A. More Implementation Details

Local Query Refinement. After obtaining the initial queries, we refine them through a series of decoder layers inspired by PQ3D. Specifically, within each decoder layer l, our goal is to better retrieve object-relevant information by enhancing the interaction between the object queries Q_t^l and the input features $\{\hat{F}_t^{2D}, \hat{F}_t^{3D}\}$. To achieve this, we first employ a cross-attention mechanism, allowing the object queries Q_t^l to attend to the input features $\{\hat{F}_t^{2D}, \hat{F}_t^{3D}\}$. This step is crucial for aggregating relevant information from both 2D and 3D features.

To further improve the efficiency and performance of the cross-attention, we adopt a masked attention mechanism. This restricts the attention scope to localized features centered around each query, ensuring that the queries focus only on relevant regions rather than the entire feature map. This approach not only enhances the model's ability to capture fine-grained details but also significantly reduces computational overhead by limiting the attention span.

Following the cross-attention, we introduce a spatial self-attention module. This module leverages the coordinates of the queries to explore their spatial relationships, enabling the model to better understand the positional context of each query. By incorporating spatial information, the model can more effectively distinguish between different objects and their locations within the scene.

Each attention layer is followed by a forward feed network (FFN) and a normalization layer to stabilize the training process and enhance the representation learning capabilities of the model. The entire process within a single decoder layer can be formulated as follows:

$$Q_t^{l'} = \text{FFN}\left(\text{Norm}\left(Q_t^l + \sum_{F \in F_{in}} \text{CrossAttn}(Q_t^l, F)\right)\right) \tag{1}$$

$$Q_t^{l+1} = \text{FFN}\left(\text{Norm}\left(\text{SpatialSelfAttn}(Q_t^{l'})\right)\right) \tag{2}$$

After L decoder layers, the refined local queries Q_t^L are expected to effectively capture the current observations. These refined queries encapsulate both the spatial and semantic information from the input features, making them highly representative of the objects within the scene.

Query Matching and Fusion. In our approach, we introduce a Dynamic Spatial Memory Bank to efficiently manage and update the queries across steps. Through extensive experiments, we observe that geometric similarity alone is often sufficient for matching queries. This observation motivates our design to leverage the geometric information of object queries to establish correspondences between the current queries and previous queries.

Given the current local queries Q_t^L and previous global queries Q_{t-1}^G, we can obtain their respective bounding boxes: the current local bounding boxes $B_t^L \in \mathbb{R}^{M \times 6}$ and the historical global bounding boxes $B_{t-1}^G \in \mathbb{R}^{N \times 6}$. Here, M and N represent the number of local and global queries, respectively.

Our model is designed to predict global geometry based on partial input, thanks to a box loss function that aligns the predicted bounding boxes with ground-truth global values. This capability is crucial for establishing correspondences between the current and previous queries, even when only partial information is available.

To measure the similarity between the current local queries and previous global queries, we compute the Intersection over Union (IoU) matrix C between the bounding boxes B_t^L and B_{t-1}^G:

$$C = \text{IoU}(B_t^L, B_{t-1}^G) \tag{3}$$

Here, $\text{IoU}(\cdot, \cdot)$ denotes the element-wise IoU score between two sets of axis-aligned bounding boxes. To ensure robust matching, we set elements in C that are smaller than a predefined threshold ϵ to $-\infty$. This step effectively filters out low-confidence matches and focuses on high-similarity pairs.

We then perform matching between B_t^L and B_{t-1}^G based on the cost matrix $-C$. Each current local bounding box b_t^L is assigned to a previous global bounding box b_{t-1}^G if their similarity score is above the threshold. If a new local bounding box $B_t^L[i]$ fails to match with any previous global bounding box b_{t-1}^G, we register the corresponding query $q_t^L[i]$ into the global queries as a new entry. This step ensures that newly detected objects are incorporated into the global representation.

For matched query pairs, we perform fusion to update the global representation. Specifically, instance segmentation masks m_t are fused through a union operation to maintain the most comprehensive segmentation information. For other representations, such as bounding box coordinates, we adopt a weighted average fusion strategy:

$$B_t^G[i] = \frac{n}{n+1} B_{t-1}^G[i] + \frac{1}{n+1} B_t^L[j] \tag{4}$$

Here, we assume that the j-th current local query is matched with the i-th previous global query. The variable n denotes the number of queries that have been merged into $Q_{t-1}^G[i]$ so far. This weighted average approach ensures that the global representation is updated smoothly, incorporating new information while retaining historical context.

Spatial Reasoning. The Spatial Reasoning Transformer is designed to integrate spatial context and language instructions for effective reasoning. It shares a similar transformer architecture with the local query refinement module but includes additional mechanisms to incorporate language

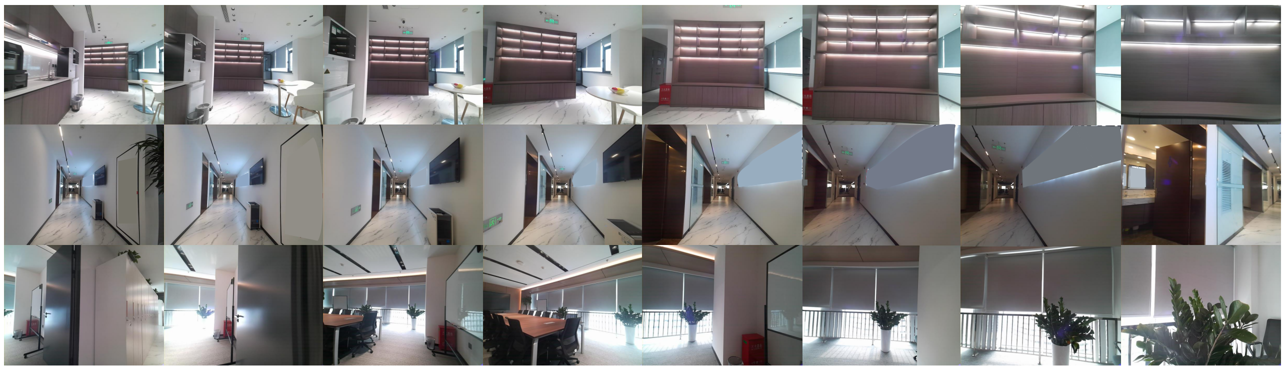

Figure 7. Real world trajectory.

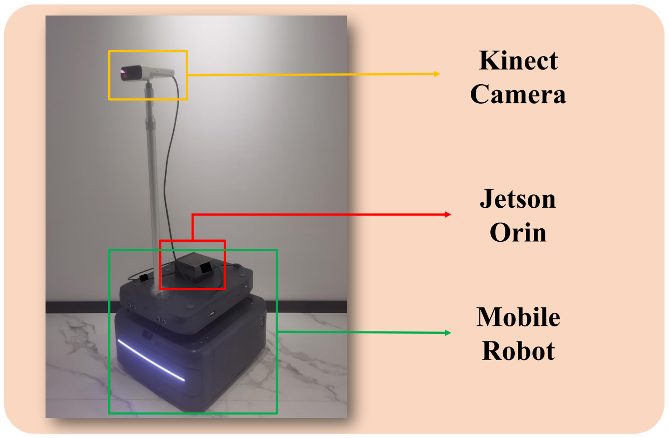

Figure 8. Real Device.

goals. Specifically, the transformer operates on concatenated global and frontier queries, denoted as Q_t^G and Q_t^F respectively, to form the initial queries Q_t^0:

$$Q_t^0 = \text{Concat}(Q_t^G, Q_t^F) \tag{5}$$

For each decoder layer l, the queries first attend to the input features F_{in} to aggregate spatial information from the current observation. This is achieved through a cross-attention mechanism followed by normalization and a feed-forward network (FFN):

$$Q_t^{l'} = \text{FFN}\left(\text{Norm}\left(Q_t^l + \sum_{F \in F_{in}} \text{CrossAttn}(Q_t^l, F)\right)\right) \tag{6}$$

Next, an additional cross-attention layer is introduced to incorporate the language goal L. This step allows the model to align its spatial reasoning with the provided language instructions:

$$Q_t^{l''} = \text{FFN}\left(\text{Norm}\left(Q_t^{l'} + \text{CrossAttn}(Q_t^{l'}, L)\right)\right) \tag{7}$$

Finally, a spatial self-attention layer is applied to capture the spatial relationships among the queries, further refining their representations:

$$Q_t^{l+1} = \text{FFN}\left(\text{Norm}\left(\text{SpatialSelfAttn}(Q_t^{l''})\right)\right) \tag{8}$$

Through the process, the Spatial Reasoning Transformer effectively fuses spatial and linguistic information for further exploration decision.

B. Benchmarks and baseline

HM3D-OVON. HM3D-OVON is an open-vocabulary navigation benchmark. Due to its large-scale test set and resource limitations, we random sample 360 episodes for evaluation. For baselines, BC trains the agent using supervised learning on expert trajectories. DAgger involves an expert providing corrective actions online during training. RL, BCRL, and DAgRL represent reinforcement learning from scratch, reinforcement learning initialized with behavior cloning, and reinforcement learning with DAgger, respectively. Uni-Navid is a video-based navigation method, while TANGO is a training-free navigation approach that leverages large language models.
Goat-Bench. Goat-Bench evaluate multi-modal life long navigation, goals include image, class, and description, due to the large-scale test set and resource limitations, we random sample 90 tasks for evaluation. Modular Goat and Modular clip one wheels represent module approaches using pre-trained detector and feature for zero-shot navigation. SenseAct-NN Skill Chain and SenseAct-NN Monotholic are RL approaches, with single head or multi head for each goat type.
SG3D. Sequential Navigation requires an agent to navigate to a target object in a specified order within a 3D simulation environment. The Embodied Video Agent is a modular

Algorithm 1 Explore an Episode

Global: *explored_map*, *visited_frontiers*, *visible_ids*
END type: **Success**, **Unreachable**, **Invisible**, **Failure**

```
function EXPLORE_AN_EPISODE(strategy, goals)
    decision_list ← []
    while not END do
        Spin and Update
        goal ← select closest goal in goals
        visible ← goal in visible_ids
        reachable ← goal in explored_map
        better_chance ← exist frontier closer to goal
        if visible and reachable then
            Record decision and Goto goal, Success
        else if better_chance then
            Record decision and Goto next frontier
        else if visible but not reachable then
            Unreachable
        else if reachable but not visible then
            Invisible
        else if not visible and not reachable then
            Failure
        end if
    end while
    status = END type
    return (decision_list, status)
end function
```

approach that incorporates persistent memory and utilizes a large language model as the planner. SenseAct-NN Monolithic is identical to the variant used in Goat-Bench, employing a unified reinforcement learning policy for all goal types.

A-EQA. A-EQA evaluates a model's ability to explore an environment in response to a given question. For A-EQA evaluation, our model is solely responsible for generating the exploration trajectory and collecting the corresponding video for each question. The question answering itself is handled by GPT-4o/V. To ensure a fair comparison, we use the same prompts and the same number of video frames as the baseline methods when measuring exploration performance.

C. Trajectory collection

Algorithm 1 outlines our trajectory collection strategy, in which we randomly select each action to simulate the behavior of agents in HM3D. Relying solely on random or ground-truth actions can lead to model overfitting.