Novelty-Driven Visual Exploration for Assistive Indoor Navigation

Viriyadhika Putra
Department of Computer Science
University of Toronto
Toronto, Canada
viriyadhika.putra@mail.utoronto.ca

Darpal Patel
Department of Computer Science
University of Toronto
Toronto, Canada
darpal.patel@mail.utoronto.ca

***Abstract*—To assist those with visual impairments, we investigate the behaviors of visual encoder and Transformer-based architectures in the context of autonomous indoor exploration without explicit task objectives. Unlike traditional navigation benchmarks, our agents rely on intrinsic motivation derived from visual novelty, without depth and pose estimation. We evaluate a range of encoders and model configurations on downstream tasks to determine the effectiveness of each architecture. Even in constrained or suboptimal regions, we find the agent is able to escape and converge toward stable local optima in certain architectures. The results indicate that compact models can support navigation-like behaviors in visually rich environments, highlighting the potential for minimalist vision-based exploration frameworks in embodied artificial intelligence.**

***Index Terms*—[Reinforcement learning, embodied AI, intrinsic motivation, vision-language models, segmentation, indoor navigation]**

I. Introduction

People with visual impairments often require assistance with navigation tasks and object-finding tasks. Existing tools such as Microsoft's SeeingAI [1] provide real-time scene descriptions through a smartphone camera, but they rely entirely on the user's physical movement and do not build a persistent understanding of the environment. To create more autonomous assistive systems, an important sub-problem is enabling a robot to identify and locate items throughout an indoor space. While PointNav (navigating to a specific point) and ObjectNav (navigating to a specific object) are well-studied tasks [2], [3], both assume that the agent is given an explicit immediate goal. In contrast, we frame our problem around an an indoor-based agent that has to periodically explore its surroundings and maintain a memory of what it has previously observed, allowing it to support object-finding queries without requiring multiple rounds of search.

To achieve this, we require an effective exploration algorithm that operates without any predefined objective, as future user queries are unknown. Motivated by these ideas, our reinforcement learning (RL) approach rely on intrinsic motivation by providing the agent with intrinsic reward derived from CLIP visual novelty. Despite having a relatively small actor network—only 3.2M trainable parameters (approx. 11 MB) — our system is able to learn the structure of the specific environment in which it is trained. Although agents eventually converge to local minima, they first navigate to regions where novelty can be most easily exploited before repeating a back-and-forth behavior. This suggests that the visual information needed for goal-directed navigation can be captured by a lightweight model, allowing the agent to implicitly represent a navigation strategy without an explicit position representation. More work is needed to understand how broadly this strategy can generalize, but studying such compact models is an important step toward efficient embodied agents.

Building upon RL-based exploration, we additionally develop a modular perception pipeline designed to identify objects of interest and characterize scene composition across the explored environment. To accomplish this, we incorporate lightweight detection and segmentation models with vision-language encoders for finer semantic filtering. This pipeline enables frame-level reasoning for objects within the environment, to ultimately evaluate the ability of the RL model to explore the 3D scene.

The code is publicly available at: https://github.com/viriyadhika/NavAssistant.

II. Related Works

A. Intrinsic Motivation and Reinforcement Learning for Exploration

Modern Vision-Language Action (VLA) systems typically rely on large language-model backbones to guide decision-making [4]. However, for embodied AI applications, a smaller and more specialized model would be more suitable for real-time control. Classical methods such as brute-force search, wall-following, and A* are also lightweight. However, these algorithms are not optimized for capturing novel or informative visual content and requires position inference. In this work, our aim is to push the limits of what can be achieved using purely visual information without intermediate position inference which makes classical path-finding approaches infeasible.

To solve this, Reinforcement learning (RL) has become a central approach for embodied navigation, particularly in

settings where agents must autonomously acquire information about their surroundings with explicit task instructions [5]. For policy optimization, Proximal Policy Optimization (PPO) remains the dominant optimization algorithm due to its stable clipped-update objective and robustness to noisy sensory inputs [6]. For the RL reward design, we are inspired by intrinsic-motivation methods which incentivize agents to explore novelty and unknown states. Early works such as the Intrinsic Curiosity Module (ICM) and Random Network Distillation (RND) demonstrate that agents can explore unseen regions by maximizing prediction error or feature-space discrepancy [7] [8].

B. Object Detection and Segmentation Models

Modern visual perception pipelines commonly rely on general-purpose object detectors and segmentation networks capable of operating across diverse indoor scenes. In recent years, You Only Look Once (YOLO) [9] models—particularly YOLOv8—have advanced object detection by combining a deep CNN backbone with a single-stage prediction head, enabling fast and lightweight detection suitable for processing long video sequences. However, detectors alone are often insufficient for tasks requiring semantic discrimination beyond bounding boxes, such as distinguishing visually similar categories or ensuring cross-video consistency in cluttered environments. To incorporate this richer level of understanding, vision–language models (VLMs) like CLIP and BLIP [10] have been utilized. BLIP offers a strong image–text matching and caption-grounded reasoning, making it effective for validating detections or enhancing object-level embeddings.

C. Indoor Scene Understanding and Simulation Environment

Computer vision systems typically rely on learned embeddings to interpret visual scenes, yet these embeddings vary substantially in the type of information they encode. For instance, ResNet-18 produces generic visual features useful for recognition tasks, Contrastive Language-Image Pre-training (CLIP) [11] provides language-aligned semantic representations, and VGGT [12] captures the 3D geometric structure of a scene. In our work, we experiment with these embeddings to shape the reward function and drive the model architecture.

More recent approaches incorporate vision–language models, using models like CLIP to define intrinsic rewards based on perceptual change, where external rewards may be sparse. CLIP-based intrinsic rewards typically defines novelty as the norm difference between the current frame and the immediately preceding one [13]. While this approach works reasonably well when paired with extrinsic task incentives, it becomes problematic in purely intrinsic motivation settings: the agent is encouraged to maximize short-term perceptual change, which in 3D environment, rotating in place always yields higher local novelty than moving [14].

To study vision-based memory and retrieval, we conduct experiments in the ProcTHOR [15] simulation platform, which offers a Python interface to a Unity-based 3D environment. ProcTHOR is chosen for its realistic egocentric RGB observations. Unless otherwise specified, experiments are performed in environment 3, with environment 8084 used for additional evaluation.

III. Methodology

A. RL Algorithm

In our setup, the state space consists solely of the agent's first-person visual frame, and the action space contains three discrete movements: MoveRight (30° rotation), MoveLeft (30° rotation), and MoveAhead (0.25 meters). An episode consists of 1024 frames, starting from a random place on the map. A roll-out consists of 4 episodes, followed by 40 gradient descend steps with AdamW optimizer.

To promote exploration, we created novel a reward functions with 3 components. The rewards is defined as follow:

1) With $\mathcal{B}$ as buffer of embeddings of all past frames, the cosine similarities between the new CLIP embedding and previous ones:

$$s_i = \mathbf{e}_t^\top \mathbf{e}_i, \qquad \mathbf{e}_i \in \mathcal{B}.$$

Get the top k similar frames:

$$\bar{s}_t^{(k)} = \frac{1}{k} \sum_{i \in \text{Top-}k(s_1,\ldots,s_{|\mathcal{B}|})} s_i.$$

The raw novelty reward is

$$r_t^{\text{raw}} = 1 - \bar{s}_t^{(k)}.$$

An exponential moving average (EMA) baseline is maintained:

$$b_t = \tau b_{t-1} + (1-\tau)\, r_t^{\text{raw}}.$$

The intrinsic reward is the deviation from this baseline:

$$r_t^{intrinsic} = r_t^{\text{raw}} - b_t.$$

2) Movement bonus: With $\mathcal{P}_t$ is a buffer of the last 16 positions: Define the mean recent position:

$$\bar{\mathbf{p}}_t = \frac{1}{|\mathcal{P}_t|} \sum_{\mathbf{p}_i \in \mathcal{P}_t} \mathbf{p}_i.$$

A small movement-based bonus encourages deviation from the recent average location:

$$r_t^{\text{move}} = \frac{1}{2} \left\| \mathbf{p}_t - \bar{\mathbf{p}}_t \right\|.$$

3) $r_t^{fail\_penalty}$ Fail penalty when the agent bumped into wall, extracted from ProcTHOR environment.

Final reward is $r_t = r_t^{intrinsic} + r_t^{move} + r_t^{fail\_penalty}$

The policy and value networks share a frozen ResNet-18 encoder, followed by a fixed PCA transform. Two separate Sliding Window Transformers then process the most recent 32 visual embeddings: one Transformer outputs action logits (actor), and the other produces value estimates (critic). The

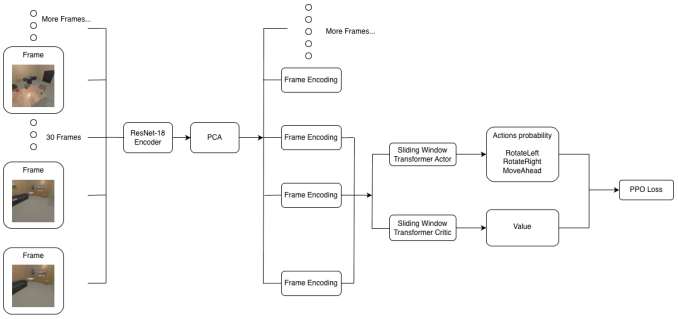

Fig. 1. Model Architecture

overall PPO loss consists of three components: clipped surrogate policy loss, value-function regression loss, and entropy bonus to encourage exploration. The schematic is shown by Fig. 1

These design choices were made based on observations from existing research setups and preliminary experiments

1) Temporal context (32-frame window).
   Intuitively, action selection should depend not only on the current observation but also on the agent's recent trajectory. Prior work using ICM and RND typically models this temporal context with LSTMs, but in our setting LSTMs tended to stay static and output a nearly uniform action predictions—likely reflecting vanishing-gradient issues over long sequences. Motivated by this, we replace the recurrent architecture with a 4-layer Transformer for temporal modeling.
   Because full-sequence attention incurs quadratic memory cost, we restrict the model to attend only to the most recent 32 frames. We explore two designs for doing so: a Local Attention Transformer and a Sliding Window Transformer. Conditioning both the policy and value functions on recent visual history facilitates short-horizon planning.
2) CLIP novelty over RND.
   We initially experimented with Random Network Distillation (RND), but found that by design, the RND prediction error decreases monotonically over time, irrespective of the agent's actual behavioral novelty. In 3D environments, the steady decline of the RND reward can overshadow subtle action-dependent differences, limiting its usefulness for guiding exploration. Moreover, the intrinsic reward is difficult to interpret: it is not obvious how much a human observer should assign a particular RND reward value to a given image. In contrast, novelty measured in the CLIP embedding space yields a more stable and semantically meaningful intrinsic reward, improving both interpretability and training stability.
3) CLIP over 3D-aware VGGT embeddings.
   We also tested novelty signals derived from 3D-aware VGGT embeddings. However, even visually unrelated simulated scenes produced VGGT Embedding similarity scores above 0.99, suggesting that the reconstruction-driven embedding is unable to differentiate different scenes from the same simulation. This makes VGGT unsuitable for novelty-driven exploration in our setting, whereas CLIP retains much better semantic discrimination as shown in Fig. 2

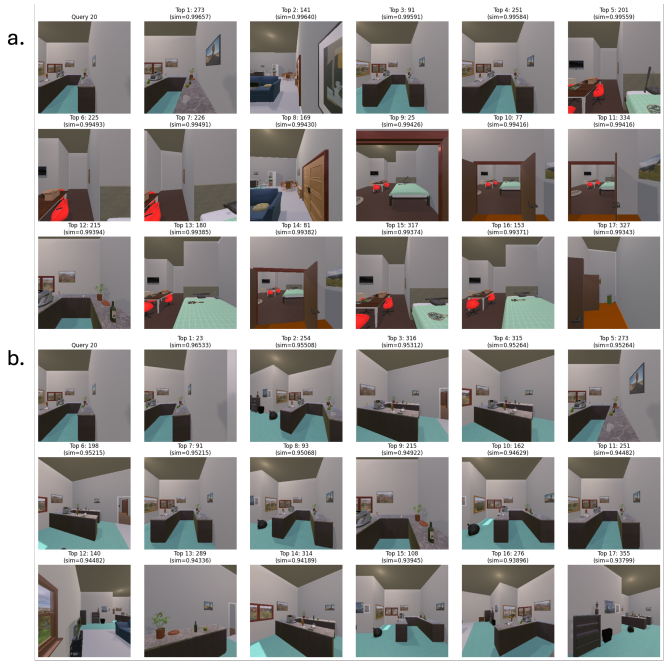

Fig. 2. A fixed dataset of 200 scenes is used. For a randomly selected query image (index 20), the Top-17 most similar embeddings are shown for: (a) VGGT and (b) CLIP. CLIP's similarity rankings align more closely with human intuition about visual novelty.

B. Segmentation and Storage

Following the RL-based exploration stage, we develop a modular perception pipeline designed to identify objects of interest and extract semantic structure from the RL-exploration. The pipeline operates offline and processes the entire exploration of egocentric RCG frames produced by the RL agent.

1) Object Detection. We employ YOLOv8 as the primary detection model due to its speed and robustness in cluttered indoor scenes. Vision transformers such as, Segment Anything Model, were considered but these models require high compute. Therefore, for a compact embodied AI applications it would be unreasonable to use. YOLO runs on a per-frame basis and produces initial bounding boxes for all class objects in its model. Although YOLO provides high recall, it often produces over-broad or noisy bounding boxes, there we treat its detections as candidate detections.
2) Vision-Language Filtering. To refine these detections, we incorporate BLIP-based image-to-text matching. For each YOLO candidate, a cropped image region is passed to BLIP along with a text query specifying the target category. The BLIP model is used to filter ambiguous YOLO detections to improve precision and enables the

pipeline to recover object instances that are otherwise difficult to classify solely from bounding boxes.

3) Semantic Embedding Storage. For each validated detection, we store both the YOLO bounding box and the corresponding BLIP embedding. These embeddings provide a richer feature representation suitable for further downstream retrieval or indexing using FAISS vector database.

IV. Results and Discussion

To trace improvement in training performance of different algorithms, we typically use the average of reward from a single episode, with a higher and consistent episodic reward corresponding to a better policy. In this section, we outline 4 experiments that led to the final design in the methodology section.

A. Local Attention Transformer

We first evaluate a Local Attention Transformer to model temporal dependencies within the observation sequence. The model operates on inputs of shape (B, S, D) [1] and uses a windowed attention mask that allows each query to attend only to the most recent W tokens. We then compare two encoders: a CNN trained from scratch, and a frozen pretrained ResNet-18 whose 512-dimensional features are projected to a 256-dimensional embedding via a learnable linear layer.

Fig. 3 shows the resulting unit-less episodic reward curves. Several characteristics are evident:

- The episodic reward increases over time but remains highly noisy, with noise growing as training progresses. This is expected because each episode initializes the agent at a random location in the environment. Episodes starting in rarely visited regions—or locations where the agent becomes stuck (e.g., near corners)—tend to yield significantly lower reward.
- Throughout training, the pretrained ResNet-backbone encoder converges to essentially the same solution as the CNN trained from scratch but does so significantly faster—highlighting the advantage of starting from pretrained visual features.

B. Sliding Window Transformer

One limitation of the Local Attention Transformer is that it must repeatedly relearn positional relationships whenever similar visual patterns appear at different points in the episode. For instance, if the agent encounters the same scene sequence near the beginning and again in the middle of an episode, the model treats these as distinct contexts because their absolute positions differ. This reduces sample efficiency and places unnecessary burden on the positional encoding mechanism.

In contrast, a Sliding Window Transformer avoids this limitation by forcing the model to attend only over a fixed-length window of recent observations, achieved by reshaping the input from (B, S, D) to $(B \times S, W, D)$. However, this comes with a significant computational burden: the sliding-window approach scales as $O(SW^2D)$, whereas a Local Attention Transformer attains a more efficient $O(SWD)$ complexity.

Since the Sliding Window Transformer is computationally expensive, it requires adjustments to the visual encoder to extract as much performance as possible from the GPU. We therefore experimented with two modifications to the Local Attention Transformer encoders:

- We replaced the learnable projection layer that followed the ResNet-18 encoder with a *fixed random* linear projection as the GPU memory is not enough for backpropagation to the encoder. Surprisingly, both the Local Attention and Sliding Window Transformers still converged to reasonable local optima.
- Although the random projection lacks strong theoretical

[1] B: the number of episodes in a roll-out (batch size), S: the number of steps in an episode, D: the transformer embedding dimension, and W: the attention window.

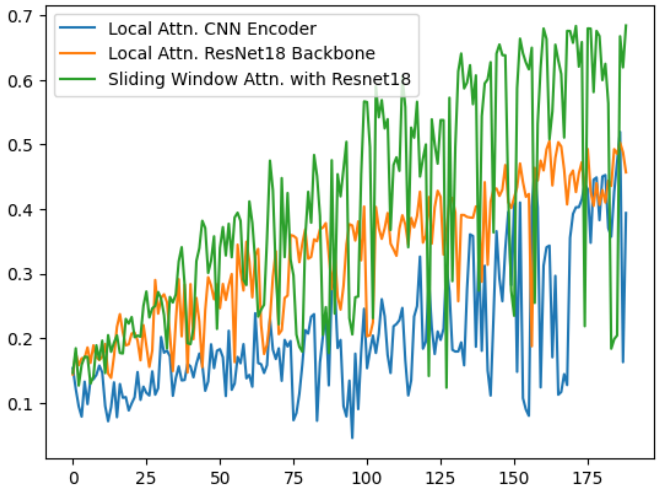

Fig. 3. Reward curve - CNN ResNet encoders

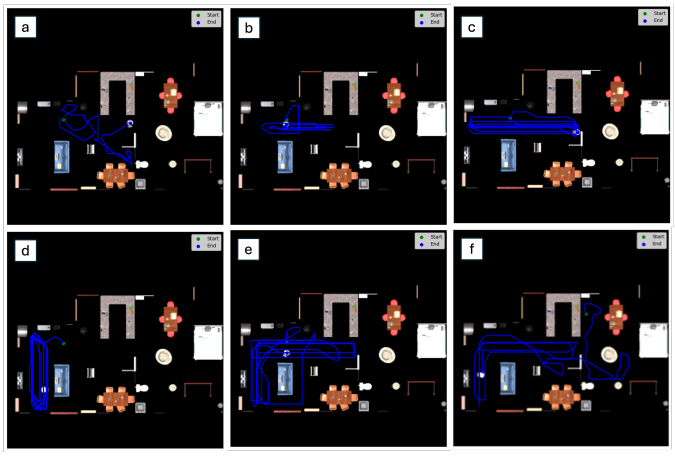

Fig. 4. Trajectory of different models a.) Random policy b.) Local Attention for ResNet18 backbone, *Raw CNN converge to same behavior c.) Sliding-window Transformer with frozen ResNet-18 with a random linear layer d.) Same as c. with no CLIP Novelty curiosity e.) ResNet-18 with a PCA projection f.) Same model as e. but starting from the bedroom

grounding in this setting, we also explored a PCA-based projection that reduces the 512-dimensional ResNet features to 256. To obtain the PCA projection matrix, we collect a buffer of images using a random policy before training. With this modification, the model can find higher episodic reward solution as shown by Fig. 3

C. Video and Training Qualitative Analysis

After training, we keep only the actor to generate a set of inference trajectories, examining the exploration behaviors it learns under different configurations. Across these settings, each model converges to a distinct local optimum driven by its inductive biases and the reward design.

Fig. 4a shows a baseline trajectory produced by a random policy with action probabilities $[MoveAhead, RotateLeft, RotateRight] = [0.5, 0.25, 0.25]$. Fig. 4b illustrates the behavior under Local Attention. Fig. 4c presents the sliding-window Transformer using a frozen ResNet-18 encoder and a fixed random projection. Its behavior resembles that of (b) but exhibits longer movements. On the other hand, ResNet-18 encoder with PCA head trained policy followed an L-shaped path that explores different side of the living room as shown in Fig. 4e. This indicates that curiosity encourages taking longer, more exploratory paths that lead to novel observations. However, it also discourages risky actions that might cause the agent to get trapped, such as entering the doorway. This policy also develops a mechanism to always reach local minima from any point in the map. For example, when initialized in a bedroom corner facing a wall, the agent exits the room and moves toward the hallway, as shown in Fig. 4f. As an ablation study, when curiosity is removed from this model, we observe yet another exploration pattern (Fig. 4d): the agent oscillates primarily along the y-axis rather than the previous L-shape pattern.

Full egocentric videos corresponding to these trajectories (including trajectory training on a new environment number 8084 for generalization) are available at: https://viriyadhika.com/projects/navassistant. [2]

D. Evaluation on Downstream Tasks

Finally, we evaluated all model variants on an object-diversity downstream tasks to access the perception richness of the learned exploration policy. For Fig. 5, we aim to measure the novelty of unique objects detected by the encoder variations to determine the exploration path novelty for each scenario. Each segmentation for unique objects in the environment is stored and summed to create another method to evaluate the novelty of the exploration.

For Fig. 6, we aim to evaluate the frame coverage of a particular object segmentation as a means to determine the proximity to objects found in the environment. However, as the segmentation from YOLO is unable to discriminate between duplicate objects in the environment, this method of evaluation poses challenges. For example,Fig. 4e with PCA shows thorough exploration compared to Fig. 4d, but Fig. 6 shows the opposite when accounting for all scenarios. However, we can use both methods of evaluations to identify that models using the sliding-window transformer consistently detect more unique objects and explore objects more thoroughly compared to the local-attention model. (Fig. 5, (Fig. 6). With the PCA-projected model achieving the strongest performance for identifying more unique objects.

[2]Since actions are stochastic, we select representative runs that best reflect the learned policies.

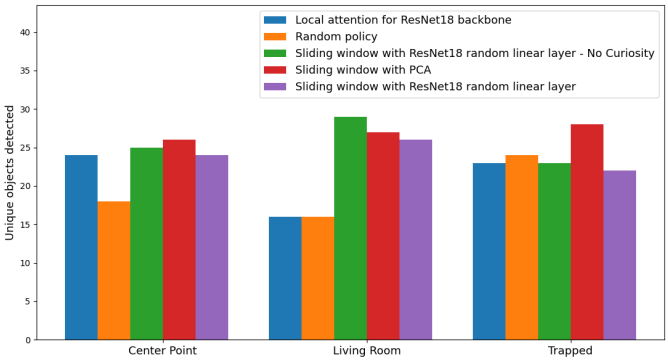

Fig. 5. Unique object detections for 5 model variations across three evaluation environments.

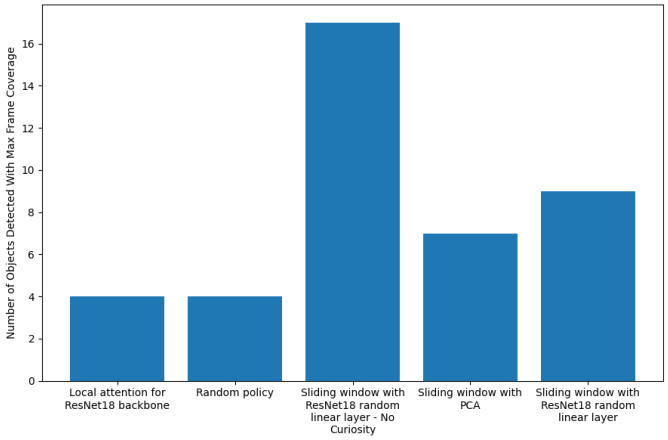

Fig. 6. Number of maximal frame coverages produced by each of the 5 model variations across three evaluation environments. Maximal frame coverage is defined by which model produces a segmentation of an object with the maximum percent of the frame covered compared to other models with same object detection.

V. Conclusion

This work examines how different visual encoders and Transformer architectures behave in a setting solely to environment exploration, without any explicit downstream objective. Across these configurations, the agents consistently display periodic behavior. Our findings show that architectural choices introduce distinct inductive biases that shape this behavior. A deeper analysis of the reward landscape is needed to fully understand these differences. Notably, the architectures we study converge to a stable and coherent strategy after

roughly 50 update steps, whereas many closely related architectures—with alternative neural inductive biases or slightly modified reward structures—fail to converge and remain noisy even after 100 updates.

We also introduce a simple evaluation method for exploration tasks based on object counting using an off-the-shelf YOLO-v8 model. Although these results do not directly generalize to broader navigation benchmarks, they suggest that compact models can still exhibit emergent, navigation-like behaviors in visually rich environments.

VI. Individual Contribution

Name	Contribution
Viriyadhika	Simulation Environment setup (ProcTHOR)
	RL Literature Review
	RL Preliminary Research Implementation
	RL Final Transformer design and training
	Inference scenarios for downstream eval
Darpal	Segmentation and FAISS exploration in downstream eval
	Segmentation Literature Review

References

[1] Microsoft, "Seeing ai: An app for visually impaired people that narrates the world around you," https://www.microsoft.com/en-us/garage/wall-of-fame/seeing-ai/, accessed: 2025-12-01.

[2] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *CoRR*, vol. abs/2007.00643, 2020. [Online]. Available: https://arxiv.org/abs/2007.00643

[3] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," 2023. [Online]. Available: https://arxiv.org/abs/2206.12403

[4] Z. Qi, Z. Zhang, Y. Yu, J. Wang, and H. Zhao, "Vln-r1: Vision-language navigation via reinforcement fine-tuning," 2025. [Online]. Available: https://arxiv.org/abs/2506.17221

[5] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357–3364.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[7] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," 2017. [Online]. Available: https://arxiv.org/abs/1705.05363

[8] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CVPR*, 2016.

[10] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," 2022. [Online]. Available: https://arxiv.org/abs/2201.12086

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[12] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," 2025. [Online]. Available: https://arxiv.org/abs/2503.11651

[13] T. Gupta, P. Karkus, T. Che, D. Xu, and M. Pavone, "Foundation models for semantic novelty in reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2211.04878

[14] C. Wu, W. Yu, W. Liao, and Y. Ou, "Deep reinforcement learning with intrinsic curiosity module based trajectory tracking control for USV," *Ocean Engineering*, vol. 308, 2024.

[15] E. Kolve, R. Mottaghi, D. Han, and et al., "Ai2-thor: An interactive 3d environment for visual ai," in *Proceedings of the 2017 Conference on Robot Learning (CoRL)*, 2017.