RoboTron-Nav: A Unified Framework for Embodied Navigation Integrating Perception, Planning, and Prediction

Supplementary Material

6. Implementation Details

In this section, we elaborate on the more detailed implementations for RoboTron-Nav in Sec. 3.

6.1. Model

6.1.1. Vison Encoder

Following the settings in previous works [19, 32], we input RGB images from both the head perspective I_t^{head} and the wrist perspective I_t^{wrist} into ViT [22] to obtain the 2D features X_t^{head} and X_t^{wrist}. Both X_t^{head} and X_t^{wrist} are then fed into UVFormer [19] to construct multi-view 3D features UV_t.

We adopt UVFormer [19] as our 3D occupancy predictor. As shown in Eq. (2), UVFormer takes the image features X_t, camera parameters Cam, and a set of learnable UniView queries Q as input, and outputs a unified view representation UV_t. The query set $Q = \{Pos, Emb\}$ comprises positional encodings $Pos \in \mathbb{R}^{L\times B\times 3P}$ and learnable embeddings $Emb \in \mathbb{R}^{L\times B\times C}$. Here, L and B (both set to 20) specify the 3D grid's spatial layout within the robot's workspace, and P is the number of uniformly sampled points along the vertical axis of each pillar cell. Each pillar cell covers 0.05^2 square meters on the ground and spans 0.5 meters in height. $Emb^{l,b} \in \mathbb{R}^C$ encodes features for each pillar cell. The camera parameters Cam correspond to N different viewpoints. The unified view representation $UV_t \in \mathbb{R}^{L\times B\times C}$ integrates information from the entire $L \times B \times P$ 3D grid and serves as the basis for downstream occupancy prediction.

In the navigation task, we adhere to the settings in [19, 32], utilizing only the wrist perspective X_t^{wrist} as the 2D feature X_t to broaden the exploration view. Conversely, in the EQA task, we use the head perspective X_t^{head} for the 2D feature X_t to maintain a first-person perspective, as the EQA pairs are generated from this perspective.

6.1.2. LLM

We utilize MPT[1] as our LLM, freezing the self-attention layers during training while fine-tuning the cross-attention layers. For the action head, we employ a multi-layer perceptron to map the final hidden states produced by the LLM from the c-dimensional space to the action space of the CHORES-$\mathbb{S}$ ObjectNav benchmark [10]. For the answer head (i.e., LLM head), we apply the *argmax* operation on the logits output by the LLM to decode the answer.

[1] https://huggingface.co/mosaicml/mpt-1b-redpajama-200b-dolly

6.2. Training Objective

To achieve **Multitask Collaboration**, we design a unified loss function that jointly optimizes navigation actions, question answering, and 3D occupancy through modality-specific components:

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{answer}} + \lambda_{\text{occ}}\mathcal{L}_{\text{occ}}, \tag{7}$$

where $\mathcal{L}_{\text{action}}$, $\mathcal{L}_{\text{answer}}$, and $\mathcal{L}_{\text{occ}}$ denote navigation action prediction loss, embodied question answering loss, and 3D occupancy prediction loss respectively. The term λ_{occ} is the weight coefficient for the occupancy loss.

Action prediction Loss. We utilize behavior cloning to train the navigation model. Given an expert trajectory $\tau = (\hat{a_0}, \cdots, \hat{a_T})$, we use the cross-entropy loss for action prediction. The loss for the trajectory is as follows:

$$\mathcal{L}_{\text{action}} = \sum_{t=1}^{T} \left(CE(a_t, \hat{a}_t)\right), \tag{8}$$

where a_t denotes the predicted action and $\hat{a}_t$ the ground-truth (GT) demonstration at timestep t.

Question answering loss. Given the GT answer $y_{1:K}$ of the input question with the length of K, we optimize the generated answer token probabilities by a conventional cross-entropy loss:

$$L_{\text{answer}} = -\sum_{k=1}^{K} \log\left(p\left(y_k \mid y_{1:k-1}\right)\right). \tag{9}$$

Occupancy loss. Following the approach used in previous works [19, 32], we utilize a standard cross-entropy loss function, denoted as L_{occ}, on the generated 3D volume.

7. Experimental Settings

7.1. Dataset and Metrics

7.1.1. Dataset

We selected the CHORES-$\mathbb{S}$ benchmark for its complex indoor environments (10K rooms) and diverse object categories (15 types), allowing for comprehensive navigation testing. The CHORES-$\mathbb{S}$ ObjectNav benchmark [10] includes 15 object categories and annotates 99k trajectories within 10k training houses, among 5M expert trajectory frames in the AI2-THOR simulated environment [16]. For the CHORES-$\mathbb{S}$ ObjectNav benchmark [10], we extend each trajectory with EQA pairs. As a result, we collect

99k EQA pairs as the corresponding EQA dataset for joint training. The CHORESNAV-$\mathbb{S}$ ObjectNavRoom benchmark [10] is similar to the ObjectNav benchmark but involves smaller trajectories. This benchmark includes 15 object categories and annotates 21k trajectories within 2k training houses out of 1M expert trajectory frames. Additionally, the ObjectNavRoom benchmark uses more diverse instructions, describing both the object's category and its room type simultaneously, such as "Find a vase in the living room." In contrast, the ObjectNav benchmark specifies only the object's category, such as "Find a vase." Similarly, we extend each trajectory in the ObjectNavRoom benchmark with EQA pairs, collecting 21k EQA pairs as the corresponding EQA dataset for ablation studies.

The action space of the ObjectNav and ObjectNavRoom benchmarks [10] includes 20 actions: Move Base (± 20 cm); Rotate Base ($\pm 6^\circ$, $\pm 30^\circ$); Move Arm (x, z) (± 2 cm, ± 10 cm); Rotate Grasper ($\pm 10^\circ$); pickup; dropoff; done with subtask; and terminate.

7.1.2. Metrics

Success rate (SR) is defined as the proportion of episodes deemed successful, which occurs when the agent executes the "end" action and the distance to the target, any instance of the category, is within a specified threshold (e.g., $2m$). **Episode-length weighted success (SEL)** [8] is a metric used to evaluate the efficiency of an agent's navigation. It compares the shortest possible path to the agent's actual path, calculated as:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{w_i}{\max(w_i, e_i)}, \tag{10}$$

where w_i represents the shortest possible episode length to the target object, e_i is the episode length produced by the agent, and S_i is a binary indicator that denotes success for episode i. **Percentage of rooms visited (%Rooms)** is a metric that measures the proportion of distinct rooms an agent successfully visits during navigation relative to the total number of rooms available in the environment. This metric reflects the agent's exploratory capability and efficiency in covering different areas within a given space.

7.2. Traing Strategy

Here, we describe the model hyper-parameters and training details of RoboTron-Nav.

7.2.1. Model Hyper-parameters

In the visual encoder, the number of image patches n_{img} is set to 64, the number of multi-view vision tokens n_{uv} is 400, and the feature dimension c is 1024. In the adaptive 3D-aware history sampling strategy, the window size W is 60, the proximity threshold ϵ is 0.1, and the number of historical frames n_{his} is 60. For the $\mathbf{MaxPool}$ operator, we use an adaptive max pooling function to reduce the number of tokens in the historical features $\mathbf{G}[i].\mathbf{v}$ to 1. In the LLM, the number of language tokens n_{L} corresponds to the length of input instructions and questions, respectively.

7.2.2. Training Details

We train the entire model with the AdamW optimizer using 8 A100 GPUs (80 GB memory per GPU), with a batch size of 48 per GPU, resulting in a total batch size of 384 for 5 epochs. A cosine learning rate strategy is employed, where the learning rate is initially set to 1×10^{-4} and finally decays to 1×10^{-6}. We evaluate checkpoints every 0.5 epoch starting from the 3rd epoch and report the metrics for the checkpoint with the highest SR on the evaluation split.

7.2.3. Training Efficiency and Convergence Stability

In terms of training efficiency, multitask training requires approximately twice as much time as single-task training. Additionally, both approaches demonstrate stable loss reduction and typically converge by the fifth epoch.

8. Qualitative Results

To visualize the effectiveness of our unified framework for embodied navigation, we provide additional qualitative results generated by our method alongside those of SPOC [10]. As shown in Fig.6, when the agent is positioned close to the target, such as within the same room, our RoboTron-Nav is capable of generating the shortest path comparable to SPOC [10]. Both methods understand the environment well, enabling efficient navigation under familiar conditions. However, when the agent needs to navigate across greater distances, such as being situated in different rooms from the target, significant differences in their performance begin to emerge. As shown in Fig. 7, SPOC [10] struggles by repeating paths, which reduces efficiency and increases the risk of looping or missing optimal routes, lowering its success rate. In contrast, RoboTron-Nav avoids revisiting areas, systematically explores new routes, and adapts to changing environments, making it effective for long-distance navigation and optimizing complex pathways.

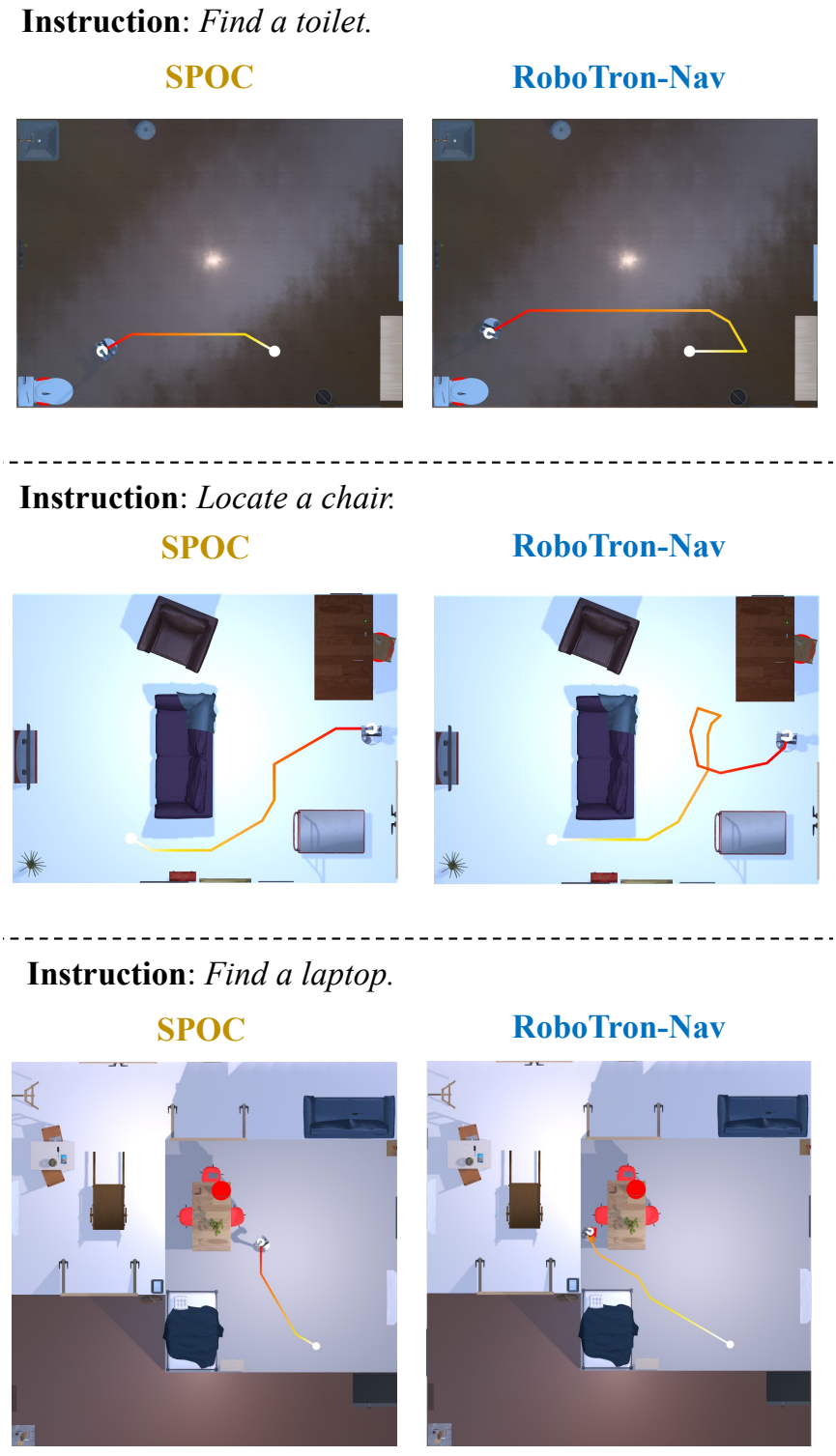

Figure 6. Qualitative comparison of trajectories generated by SPOC [10] and RoboTron-Nav in the same room.

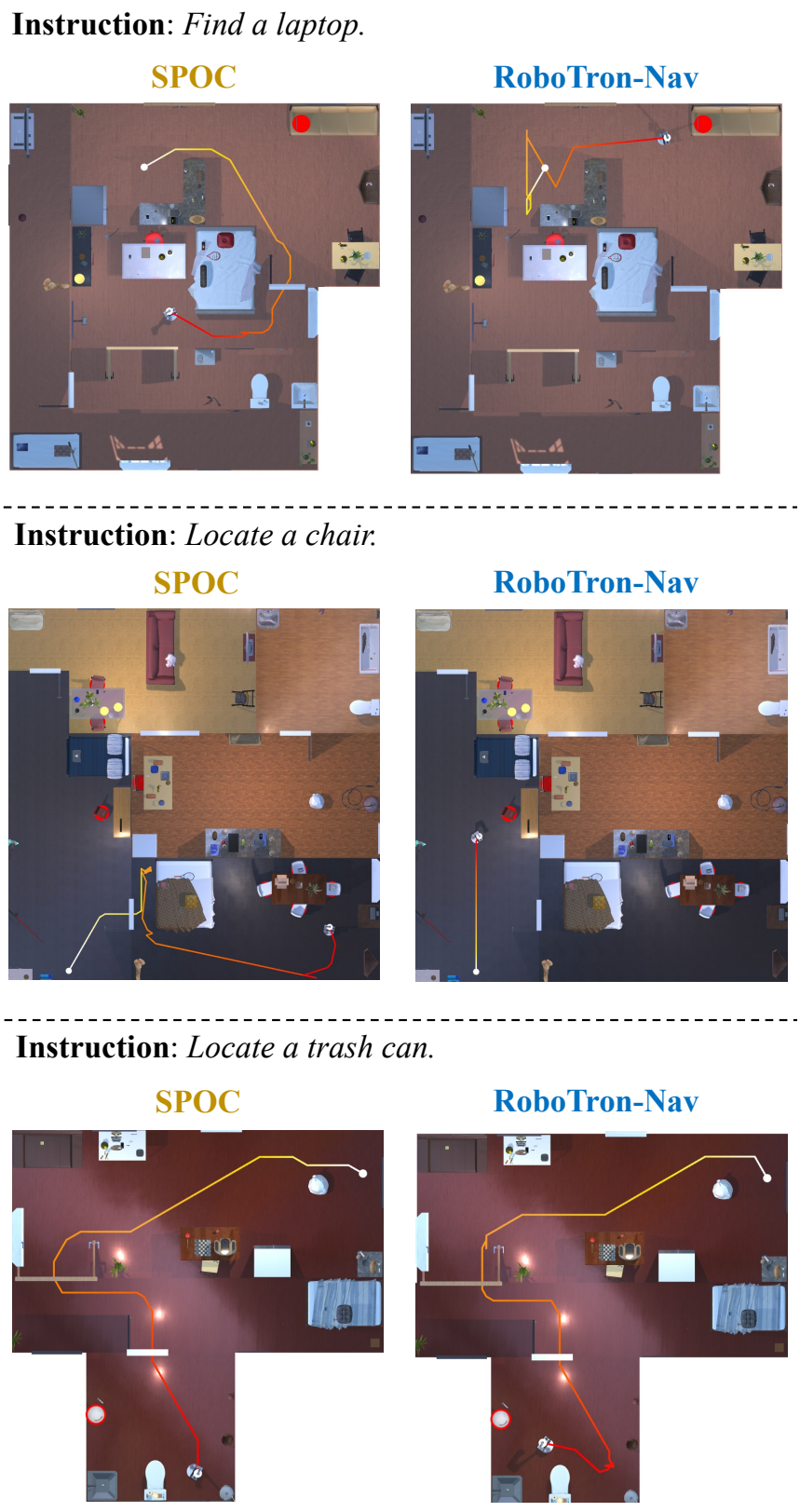

Figure 7. Qualitative comparison of trajectories generated by SPOC [10] and RoboTron-Nav in different rooms.