What Matters in RL-Based Methods for Object-Goal Navigation? An Empirical Study and A Unified Framework

Author Names Omitted for Anonymous Review.

***Abstract*—Object-Goal Navigation (ObjectNav) is a key capability for deploying mobile robots in everyday environments such as homes, schools, and workplaces. In this task, an agent must locate an instance of a target object category in previously unseen environments using only onboard perception, requiring the integration of semantic understanding, spatial reasoning, and long-horizon planning. Reinforcement learning (RL) has become a dominant paradigm for ObjectNav, yet modern systems involve numerous design choices across perception modules, policy architectures, and inference-time strategies. The relative impact of these components, however, remains poorly understood. In this work, we present a large-scale empirical study of modular RL-based ObjectNav systems. We decompose the navigation pipeline into three key components: *perception*, *policy*, and *test-time enhancement*, and conduct extensive controlled experiments to analyze their individual contributions. Our results suggest that improvements in perception quality and test-time strategies often yield larger performance gains than policy improvements alone, highlighting the importance of understanding how different components interact within modular navigation systems. Motivated by these findings, we introduce a unified framework for systematically studying modular ObjectNav systems. Guided by our analysis, we build an enhanced system that achieves state-of-the-art performance on the Gibson benchmark, improving SPL by 6.6% and success rate by 2.7% over prior methods. We also introduce a human expert baseline, achieving 98% success, highlighting the significant gap between current RL agents and human-level navigation. Finally, we provide practical insights and design recommendations for each module to help guide future research.**

I. Introduction

Recent advances in computer vision and deep learning have inspired growing interest in interdisciplinary applications that bridge perception, reasoning, and control, especially in robotics. Among these, vision-based navigation has emerged as a foundational capability for autonomous mobile agents. A key benchmark in this domain is *Object-Goal Navigation (ObjectNav)*, where a robot must navigate to an instance of a specified object category in an unseen environment, relying solely on its onboard sensors[4, 30, 18, 2]. This task is both practically important and technically challenging: it requires semantic understanding, spatial reasoning, and long-horizon planning. Among many approaches, Reinforcement Learning (RL) has become a dominant paradigm for ObjectNav, offering a structured framework to learn directly through trial-and-error and showing steady progress across various benchmarks[4, 6, 22, 36]. While end-to-end RL policies are common, modular RL approaches have shown greater robustness and improved generalization. By decomposing the system into interpretable and tunable components, such as perception, mapping, policy, and action execution, these methods align with the multi-faceted nature of ObjectNav and often achieve better sim-to-real transfer [10]. However, this modular design also increases overall system complexity, and the standalone contribution of each component has not been systematically studied in recent literature. As a result, current bottlenecks remain unclear, and the lack of a unified design guide makes it difficult to fairly evaluate and advance modular RL systems. Despite its importance, this question has received limited attention in the community, leaving the relative impact of different modules largely unexplored. With these observations, in this work, we first disentangle the components of modular RL-based ObjectNav and ask a fundamental question: *Which design choices truly matter for RL-based ObjectNav?* We establish a principled decomposition of modular ObjectNav systems into three essential parts, **perception**, **policy**, and **test-time enhancement**. For each component, we categorize representative design choices from the literature and, through extensive experiments, isolate the individual contribution of each, an overview of our approach and results is shown in Fig. 1.

Our key findings are summarized as follows: (a) The capability of the perception module has a substantial impact on overall navigation performance. (b) Test-time enhancement strategies are often overlooked, yet they can provide significant improvements during deployment. (c) By establishing a unified configuration for perception and test-time strategies, differences between policy designs can be more reliably measured and analyzed, enabling clearer evaluation and facilitating future improvements in policy learning.

Based on these findings, we offer practical recommendations for selecting and combining system components, along with insights into the reasoning behind them. Following these guidelines, we design an enhanced modular system that achieves state-of-the-art performance with 47.5% SPL (success weighted by path length) and 85.3% success rate, surpassing the best prior methods by 6.6% in SPL and 2.7% in success rate. In addition, we introduce a human baseline comparison, where expert participants operate under the same training setup, test environments, and observation modalities as the agent, achieving an average of 98.0% success rate and 53.3% SPL. This contrast reveals a clear gap between current RL-based systems and human-level navigation, emphasizing the need for new algorithms that can enhance both performance

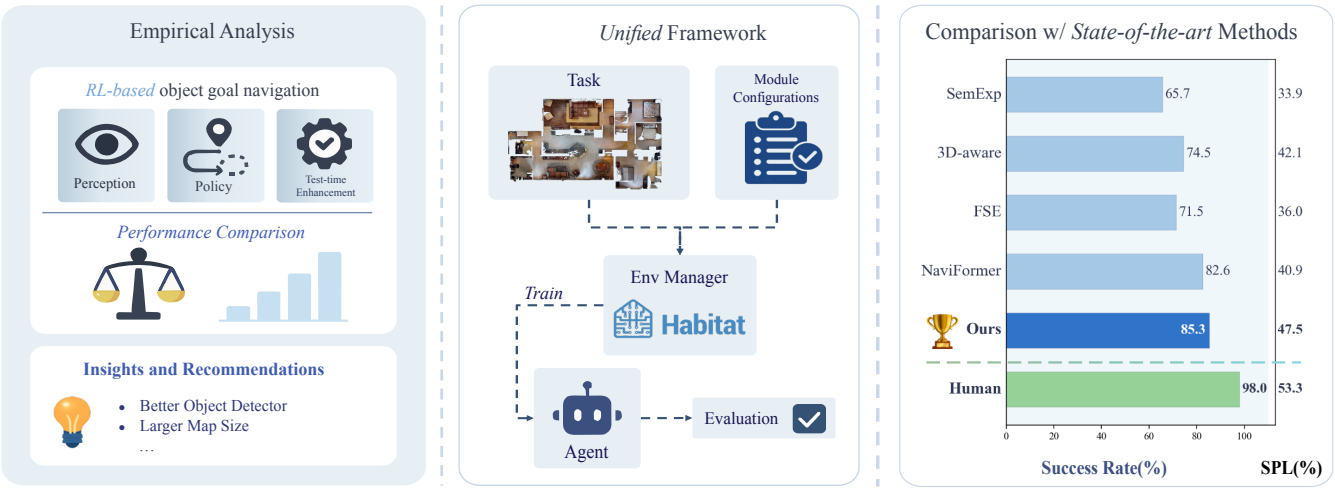

Fig. 1. Our framework encompasses: (1) an empirical study analyzing the impact of different modules, and (2) a unified framework with interchangeable components, enabling users to customize their own object-goal navigation policies.

and robustness in ObjectNav.

Beyond empirical improvements, our goal is to provide broader insights for the vision-based navigation community, from guiding the design of future algorithms to enabling fairer evaluation of modular ObjectNav systems. To facilitate further research, we will publicly release our code and evaluation tools.

II. Related Work

End-to-end Object-Goal Navigation. These approaches leverage reinforcement learning or imitation learning to train a policy that directly maps visual observations to low-level actions. Early works by [42, 23, 24] followed the architectural blueprint of DD-PPO [31], employing convolutional neural networks (CNNs) coupled with recurrent neural networks (RNNs). Subsequent methods [39, 15] enhanced this framework by replacing the RGB encoder with more powerful self-supervised vision models, leading to improved performance. More recent efforts [46, 8] have adopted transformer-based policy architectures to better capture long-term spatial dependencies, achieving state-of-the-art results. To further enrich spatial reasoning, several works have proposed integrating structured scene representations [9] into the policy network. These methods [41, 20, 52, 11, 48] build scene graphs from RGB inputs, encode them with Graph Neural Networks (GNNs), and feed the features into the policy. One major downside of end-to-end methods is that they require large-scale data and struggle to generalize across the sim-to-real gap [10].

Modular RL Methods for Object-Goal Navigation. These approaches integrate learning-based components with classical map-based navigation to achieve improved data efficiency, lower computational overhead, and competitive performance. One representative approach is ANS by [5], which employs a reinforcement learning-based pipeline for navigation in 3D environments. ANS consists of a mapping module that projects RGB-D observations into a top-down occupancy map, along with a long-term goal policy trained via reinforcement learning to guide exploration. Building on ANS, SemExp [4] introduces a semantic mapping module that augments the top-down map with object-level semantic annotations. Subsequent work largely follows this modular reinforcement learning framework with various enhancements. More recently, vision-based navigation has attracted increasing interest from the computer vision community, with several works focusing on improving visual representations and spatial reasoning for embodied navigation [50, 29, 46, 47, 7]. For instance, FSE by [43] incorporates frontier detection into the mapping process and selects sub-goals based on the centers of frontier regions. 3D-Aware [47] further utilizes 3D structural cues to improve scene understanding and predicts corner-based sub-goals, inspired by heuristic strategies like those in [17]. More recently, NaviFormer [36] achieves state-of-the-art performance by replacing traditional map encoders with a transformer-based architecture, enabling richer spatial-semantic representation learning. While existing modular methods primarily focus on improving navigation policies, the relative contribution of different system components remains insufficiently understood. In this work, we present a comprehensive study of modular ObjectNav architectures, systematically evaluating how individual components influence overall navigation performance.

III. Study Design

Problem Formulation. In the context of ObjectGoal navigation, the agent is required to navigate to an instance of a specified object category within an unseen environment. At each timestep t, the agent receives an RGB-D observation $o_t \in \mathbb{R}^{4\times H\times W}$, a goal category label g, and its current pose p_t. Based on these inputs, the agent generates actions to move towards the goal. We use Habitat [19] as a unified simulation testbed, which has a built-in low-level controller that outputs one of four discrete actions: `move forward`, `turn left`, `turn right`, and `stop`.

System Overview. Figure 1 outlines the structure of our

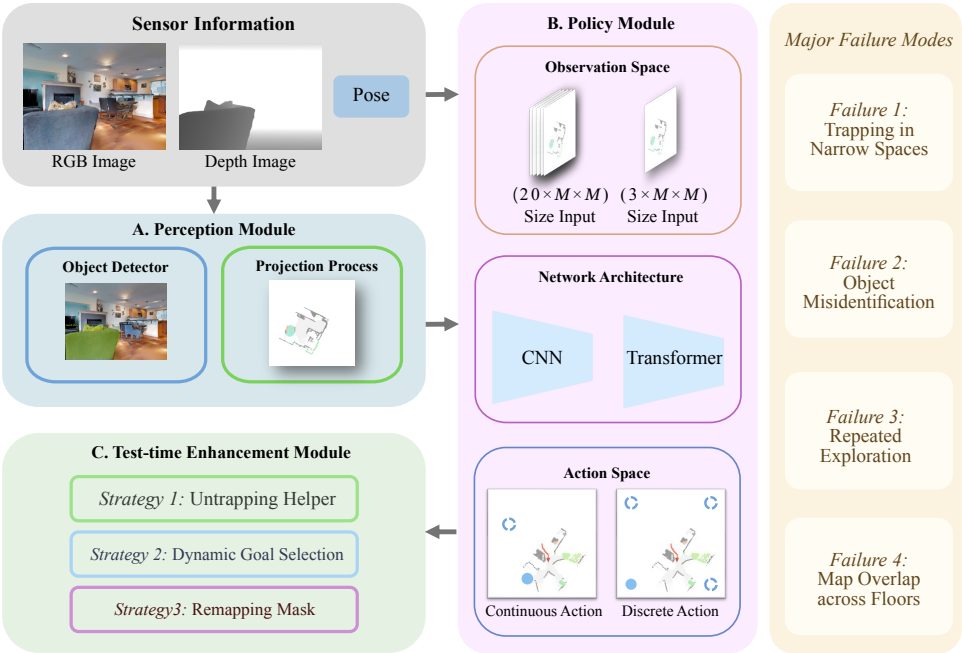

Fig. 2. Unified Framework of our experimental setting. *A. Perception:* RGB-D and pose are fused into a top-down semantic map. *B. Policy:* The map and auxiliary inputs (e.g., category, orientation) guide action prediction. *C. Test-Time Enhancement:* Plug-and-play strategies applied at evaluation to boost performance without retraining. The figure also highlights the major failure modes addressed by these strategies.

study and analysis. We follow the natural pipeline of robot navigation and decompose a typical RL-based approach into three core modules: perception, policy, and test-time enhancement. The perception module covers the process from raw sensor inputs to building representations of the environment. The policy module takes the output of the perception module and predicts the next action based on its action space. The test-time enhancement module contains inference-time strategies that improve performance during execution by adjusting the agent's behavior without retraining. These strategies operate alongside the learned policy to correct common failure patterns and strengthen the overall perception–action loop. For each of these three components, we survey a wide range of design choices found in recent literature and conduct extensive experiments under controlled settings to isolate and quantify their individual contributions to the final navigation performance.

Perception Module Design Choices. A typical representation built by the perception module is a top-down semantic map, which captures both geometry and semantic information based on historical input images. At each timestep, an object detector extracts semantic labels from RGB images, which are then fused with depth to build a structured 3D representation. Aggregating this representation along the height axis yields a 2D semantic map suitable for navigation.

In our study, we focus on three widely adopted design choices for the perception module, *object detector*, *map augmentation*, and *map size*, as these have been repeatedly emphasized in prior ObjectNav systems [5, 4, 43, 47, 36]. Concretely, semantic labels are projected onto a normalized 3D point cloud discretized into a voxel grid, and a 2D top-down map is obtained by aggregating voxels along the height axis: summing all height levels yields an *exploration map*, while summing around the agent's height yields an *obstacle map*. The final map is a $K \times M \times M$ tensor with $K = C + 2$ (one obstacle channel, one exploration channel, and C object-category channels), where the map size M trades off spatial coverage against memory and computation.

Policy Module Design Choices. With the top-down semantic map in place, the policy module infers a goal location that guides the agent's navigation. Typical RL-based policy for long-term goal prediction and identification consists of four key design choices: the *observation space*, the *action space*, the *network architecture*, and the *reward design*. Each of these components influences how the agent interprets the semantic map and decides on navigation strategies. In our experiments, we adopt the common setup established in prior work [5, 4, 43, 36, 33]. For the *observation space*, prior work [4, 43, 36] feeds a high-dimensional $24 \times M \times M$ tensor (local/global maps and auxiliary channels) to the policy; since these channels are typically sparse [17], we additionally consider a compact 3-channel RGB encoding of the same map. For the *action space*, we study three families: *continuous*, where the policy outputs $(a_1, a_2) \in [0, 1]^2$ mapped to map coordinates; *discrete*, where it selects an index over a fixed candidate set (*e.g.*, the four map corners); and *frontier-based*, where it selects among the boundaries between explored and unexplored regions. For the *network architecture*, we compare a ResNet-18 CNN [12] against a ViT-Base backbone [32]. The *reward* combines four terms,

$$r_{\text{exp}} = \alpha_1(A_{\text{cur}} - A_{\text{prev}}), \qquad r_{\text{dist}} = \alpha_2(d_{\text{prev}} - d_{\text{cur}}),$$
$$r_{\text{suc}} = \alpha_3 \mathbb{1}[\text{success}], \qquad r_{\text{step}} = -\alpha_4,$$

where A is the explored area, d the distance to the target, and $\alpha_{1:4}$ are scalar weights. We compare $r_1 = r_{\text{exp}} + r_{\text{dist}}$ against $r_2 = r_{\text{exp}} + r_{\text{suc}} + r_{\text{step}}$.

Policy Learning Algorithm. We adopt PPO [28] as the default learning algorithm for the policy module because it is the dominant choice in prior ObjectNav work and consistently offers stable performance under sparse rewards and long-horizon navigation. This observation is consistent with broader trends in robotics RL, where PPO is the dominant choice across quadrupedal locomotion [16, 13], agile aerial navigation [38, 37], and humanoid control [21] due to its reliability and ease of tuning. To illustrate how policy-learning choices affect performance, we also include a small ablation on PPO's clipping threshold in the Supplementary Materials. The standard setting ($\epsilon = 0.2$) clearly outperforms an extremely restrictive value ($\epsilon = 0.001$), which severely limits learning progress. This highlights the importance of maintaining a reasonable update range for effective policy optimization.

Test-time Enhancement Module Design Choices. Common failure modes found during evaluations are categorized into: (a) trapping in narrow spaces, (b) object misidentification, (c) repeated exploration, and (d) map overlap across floors due to undetected staircases (Figure 2). To mitigate these issues, several strategies have been used in previous work, but were not formally documented. We summarize them into three plug-and-play strategies: (i) an *untrapping helper* to prevent the agent from getting stuck, (ii) *dynamic goal selection* to avoid redundant exploration, and (iii) a *remapping mask* to handle multi-floor map overlap.

IV. Experiments, Results, and Recommendations

Following our definition of three main modules and different design choices for each one, we organize our experiments by analyzing the contribution of individual design choices to overall navigation performance. For each experiment, we provide a detailed setup and report quantitative results, followed by in-depth interpretation and analysis. Alongside our findings, we offer actionable insights and practical recommendations, which can serve as a foundation for designing effective pipelines in future research.

To evaluate the performance of the proposed methods, we consider two evaluation settings: *Fixed timestep* and *Dynamic timestep*. In the fixed-timestep setting, the agent is allowed to interact with the environment for up to 500 steps. In the dynamic timestep setting, the maximum number of steps is set proportional to the effort an optimal planner needs to reach the goal,

$$\text{Max Dynamic Steps} = \alpha \left(\frac{D}{d} + \frac{360^\circ}{\theta} \right),$$

where D is the initial geodesic distance to the target, d the per-step translation, θ the per-step rotation, and α a scaling factor (we use $\alpha = 5$; smaller values make the protocol less discriminative). This stricter budget better exposes differences that the lenient 500-step setting masks. In both cases, the task is considered successful if the agent is within 1 meter of the target object instance at the end of the episode. All policies are trained for 2M frames on NVIDIA RTX 4090 GPUs, within the 0.1–10M range used by prior work [4, 43, 47, 36], where ∼1M frames already yields good performance, so that architecture comparisons are made under matched compute.

We adopt three standard evaluation metrics proposed by [1]: Success Rate (SR): the ratio of successful episodes to the total number of episodes. Success weighted by Path Length (SPL): the ratio between the shortest-path distance and the actual path length taken by the agent, averaged over successful episodes. Distance to Success (DTS): the agent's distance to the target object at the end of an episode. In the Dynamic timestep setting, we report the corresponding variants: D-SR, D-SPL, and D-DTS.

A. Effect of Perception Module

To isolate the influence of the perception module itself, we implement two heuristic-based long-term goal policies that use heuristic rules for navigation instead of a neural network policy conditioned on the top-down semantic map. This allows us to evaluate the Perception Module independently of long-term goal policy effects.

(i) *Corner Goal Policy*: This method adopts a similar action prediction strategy as Stubborn [17]. Instead of selecting goals in a deterministic or fixed pattern, we introduce randomness into the action selection process, similar to MOPA [26]. This enables more exploratory behavior in the environment, allowing us to better evaluate the influence of components in the Perception Module under varied navigation conditions.

(ii) *Frontier-Based Policy*: As there is currently no standard implementation of the Frontier-Based Policy [40], we adopt a similar setup to FSE [43], but replace the learned policy with a random action policy. Since FSE already includes strong heuristic mechanisms for frontier selection, randomly selecting a goal from the list of frontiers serves as a reasonable approximation of the classical Frontier Exploration method.

1) Object Detector: The object detector takes an RGB or RGB-D image as input and outputs the semantic segmentation of the image. It has a dominant impact on overall performance, as the semantic map depends on its output and it directly serves as input to the policy module. We summarize all the object detectors used in the prior works: *Mask R-CNN with Default Checkpoint*: This variant uses the R50-FPN checkpoint from the Detectron2 model zoo [34] without any domain-specific fine-tuning. *Mask R-CNN with Gibson-Finetuned Checkpoint*: This checkpoint, originally introduced in PONI [22], is fine-tuned on the Gibson dataset [35] to better adapt to the domain characteristics of indoor navigation environments. *RedNet*: A network specifically designed for indoor RGB-D semantic segmentation. It is fine-tuned on 100K randomly sampled views from the MP3D [3] dataset. This model was first utilized in Stubborn [17]. Additionally, we evaluate two recent object detectors, *YOLOv11* [14] and *RF-DETR* [27], and further explore combining them with the foundation model *SAM2* [25] to obtain segmentation masks through an alternative approach.

Interpretation. From Table I, we observe that both the Corner Goal Policy and Frontier-Based Policy experiments

TABLE I

STUDY RESULTS OF PERCEPTION MODULE FOR CORNER (C) AND FRONTIER (F) GOAL POLICIES. MRCNN DENOTES MASK R-CNN WITH THE DEFAULT CHECKPOINT, WHILE FT-MRCNN DENOTES MASK R-CNN WITH THE GIBSON-FINETUNED CHECKPOINT.

Pol.	Det.	Size	Aug.	SR (%)	SPL (%)	DTS (m)	D-SR (%)	D-SPL (%)	D-DTS (m)
C	MRCNN	480	✓	77.7	40.9	1.047	61.6	38.6	1.641
C	RedNet	480	✓	80.9	43.9	0.851	65.5	41.6	1.488
C	RF-DETR-Seg	480	✓	77.6	44.2	0.991	64.1	42.4	1.512
C	RF-DETR+SAM2	480	✓	77.1	44.7	1.112	63.8	42.6	1.669
C	YOLO11-XL	480	✓	76.0	42.6	1.147	62.8	40.8	1.668
C	FT-MRCNN	480	✓	83.8	47.6	0.800	69.5	45.3	1.425
C	FT-MRCNN	240	✓	78.7	45.2	1.135	64.8	43.1	1.658
C	FT-MRCNN	480	✗	83.3	44.2	0.771	63.5	41.2	1.727
F	MRCNN	480	✓	71.2	38.5	1.428	49.1	33.5	2.086
F	RedNet	480	✓	75.0	40.9	1.114	53.1	35.7	1.876
F	FT-MRCNN	480	✓	79.9	45.8	0.892	57.2	39.7	1.714
F	RF-DETR-Seg	480	✓	72.9	40.3	1.183	53.6	36.6	1.871
F	RF-DETR+SAM2	480	✓	72.8	43.7	1.416	53.6	39.2	2.006
F	YOLO11-XL	480	✓	69.5	39.0	1.514	49.8	34.6	2.106
F	FT-MRCNN	240	✓	79.6	43.6	1.033	61.0	40.1	1.742
F	FT-MRCNN	480	✗	78.0	42.5	1.000	53.4	36.3	1.791

exhibit similar trends. More advanced perception models generally yield better performance. And a fine-tuned object detector consistently leads to an obvious performance boost across all evaluation metrics. Specifically, for the Corner Goal Policy, using a fine-tuned Mask R-CNN model results in a 6.8% increase in Success Rate, a 6.7% improvement in Success weighted by Path Length, and a 0.247m reduction in Distance to Success. Similar improvement occurs when using our Dynamic metrics. Additionally, although RedNet is not trained on Gibson, fine-tuning on indoor scenes also helps reduce distribution errors.

Recommendation. Our results highlight that the perception module plays a critical role in the overall navigation pipeline, as errors in object detection directly propagate to the semantic map and subsequently influence policy decisions. This suggests that improving the reliability and domain alignment of perception models can substantially enhance downstream navigation performance. In practice, prioritizing robust perception models and ensuring reasonable alignment between the detector's training data and the deployment environment can provide consistent gains in navigation robustness.

2) Map Size: Map size defines the agent's field of view. Larger maps reveal more of the environment, but increase computational cost. To investigate its effect, we experiment with two settings: 240×240 and 480×480. We focus on the local ego-centric map, as the global map is only used to update the local map and is not directly used for navigation. Therefore, when we refer to the top-down semantic map in this context, we mean the local top-down semantic map.

Interpretation. From Table I, we observe that for the Corner Goal Policy, a larger top-down semantic map generally yields better performance than a smaller one. However, this trend does not hold for the Frontier-Based Policy. In this setting, the performance differences between the two map sizes are minimal. In fact, for some Dynamic metrics such as D-SR and D-SPL, the smaller map size slightly outperforms the larger one. We find that map size influences policies differently: larger maps encourage the Corner Goal Policy to explore wider areas and thus improve performance, while the Frontier-Based Policy is less sensitive to map size and can even degrade with higher-resolution maps due to frontier extraction errors.

Recommendation. Our results suggest that the impact of map size depends on how the navigation policy utilizes spatial context. For goal-based policies such as the Corner Goal Policy, larger maps provide a broader spatial context that encourages wider exploration and can improve performance. In contrast, frontier-based policies rely primarily on local frontier extraction and therefore benefit less from increased map resolution. This observation highlights that the optimal map representation should be considered jointly with the policy design rather than treated as an independent system parameter.

3) Map Augmentation: Map augmentation typically refers to the techniques applied during the projection from 3D voxels to 2D maps [43, 44, 36]. It often involves fine-grained enhancements that aim to make the top-down map more accurately reflect the real-world scene. To study its impact, we conduct two sets of experiments with and without augmentation.

Interpretation. As shown in Table I, under the standard evaluation setting, both the Corner Goal Policy and the Frontier-based Policy achieve slightly better or comparable performance when map augmentation is applied. However, under our proposed evaluation framework, map augmentation leads to a significant improvement in performance for both policies. We find that map augmentation mainly improves the quality of the constructed maps, enabling richer and more accurate representations within the same number of steps. This leads to clear gains under stricter evaluation, while the benefit diminishes under standard evaluation, where agents have more time to explore.

Recommendation. Our results indicate that map augmentation improves the fidelity of the constructed semantic map, leading to more accurate spatial representations for navigation. This benefit becomes particularly evident under stricter evaluation settings, where the agent must build reliable scene representations within a limited number of steps. Under more relaxed settings, where the agent has sufficient time to explore and correct mapping errors, the impact of map augmentation becomes less pronounced. These findings suggest that map augmentation is especially valuable in scenarios requiring efficient exploration and rapid environment understanding.

B. Effect of Policy Module

The policy module plays a central role in predicting goals for the agent to reach, and its decision-making process directly influences the agent's ability to explore the environment effectively. To better isolate the contribution of different components within the policy module, we fix the parameters of the perception module. Additionally, when analyzing the effect of a single component, we keep all other factors constant.

1) Observation Space: To investigate the impact of different observation space choices on performance, we consider two options: *Top-down semantic map (standard)*: Following most prior work, this setup uses multiple channels to store different types of information, including the exploration map, obstacle map, frontier map, and semantic maps. *Top-down semantic map (compressed)*: In this setting, we propose to use a compressed version of the top-down semantic map. Specifically, we compress the original 20 channel semantic map into a single 3-channel RGB image.

Interpretation. Table II shows that the performance differences between the standard and compressed configurations are minimal: less than 0.5% in SR and D-SR, less than 0.7% in SPL and D-SPL, and less than 0.1m in DTS and D-DTS. These negligible differences suggest that the compressed top-down semantic map configuration retains sufficient spatial information for effective policy learning, while reducing the observation size by more than a factor of six.

Recommendation. Our results indicate that compressed observations preserve the spatial information required for policy learning while significantly reducing the observation dimensionality, suggesting that compact representations can improve efficiency without degrading navigation performance.

2) Action Space: The action space determines how the policy generates goals on the map. We investigate the impact of different action space designs by considering the following two options: *Continuous Action Space*: The policy predicts a goal location directly on the map. The goal can be any coordinate within the map boundaries. *Discrete Action Space*: A set of possible goal locations, typically the four corners of the map, is predefined as candidate actions. The policy then predicts the index of one of these predefined goal locations.

Interpretation. The 2nd and 3rd row of Table II shows that continuous action space outperforms the discrete one. Compared with the continuous action space, the discrete action space is more akin to predicting a direction for the agent rather than an exact goal. Given the scale of the map, the agent's movement range during local policy navigation is limited and often insufficient to directly reach the final goal. As a result, the selected goal in the discrete setting effectively serves as a directional signal, guiding the agent toward further exploration. Therefore, the discrete action space is coarser compared to the continuous action space, as it provides less precise control over goal selection.

Recommendation. Without additional strategies to further enhance the discrete action space, the continuous action space is a more favorable choice.

3) Network Architecture: Previous works commonly used CNNs to process the observations, more recent studies have shifted to using Transformers. Specifically, in our study, we choose: *CNN*: For the CNN-based model, we adopt ResNet-18 [12] as the policy backbone. *Transformer*: For our transformer model, we adopt the ViT-Base architecture [32] as the backbone for map feature extraction.

Interpretation. The results in Table II indicate no significant difference in performance between the two network architectures. These findings indicate that, when keeping other modules unchanged, varying the network architecture for map feature extraction does not substantially affect the final outcomes. This also indicates that the overall performance is sensitive to the policy network as other modules.

Recommendation. Our results show no significant performance difference between CNN and Transformer backbones for map feature extraction. This may be because the structured top-down semantic map already encodes much of the spatial information required for navigation, reducing the need for highly expressive feature extractors. Consequently, lightweight CNN backbones can offer a more efficient alternative without noticeable performance degradation.

4) Reward Design: Reward design is always a critical challenge for reinforcement learning algorithms. In this study, We consider two types of reward designs: $r_1 = r_{\text{exploration}} + r_{\text{distance to target}}$, and $r_2 = r_{\text{exploration}} + r_{\text{success}} + r_{\text{step penalty}}$.

Interpretation. Table II shows that the policy trained with r_1 outperforms that trained with r_2, yielding a 0.8% improvement in Success Rate and a 2.5% improvement in SPL. With r_1, the agent is rewarded for moving closer to the target object, with semantic information implicitly embedded in the reward signal. In contrast, r_2 is designed solely for exploration, where only task success is rewarded and additional steps are penalized; the agent receives no reward for approaching the target.

Recommendation. Our results suggest that incorporating a distance-to-target component in the reward function provides a denser and more informative learning signal during policy optimization. Compared to purely exploration-driven rewards, this intermediate progress signal helps reduce ambiguity in the navigation objective and encourages the policy to align exploration with goal-directed behavior. These findings highlight the importance of reward structures that provide meaningful progress signals in long-horizon navigation tasks.

C. Effect of Test-time Enhancement Module

To improve robustness during deployment, many ObjectNav systems incorporate simple test-time strategies alongside the

TABLE II
STUDY RESULTS OF KEY ARCHITECTURAL AND TRAINING CHOICES, INCLUDING OBSERVATION/ACTION SPACES, NETWORK ARCHITECTURES, AND REWARD DESIGN.

Observation Space	Action Space	Network Arch.	Reward Design	SR (%)	SPL (%)	DTS (m)	D-SR (%)	D-SPL (%)	D-DTS (m)
Standard	Discrete	Transformer	Type 2	83.4	43.8	0.717	66.3	41.0	1.501
Compressed	Discrete	Transformer	Type 2	83.8	43.3	0.685	65.8	40.3	1.572
Compressed	Continuous	Transformer	Type 2	84.4	48.2	0.741	69.6	45.8	1.415
Compressed	Discrete	CNN	Type 2	83.7	43.3	0.706	65.8	40.5	1.548
Compressed	Discrete	Transformer	Type 1	85.2	45.8	0.664	67.8	43.0	1.346

learned policy. Although these techniques are widely implemented in practice, they are rarely discussed or systematically analyzed in prior work, as they are often treated as engineering details rather than research contributions. In this section, we summarize several commonly used strategies and investigate their impact on navigation performance through controlled experiments.

1) Untrapping helper.: The untrapping helper uses predefined rules to help the agent escape from stuck situations. To assess its impact, we compare our continuous-action RL policy with and without the helper enabled.

Interpretation. As shown in Table III, the use of the untrapping helper clearly improves performance in both evaluation settings. The improvement is more obvious in the standard evaluation. Furthermore, we observe that the untrapping helper has a greater impact on long-distance navigation. When the target goal is farther from the agent, the likelihood of the agent getting trapped increases due to the extended navigation path. This explains why the performance improvement is more pronounced in the standard evaluation setting.

Recommendation. Our results show that simple recovery mechanisms can significantly improve navigation robustness. In particular, the untrapping helper helps the agent recover from local navigation failures without modifying the learned policy.

2) Dynamic goal selection: Dynamic goal selection is a strategy originally designed for heuristic discrete policies to prevent the agent from engaging in redundant exploration. We further demonstrate that it can also be incorporated into RL-based discrete action space policies to guide the local policy more efficiently. To assess its impact, we compare the performance of our trained discrete policy with and without this mechanism.

Interpretation. As shown in Table III, the improvement provided by dynamic goal selection is significant. In both evaluation settings, we observe substantial performance gains. This is primarily because dynamic goal selection effectively reduces repeated exploration, thereby enhancing navigation efficiency.

Recommendation. Our analysis shows that dynamic goal selection reduces repeated exploration by discouraging the agent from revisiting previously explored regions. This highlights the benefit of lightweight exploration management strategies alongside learned policies.

3) Remapping mask: The remapping mask assists the agent in regenerating the map when overlapping issues occur during navigation. To evaluate the effectiveness of this mechanism, we perform an ablation study using our trained continuous action space policy.

Interpretation. Similar to the effect of the untrapping helper, the remapping mask has a more significant impact in the standard evaluation setting, as shown in Table III, compared to our more stringent setting. This is because long distance navigation increases the likelihood of the agent encountering map overlap issues, such as revisiting previously seen areas from different floors (e.g., via stairs). In contrast, during short distance navigation tasks, if the agent can reach the target object within just a few steps, it is unlikely to traverse into stair regions that could cause overlaps in the top-down semantic map. As a result, the benefit of the remapping mask in these short scenarios is minimal.

Recommendation. Our results suggest that the remapping mask improves robustness when map inconsistencies occur, such as during multi-floor navigation. Maintaining reliable map representations is therefore important for stable navigation.

D. Comparisons with the State-of-the-art and Human experts

After clearly understanding the impact of different components from various modules on overall performance, we selected four policies, each configured with the best-performing components, to compare against prior work. Standard evaluation metrics were used for this comparison. To enable comparison with state-of-the-art methods, we design four customized policies: (i) Corner Goal Policy, (ii) Frontier-Based Policy, (iii) RL policy with a discrete action space, and (iv) RL policy with a continuous action space. We report in Table IV the results from the policies using a continuous action space. For all policies, we adopt the best-performing configurations across all modules. From Table IV, we observe that after thoroughly

TABLE III
STUDY RESULTS OF TEST-TIME ENHANCEMENT STRATEGIES, INCLUDING UNTRAPPING HELPER, DYNAMIC GOAL SELECTION AND REMAPPING MASK.

Policy	Untrapping Helper	Dynamic Goal Selection	Remapping Mask	SR (%)	SPL (%)	DTS (m)	D-SR (%)	D-SPL (%)	D-DTS (m)
RL (Continuous)	✗	✓	✓	81.3	47.3	0.741	69.1	45.3	1.509
RL (Continuous)	✓	✓	✗	82.6	48.0	0.817	70.1	45.9	1.384
RL (Continuous)	✓	✓	✓	84.4	48.2	0.741	69.6	45.8	1.415
RL (Discrete)	✓	✗	✓	83.8	43.3	0.685	65.8	40.3	1.572
RL (Discrete)	✓	✓	✓	85.3	47.5	0.632	69.8	44.9	1.397

analyzing the impact of different components, our configured policy outperforms previous state-of-the-art algorithms on the Gibson benchmark. Since modular methods consist of multiple components that jointly contribute to overall performance, it is essential to understand the impact of each component. Without a clear analysis of these components, focusing solely on designing complex policy networks is unlikely to achieve optimal performance. Moreover, complex policy designs often introduce additional challenges, such as increased computational cost and reduced inference speed. Hence, a clear understanding of modular components is essential and can provide valuable insights for future research.

To better contextualize system performance, we introduce a human expert baseline under the same training setup, test environments, and observation modalities as our agents. Five researchers with robotics expertise (but no prior ObjectNav experience) were given practice on the training scenes before evaluation. As shown in Table IV, they achieved near-perfect performance, with an average SR of 98%, consistently outperforming all existing autonomous methods. This highlights the substantial *algorithmic* gap between human and system performance, underscoring the need for more intelligent perception, reasoning, and exploration strategies to approach human-level robustness.

TABLE IV
COMPARISONS WITH PRIOR METHODS ON GIBSON. THE METRICS OF THE TWO BEST-PERFORMING METHODS ARE HIGHLIGHTED IN GREEN AND ORANGE. OUR METHOD OUTPERFORMS ALL PRIOR APPROACHES.

Method	SR(%)	SPL(%)	DTS(m)
SemExp [4]	65.7	33.9	1.47
SemExp* [22]	71.7	39.6	1.39
PONI [22]	73.6	41.0	1.25
3D-aware [47]	74.5	42.1	1.16
FSE [43]	71.5	36.0	1.35
SGM [49]	78.0	44.0	1.11
L3MVN [44]	76.8	38.8	1.01
NaviFormer [36]	82.6	40.9	0.76
T-Diff [45]	79.6	44.9	1.00
HOZ++ [51]	78.2	44.9	1.13
Ours	84.4	48.2	0.74
Human Experts	98.0	53.3	0.26

E. Failure Case Analysis

To understand where the remaining gap comes from, we manually inspect 1,000 test episodes (200 per policy) across five scenes. We label 880 episodes as successful and 120 as failures; an additional 21 episodes succeed by human judgment but are counted as failures under Habitat's strict success criterion. Among the genuine failures, *68 are caused by detector mis-detections* (most often a bed misread as a sofa), making perception by far the dominant error source, directly reinforcing our finding that the perception module governs overall performance. The remaining failures stem from issues that test-time strategies target: 17 from getting trapped in narrow spaces, 15 from incomplete exploration within the step budget, 11 from navigating to the wrong floor, 7 from not stopping after locating the target, and 2 from map-localization errors. This breakdown explains both why stronger detectors yield outsized gains and why simple test-time enhancements meaningfully improve robustness.

V. CONCLUSION

In this work, we present a large-scale empirical study that systematically analyzes the contributions of different components in RL-based Object-Goal Navigation systems. By decomposing these systems into three key modules, perception, policy, and test-time strategies, we provide a clearer understanding of how different design choices influence navigation performance. Our analysis shows that improvements in perception quality and test-time strategies can lead to substantial gains in navigation robustness and efficiency, while controlled configurations enable more reliable evaluation of policy designs. We further demonstrate that simple test-time strategies, such as dynamic goal selection and untrapping helpers, can significantly improve system robustness without requiring additional training. Guided by these findings, we develop an enhanced modular navigation system that achieves state-of-the-art performance on standard benchmarks. In addition, our human baseline evaluation highlights the significant gap between current autonomous agents and human-level navigation. Beyond empirical improvements, this study provides practical insights and design recommendations for modular ObjectNav systems, offering a foundation for future research on more robust evaluation, improved system design, and real-world deployment of embodied navigation agents.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018. URL https://arxiv.org/abs/1807.06757.

[2] Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. Cognav: Cognitive process modeling for object goal navigation with llms. *arXiv preprint arXiv:2412.10439*, 2024.

[3] 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. *International Conference on 3D Vision (3DV)*, 2017.

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *In Neural Information Processing Systems*, 2020.

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[6] Devendra Singh Chaplot, Murtaza Dalal, Saurabh Gupta, Jitendra Malik, and Russ R Salakhutdinov. Seal: Self-supervised embodied active learning using exploration and 3d consistency. *Advances in neural information processing systems*, 34:13086–13098, 2021.

[7] Heming Du, Lincheng Li, Zi Huang, and Xin Yu. Object-goal visual navigation via effective exploration of relations among historical navigation states. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2563–2573, 2023.

[8] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, Ranjay Krishna, Dustin Schwenk, Eli VanderBilt, and Aniruddha Kembhavi. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world, 2024. URL https://arxiv.org/abs/2312.02976.

[9] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14849–14859, 2022.

[10] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *Science Robotics*, 8(79):eadf6991, 2023. doi: 10.1126/scirobotics.adf6991. URL https://www.science.org/doi/abs/10.1126/scirobotics.adf6991.

[11] Jingwen Guo, Zhisheng Lu, Ti Wang, Weibo Huang, and Hong Liu. Object goal visual navigation using semantic spatial relationships. In *CAAI international conference on artificial intelligence*, pages 77–88. Springer, 2021.

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. URL https://api.semanticscholar.org/CorpusID:206594692.

[13] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[14] Glenn Jocher and Jing Qiu. Ultralytics yolo11. https://github.com/ultralytics/ultralytics, 2024. Version 11.0.0, AGPL-3.0 License.

[15] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[16] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47):eabc5986, 2020.

[17] Haokuan Luo, Albert Yue, Zhang-Wei Hong, and Pulkit Agrawal. Stubborn: A strong baseline for indoor object navigation, 2022. URL https://arxiv.org/abs/2203.07359.

[18] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimír Vondruš, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[19] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, Vladimír Vondruš, Theophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023. URL https://arxiv.org/abs/2310.13724.

[20] Yiding Qiu, Anwesan Pal, and Henrik I. Christensen. Learning hierarchical relationships for object-goal navigation, 2020. URL https://arxiv.org/abs/2003.06749.

[21] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[22] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning, 2022. URL https://arxiv.org/abs/2201.10029.

[23] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale, 2022. URL https://arxiv.org/abs/2204.03514.

[24] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav, 2023. URL https://arxiv.org/abs/2301.07302.

[25] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

[26] Sonia Raychaudhuri, Tommaso Campari, Unnat Jain, Manolis Savva, and Angel X Chang. Mopa: Modular object navigation with pointgoal agents. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5763–5773, 2024.

[27] Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. Rf-detr: Neural architecture search for real-time detection transformers, 2025. URL https://arxiv.org/abs/2511.09554.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[29] Hardik Shah, Jiaxu Xing, Nico Messikommer, Boyang Sun, Marc Pollefeys, and Davide Scaramuzza. Foresightnav: Learning scene imagination for efficient exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2025.

[30] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat, 2022. URL https://arxiv.org/abs/2106.14405.

[31] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, 2020. URL https://arxiv.org/abs/1911.00357.

[32] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision, 2020.

[33] Qiaoyun Wu, Jun Wang, Jing Liang, Xiaoxi Gong, and Dinesh Manocha. Image-goal navigation in complex environments via modular learning. *IEEE Robotics and Automation Letters*, 7(3):6902–6909, 2022.

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[35] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[36] Wei Xie, Haobo Jiang, Yun Zhu, Jianjun Qian, and Jin Xie. Naviformer: A spatio-temporal context-aware transformer for object navigation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14):14708–14716, Apr. 2025. doi: 10.1609/aaai.v39i14.33612. URL https://ojs.aaai.org/index.php/AAAI/article/view/33612.

[37] Jiaxu Xing, Ismail Geles, Yunlong Song, Elie Aljalbout, and Davide Scaramuzza. Multi-task reinforcement learning for quadrotors. *IEEE Robotics and Automation Letters*, 2024.

[38] Jiaxu Xing, Angel Romero, Leonard Bauersfeld, and Davide Scaramuzza. Bootstrapping reinforcement learning with imitation for vision-based agile flight. *arXiv preprint arXiv:2403.12203*, 2024.

[39] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation, 2022. URL https://arxiv.org/abs/2204.13226.

[40] Brian Yamauchi. A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997. URL https://api.semanticscholar.org/CorpusID:206561621.

[41] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors, 2018. URL https://arxiv.org/abs/1810.06543.

[42] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectnav, 2021. URL https://arxiv.org/abs/2104.04112.

[43] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. Frontier semantic exploration for visual target navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, page 4099–4105. IEEE, May 2023. doi: 10.1109/icra48891.2023.10161059. URL http://dx.doi.org/10.1109/ICRA48891.2023.10161059.

[44] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3554–3560. IEEE, October 2023. doi: 10.1109/iros55552.2023.10342512. URL http://dx.doi.org/10.1109/IROS55552.2023.10342512.

[45] Xinyao Yu, Sixian Zhang, Xinhang Song, Xiaorong

Qin, and Shuqiang Jiang. Trajectory diffusion for objectgoal navigation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 110388–110411. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c72861451d6fa9dfa64831102b9bb71a-Paper-Conference.pdf.

[46] Kuo-Hao Zeng, Zichen Zhang, Kiana Ehsani, Rose Hendrix, Jordi Salvador, Alvaro Herrasti, Ross Girshick, Aniruddha Kembhavi, and Luca Weihs. Poliformer: Scaling on-policy rl with transformers results in masterful navigators, 2024. URL https://arxiv.org/abs/2406.20083.

[47] Jiazhao Zhang, Liu Dai, Fanpeng Meng, Qingnan Fan, Xuelin Chen, Kai Xu, and He Wang. 3d-aware object goal navigation via simultaneous exploration and identification, 2023. URL https://arxiv.org/abs/2212.00338.

[48] Sixian Zhang, Xinhang Song, Yubing Bai, Weijie Li, Yakui Chu, and Shuqiang Jiang. Hierarchical object-to-zone graph for object navigation, 2021. URL https://arxiv.org/abs/2109.02066.

[49] Sixian Zhang, Xinyao Yu, Xinhang Song, Xiaohan Wang, and Shuqiang Jiang. Imagine before go: Self-supervised generative map for object goal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16414–16425, June 2024.

[50] Sixian Zhang, Xinyao Yu, Xinhang Song, Xiaohan Wang, and Shuqiang Jiang. Imagine before go: Self-supervised generative map for object goal navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16414–16425, 2024.

[51] Sixian Zhang, Xinhang Song, Xinyao Yu, Yubing Bai, Xinlong Guo, Weijie Li, and Shuqiang Jiang. Hoz++: Versatile hierarchical object-to-zone graph for object navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2025. doi: 10.1109/TPAMI.2025.3552987.

[52] Kang Zhou, Chi Guo, and Huyin Zhang. Visual navigation via reinforcement learning and relational reasoning. In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pages 131–138, 2021. doi: 10.1109/SWC50871.2021.00027.