Appendix

We provide additional information about our method, experiment settings and supporting qualitative visualizations. Below is a summary of the sections in the supplementary:

- Section A reports the details of 3D points construction algorithm.
- Section B reports the details of simultaneously running exploration and identification policies.
- Section C reports the computational cost of the proposed method.
- Section D reports the details of datasets used in our experiments.
- Section E reports additional experiments.
- Section F shows detailed progressive results of the proposed method on Matterport3D.

A. 3D Point Fusion Implementation Details

We introduce the 3D point construction algorithm [53] utilized in our paper (Figure 9). The inputs of the construction algorithm are a sequence of posed color image $I_c^{(t)}$ and depth images $I_d^{(t)}$ at time step t. First, we can obtain the 3D points p via back-projection. Then, we dynamically allocate 3D blocks $\{\mathcal{B}_k\}$, which are composed of the occupied 3D points. To be specific, we divide the 3D world space into a set of adjacent 3D blocks $\{B_k\}$, where each block B_k is defined by the boundary of constant length τ_b along the X, Y and Z axes, *e.g.*, $[X_{min}(B_k), X_{max}(B_k)]$. Two adjacent blocks B_k, B_j along the X axis meet the requirement, $X_{min}(B_k) = X_{max}(B_j)$ or $X_{max}(B_k) = X_{min}(B_j)$. The same requirement holds for Y and Z axes. Given the scene point cloud $P^{(t-1)}$ at time step $t-1$, we allocate all the 3D points into each of the 3D blocks $\{B_k\}$, hence a block-wise point retrieval can be easily achieved.

After constructing the blocks, we can achieve efficient point searching and neighborhood retrieval for any given 3D point p. However, the points within blocks are still unstructured. To obtain the fine-grained relationship of points, we further build a one-level octree $\mathcal{O}_i$ for each point $p_i \in P$. Specifically, for each 3D point back-projected from the instant sensor reading, we perform its nearest neighbor search only among its occupied 3D block and adjacent blocks. Then, we connect the point with the nearest points in the eight quadrants of the Cartesian coordinate system. Now, given any point, we can search the nearest points in eight directions and expand the search region as large as we want. In our implementation, we randomly sample 512 points for each frame. And we only connect the points with distance range in $[4cm, 15cm]$.

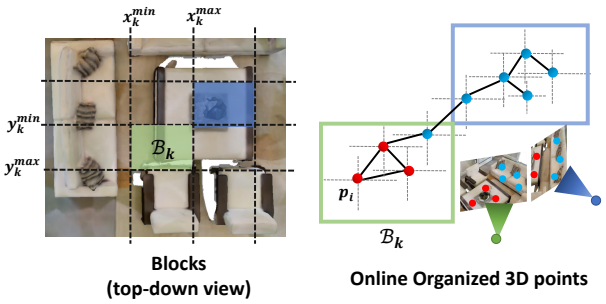


Figure 9. Illustration of online 3D point fusion. **(Left)** dynamically allocated blocks $\mathcal{B}$ based on coordinate intervals . **(Right)** The points p are organized by blocks $\mathcal{B}$ and per-point octrees $\mathcal{O}$, which can be used to query neighborhood points of any given point.

Algorithm 1: Simultaneous Exploration and Identification for 3D-Aware ObjectNav

Input : Existed reconstructed points $P_{l,s,c}^{(ts)}$, RGB image $I_c^{(t)}$, Depth image $I_d^{(t)}$, Agent Pose $o_{pose}^{(t)}$, Target category index o_{ID} at time step t.

Output: Agent action $a^{(t)}$.

// Output a low-level action to drive the agent

// $a^{(t)} \in$ {`move_forward`, `turn_left`, `turn_right` and `stop`}

$P_{l,s}^{(t)} \leftarrow$ `BackProject`$(I_d^{(t)},$ `SemanticPredictor`$(I_c^{(t)}))$;

// Obtain the location and semantic prediction of observed points

$P_{l,s}^{(ts)} \leftarrow$ `PointFusion`$(P_{l,s}^{(ts)}, P_{l,s}^{(t)})$;

$P_c^{(ts)} \leftarrow$ `UpdateConsistency`$(P_{l,s}^{(ts)})$;

// Fuse the new captured points and update the consistency

$g_e^{(t)} \leftarrow$ `ExplorationPolicy`$(P_{l,s,c}^{(ts)},$ `Project`$(P_{l,s,c}^{(ts)}))$;

$\tau^{(t)} \leftarrow$ `IdentificationPolicy`$(P_{l,s,c}^{(ts)})$;

if $p \in P_{l,s,c}$ *satisfy* $p_s > \tau^{(t)}$ *and* `ConsistencyCheck`$(p, o_{ID}, P_{l,s,c})$ **then**

  $g_f^{(t)} \leftarrow$ `Location`(p);

// Simultaneously run the exploration and identification polices

if $g_f^{(t)}$ *exists* **then**

  $a^{(t)} \leftarrow$ `Planning`$(g_f^{(t)})$;

else

  $a^{(t)} \leftarrow$ `Planning`$(g_e^{(t)})$;

return $a^{(t)}$;

B. Pipeline Implementation Details

In Algorithm 1, we describe the the details of simultaneously running exploration and identification policies. Here, we reuse the notions in the main paper.

B.1. Policy Implementation Details

Our **corner-guided exploration policy** takes the 3D observation $x_3D^{(t)}$, 2D observation $x_2D^{(t)}$ and extra information as inputs. The extra information comprises the

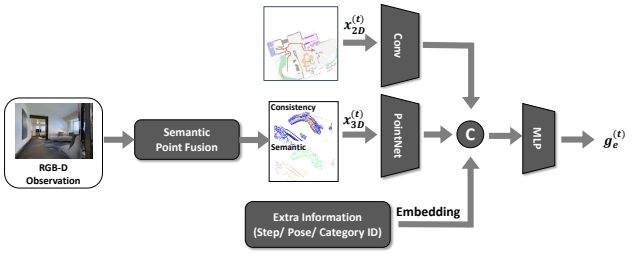


Figure 10. Network architecture of our exploration policy

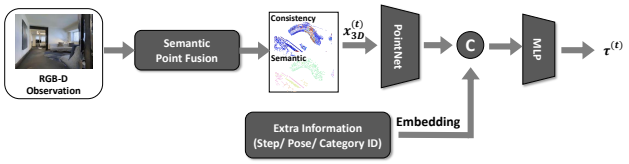


Figure 11. Network architecture of our verification policy

agent's pose, the number of steps, and the target category ID. The proposed exploration policy predicts a discrete corner goal $g_e^{(t)}$ to navigate the robot (Figure 10). Specifically, the policy uses a PointNet [33] to encode the 3D points information (position $p_l^{(t)}$, semantics $p_s^{(t)}$, and consistency $p_c^{(t)}$) to obtain a global feature (256D). The 2D top-down map will be passed to a fully convolutional network [25] and flattened to a feature vector (256D). And the extra information is embedded into a feature vector (24D). Note that the processing of the 2D top-down map and extra information has also been reported in other existing methods [8, 9]. Then, the three feature vectors are concatenated and sent to linear networks, which will output the final target corner goal $g_e^{(t)}$. The **category-aware identification policy** takes 3D observation $x_3D^{(t)}$ and extra information as inputs, and uses the same 3D observation and extra information branches as exploration policy. The identification policy outputs the threshold $\tau^{(t)}$ for target goal selection (See algorithm 1).

C. Computational Cost

Due to the fact that there are always significant overlaps between consecutive frames, when we perform point fusion, we can reuse most of the constructed 3D blocks ($\sim 60\%$). Our algorithm for constructing the 3D scene representation runs at 15 FPS. The memory requirement of one scene can range from 200MB to 500MB during navigation.

We have implemented our core algorithm in python, PyTorch and PyCUDA. Both the point construction and policies run on a workstation with an Intel® Xeon® Gold 6240 CPU CPU @ 3.50GHz × 12 with 64GB RAM and an Nvidia V100 GPU with 32GB memory.

Method	SPL(%)	Improvement
(1.a) 4 corner goal heuristics w/o iden. policy	13.1	-
(1.b) 4 corner goal heuristics w/ iden. policy	**13.9**	6.1% ↑
(2.a) learn continuous goal policy w/o iden. policy	10.1	-
(2.b) learn continuous goal policy w/ iden. policy	**12.7**	25.7% ↑
(**Ours**) learn 4 corner goal w/o iden. policy	13.7	-
(**Ours**) learn 4 corner goal w/ iden. policy	**14.6**	6.6% ↑

Table 6. More ablations on exploration and identification policies.

Noise Setting	SPL (%)	Succ. (%)	DTS (m)
(1) Ours (noiseless)	**14.6**	**34.0**	**4.74**
(2) w. Noisy Pose	13.5	33.1	5.51
(3) w. Gau. Noisy Depth	14.3	33.6	5.36
(4) w. Rdw. Noisy Depth	13.7	31.7	5.45
(5) w. Noisy Depth (Gau.) and Noisy Pose	13.6	31.6	5.50
(6) w. Noisy Depth (Rdw.) and Noisy Pose	13.1	30.1	6.02
(7) PONI (baseline, noiseless)	12.1	31.8	5.10
(8) Stubborn (baseline, noiseless)	13.5	31.2	5.01

Table 7. Results under various noise settings. Gau. indicates Gaussian. Rdw. indicates Redwood.

D. Dataset

Here, we provide further details of the datasets where we validate our method for reference.

Matterport 3D (MP3D) [6] MP3D offers photorealistic reconstructions of building-scale scenes. Following the setting in Habitat Challenge 2021 [1], we consider 21 object categories: *chair*, *table*, *picture*, *cabinet*, *cushion*, *sofa*, *bed*, *chest of drawers*, *plant*, *sink*, *toilet*, *stool*, *towel*, *tv monitor*, *shower*, *bathtub*, *counter*, *fireplace*, *gym equipment*, *seating* and *clothes*. We split the dataset into 61 train / 11 val scenes, containing 2,632,422 / 2,195 episodes, respectively.

MP3D-L2M. In L2M [14], they validate their method on a self-made dataset consisting of 781 episodes from 10 MP3D (val) scenes, which we call MP3D-L2M. It covers 6 object categories: *chair*, *couch(sofa)*, *plant*, *bed*, *toilet* and *tv*. For a fair comparison, we also report our validation results on this MP3D-L2M in main paper table 2.

E. Additional Experiments

Ablation study on exploration and identification policies. As shown in Table 6, 1) coupling our identification policy with exploration heuristics [26]; 2) joint learning our policy with 2D map-based exploration policy [9] till **full convergence** using double training steps of [9]. We observe that the results of various methods are improved by our identification policy, especially the continuous goal exploration strategy (25.7% ↑ on SPL). Moreover, the fully trained continuous goal strategy does not outperform our corner-guided method, due to larger action space and therefore a harder RL problem.

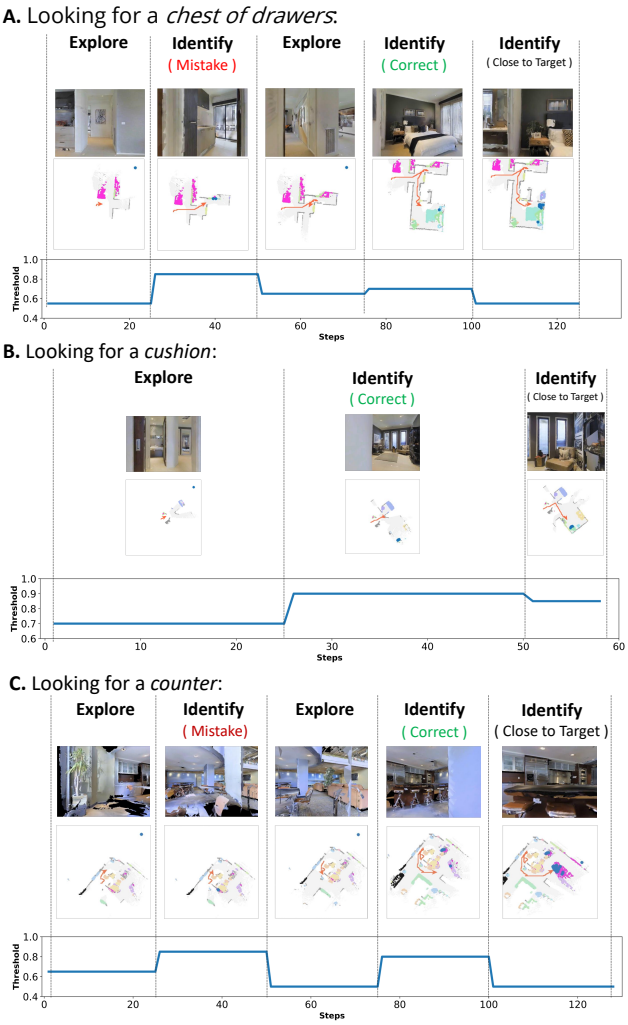


Figure 12. Visualization of threshold changes during navigation.

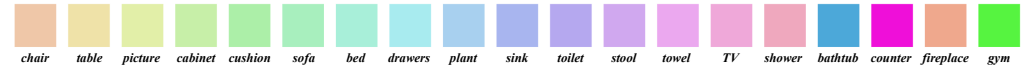
Robustness to noises. We conduct a series of experiments (Table 7) to evaluate our method with noisy pose and various depth noise models. Specifically, for the pose noise, we adopt the same simulation methods as in [38]; for depth noise, we consider Redwood [11] noise model and Gaussian noise model. Under the most challenging noises setting (6), our method has a minor drop of 1.5 point on SPL, which however still performs as a strong competitor against the noiseless baselines (7, 8).

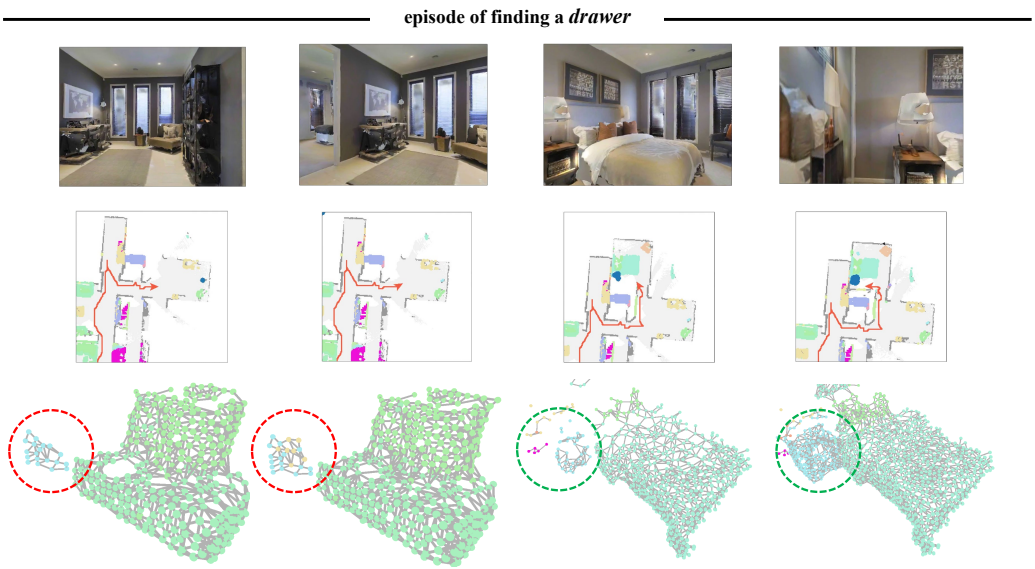
Qualitative examples of the identification policy. Here we provide three qualitative examples (Figure 12) of our identification policy to navigate to various target objects. Note that, the identification policy is executed every 25 steps for the purpose of acceleration. Accordingly, we delineate the exploration and identification phases based on the primary policy utilized during the 25-step interval.

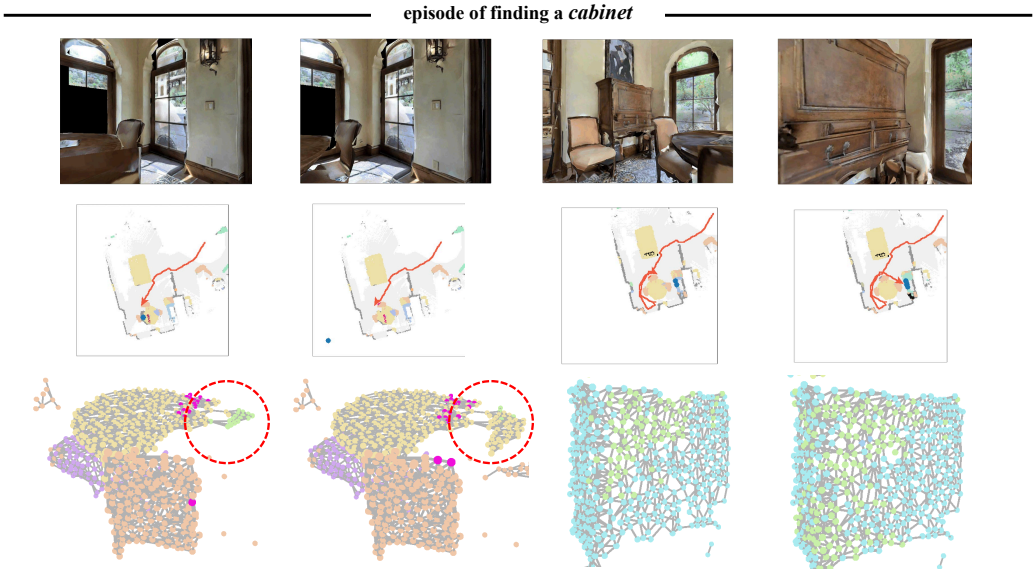
Based on the experimental results, the predicted threshold is dynamically adjusted during navigation. Specifically, when the agent is under the control of the exploration policy, the identification policy predicts a relatively low threshold to facilitate rapid searching of potential targets. Conversely, when the agent is guided by the identification policy, the threshold is fine-tuned to achieve a trade-off between accuracy and efficiency. In general, the identification policy gives a high threshold to ensure a successful stop. Nevertheless, in the event that the agent is in close proximity to the target object, a low threshold is predicted for a quick stop.

F. Visualizing ObjectNav episodes

Figure 13 and Figure 14 illustrate a more detailed visualization of the results obtained through the implementation of our 3D point-based fusion algorithm on the Matterport3D Dataset. It can be observed that during navigation, there is a significant improvement in the semantic prediction and spatial consistency of the points. To further substantiate these findings, Figure 15 to 17 provide visualizations of additional episodes. For a comprehensive demonstration, the attached video in the supplemental material is recommended.







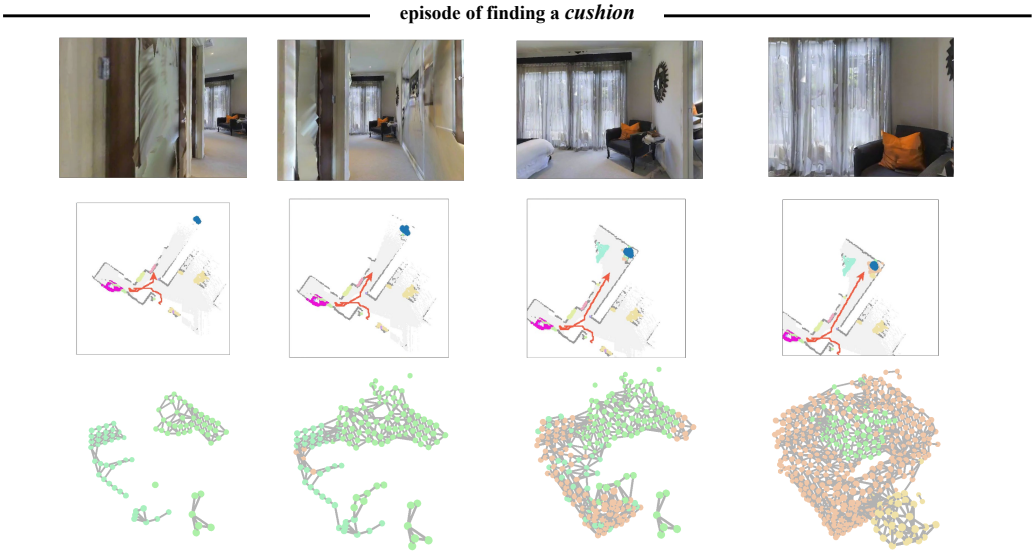
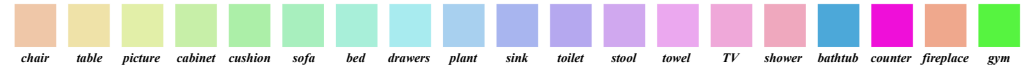


Figure 13. EPS Page1.



episode of finding a *chair*

episode of finding a *sink*

episode of finding a *table*

Figure 14. EPS Page2.

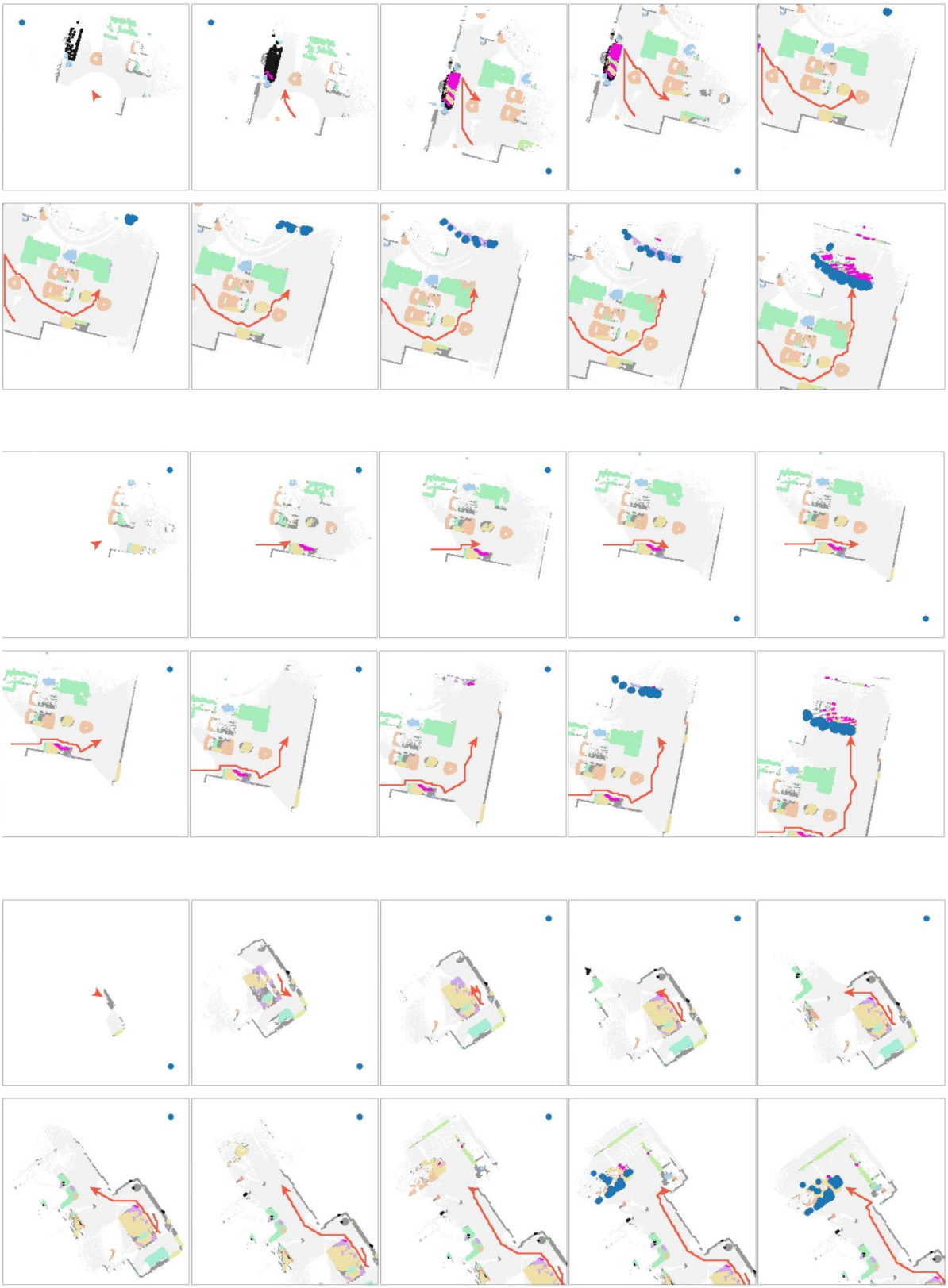


Figure 15. Episode results (1/3).

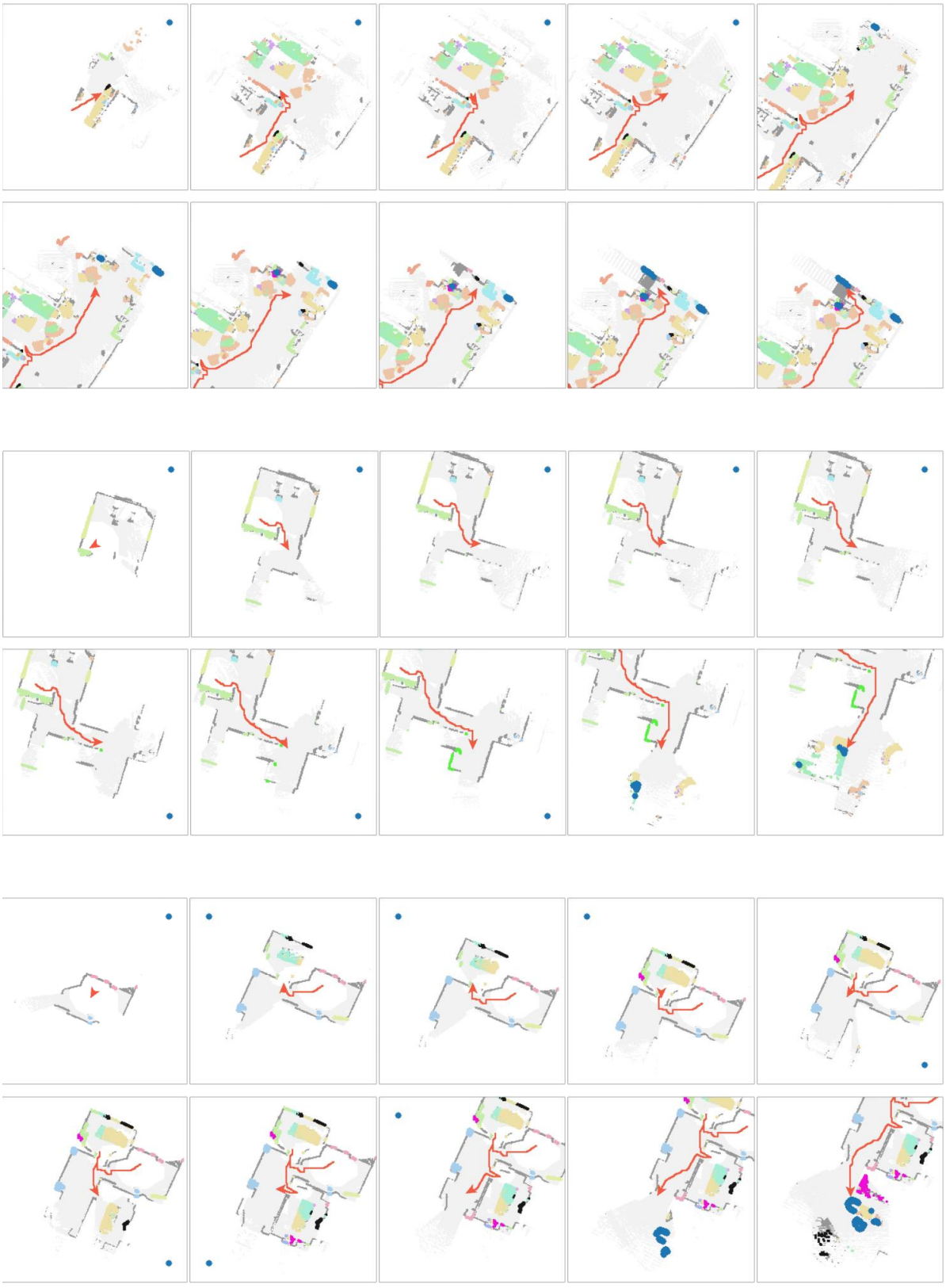


Figure 16. Episode results (2/3).

Figure 17. Episode results (3/3).