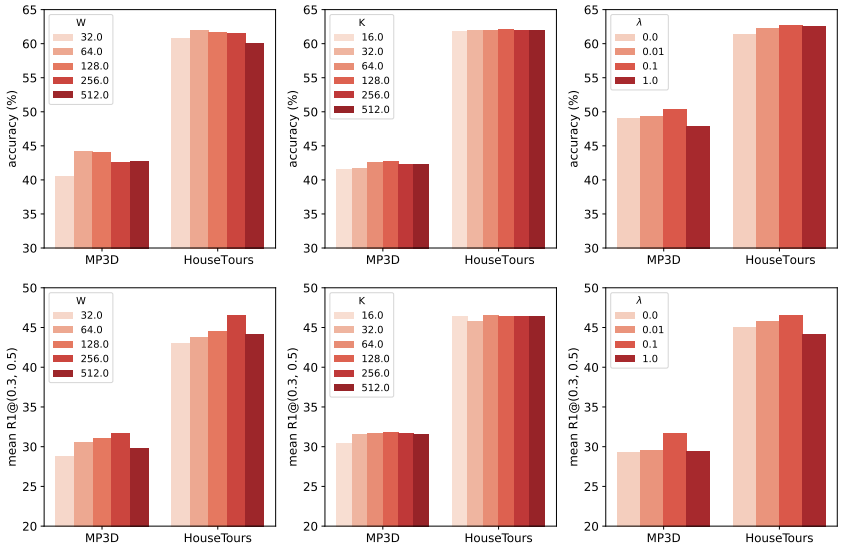


Figure 12: Ablation experiments on ROOMPRED (top) and NLQ (bottom). We ablate model hyperparameters: window size W (left), memory size K (middle) and loss weight term λ (right). See text for analysis.

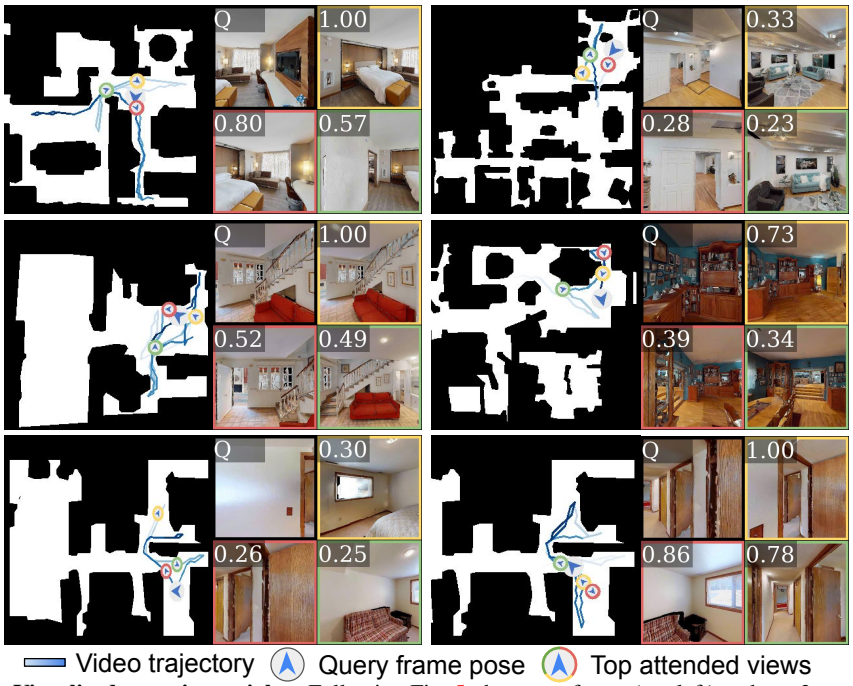


Figure 13: Visualized attention weights. Following Fig. 5, the query frame (top left) and top-3 attended views (colored boxes), their positions along the trajectory (colored circles), and their associated attention scores are shown.

of this entropy score across all three datasets. In general, Ego4D contains the hardest instances due to characteristic egocentric motion patterns, while HouseTours contains easier examples where the camera-wearer tends to dwell in one particular location to showcase it. We show examples of easy vs. hard instances in Fig. 10 (bottom). Note that the figure only shows the center frame of the clip that is used to predict the room label to highlight the difference between easy and hard frames. Our results in Fig. 6 highlight the advantage of our approach on these hard instances where environment-level reasoning is essential. See our video in Sec. A for more context.