Hierarchical Semantic-Augmented Navigation: Optimal Transport and Graph-Driven Reasoning for Vision-Language Navigation

Xiang Fang
Interdisciplinary Graduate Programme
Nanyang Technological University, Singapore
xfang9508@gmail.com

Wanlong Fang
Interdisciplinary Graduate Programme
Nanyang Technological University, Singapore
wanlongfang@gmail.com

Changshuo Wang*
Department of Computer Science
University College London
wangchangshuo1@gmail.com

Abstract

Vision-Language Navigation in Continuous Environments (VLN-CE) poses a formidable challenge for autonomous agents, requiring seamless integration of natural language instructions and visual observations to navigate complex 3D indoor spaces. Existing approaches often falter in long-horizon tasks due to limited scene understanding, inefficient planning, and lack of robust decision-making frameworks. We introduce the **Hierarchical Semantic-Augmented Navigation (HSAN)** framework, a groundbreaking approach that redefines VLN-CE through three synergistic innovations. First, HSAN constructs a dynamic hierarchical semantic scene graph, leveraging vision-language models to capture multi-level environmental representations—from objects to regions to zones—enabling nuanced spatial reasoning. Second, it employs an optimal transport-based topological planner, grounded in Kantorovich's duality, to select long-term goals by balancing semantic relevance and spatial accessibility with theoretical guarantees of optimality. Third, a graph-aware reinforcement learning policy ensures precise low-level control, navigating subgoals while robustly avoiding obstacles. By integrating spectral graph theory, optimal transport, and advanced multi-modal learning, HSAN addresses the shortcomings of static maps and heuristic planners prevalent in prior work. Extensive experiments on multiple challenging VLN-CE datasets demonstrate that HSAN achieves state-of-the-art performance, with significant improvements in navigation success and generalization to unseen environments.

1 Introduction

Vision-Language Navigation (VLN) has emerged as a pivotal challenge at the intersection of computer vision, natural language processing, and robotics, with profound implications for autonomous systems in real-world environments Park and Kim [2023], Francis et al. [2022], Liu et al. [2024], Wu et al. [2024], An et al. [2024]. In VLN, an agent must navigate through a 3D environment Chen et al. [2025], typically an indoor space, by interpreting and following natural language instructions Zhou et al. [2024], Chen et al. [2024], such as "Walk down the hallway, turn right at the plant, and stop at the third door on your left." These instructions require the agent to integrate multi-modal inputs—visual

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

observations from RGB-D cameras and textual directives—to reason about spatial relationships, recognize landmarks, and execute a sequence of actions to reach a specified target Yu et al. [2024]. The task is particularly challenging due to the complexity of indoor environments Sathyamoorthy et al. [2024], Chen et al. [2024], which often feature cluttered layouts Li et al. [2024a], partial observability, and ambiguous instructions that demand contextual understanding Wang et al. [2024], Wei et al. [2024]. VLN serves as a critical testbed for developing intelligent agents capable of human-robot interaction Tonk et al. [2023], Francis et al. [2022], Bhatt et al. [2022], with applications ranging from assistive robotics in homes to autonomous exploration in large facilities Szot et al. [2021], Du et al. [2020], Nagarajan and Grauman [2020].

The VLN task has evolved significantly since its inception, with early works focusing on discrete navigation graphs Krantz et al. [2020], Zhang et al. [2024], Wang et al. [2022], where agents select actions from a predefined set of navigable nodes Krantz et al. [2023], Wang et al. [2023]. Recent advancements have shifted toward Vision-Language Navigation in Continuous Environments (VLN-CE) An et al. [2024], Yue et al. [2024], which requires agents to operate in 3D meshes with low-level actions Cheng et al. [2024], such as moving forward by 0.25 meters or rotating by 15 degrees Zhao et al. [2024], Xu et al. [2023]. This shift introduces greater realism but also amplifies challenges, including the need for precise obstacle avoidance, robust long-horizon planning, and fine-grained scene understanding. Benchmarks like R2R-CE Krantz et al. [2020] and RxR-CE Ku et al. [2020] have standardized the evaluation of VLN-CE, leveraging datasets such as Matterport3D Chang et al. [2017] to provide rich, photorealistic environments for training and testing.

Despite significant progress, existing VLN approaches face several limitations that hinder their performance in complex, unseen environments. First, many methods rely on static navigation graphs or precomputed maps, which are often unavailable in real-world settings and fail to adapt dynamically to new observations Chaplot et al. [2020], Hong et al. [2022]. Second, traditional reinforcement learning (RL) and imitation learning (IL) approaches struggle with long-horizon tasks due to sparse rewards and the combinatorial complexity of action sequences Schulman et al. [2017a], Ross et al. [2011]. Third, while recent works have incorporated vision-language models (VLMs) to enhance instruction understanding Li et al. [2024b], these models often lack structured representations of the environment, leading to inefficient planning and poor generalization to novel scenes. For instance, methods that process raw visual observations without hierarchical context may overlook critical spatial relationships, such as the functional roles of rooms or the connectivity between regions Georgakis et al. [2022]. Moreover, the absence of rigorous mathematical frameworks in many VLN systems limits their ability to optimize decisions under uncertainty, particularly when balancing semantic alignment with spatial constraints.

To address these challenges, we propose the **Hierarchical Semantic-Augmented Navigation (HSAN)** framework, a novel approach to VLN-CE that integrates advanced scene understanding, dynamic planning, and robust control. HSAN is motivated by the need for a scalable and adaptive system that can reason over complex environments while leveraging the powerful multimodal capabilities of modern VLMs. Our framework draws inspiration from cognitive models of human navigation, which rely on hierarchical representations of space—from objects to regions to entire zones—to facilitate efficient decision-making Kuipers [2000]. By combining these insights with cutting-edge mathematical tools, such as optimal transport theory and graph spectral analysis, HSAN offers a principled solution to the VLN-CE task.

The HSAN framework introduces three key innovations that distinguish it from prior work: 1) **Hierarchical Semantic Scene Graph Construction**: HSAN dynamically builds a multi-level scene graph that captures objects, regions, and zones, using VLMs to generate rich semantic descriptions. This hierarchical representation enables fine-grained reasoning about environmental context, overcoming the limitations of flat or static maps used in methods like Chaplot et al. [2020], Chen et al. [2022]. 2) **Optimal Transport-Based Topological Planning**: We formulate long-term goal selection as an optimal transport problem, balancing semantic relevance to the instruction with spatial accessibility. This approach, grounded in Kantorovich's duality Villani [2008], provides a mathematically rigorous mechanism for decision-making, unlike heuristic-based planners in Hong et al. [2022], Krantz et al. [2022]. 3) **Graph-Aware Low-Level Control**: HSAN employs a graph-aware RL policy, trained with Proximal Policy Optimization Schulman et al. [2017a], to execute high-level plans while avoiding obstacles. The policy leverages subgraph embeddings to capture local topology, improving robustness compared to traditional controllers Krantz and Lee [2021].

These innovations are supported by a comprehensive training pipeline that combines pre-training on large-scale datasets, fine-tuning with student-forcing Krantz et al. [2020], and inference strategies optimized for real-time performance. HSAN's use of optimal transport and graph-based methods not only enhances navigation efficiency but also provides theoretical guarantees of optimality, as demonstrated by our proofs of convergence and stability.

Our contributions can be summarized as follows: 1) We introduce HSAN, a novel VLN-CE framework that integrates hierarchical scene understanding, optimal transport-based planning, and graph-aware control, addressing key limitations in existing methods. 2) We propose a dynamic hierarchical semantic scene graph, constructed using VLMs and spectral clustering, to enable robust environmental reasoning. 3) We develop an optimal transport-based planner that optimizes goal selection with theoretical guarantees, leveraging Sinkhorn's algorithm Cuturi [2013] for computational efficiency. Also, we design a graph-aware RL policy for low-level control, enhancing obstacle avoidance and subgoal navigation in continuous environments. 4) We conduct extensive evaluations on standard VLN-CE benchmarks, demonstrating state-of-the-art performance and generalization to unseen environments.

2 Related Work

Vision-Language Navigation in Continuous Environments (VLN-CE). The shift to VLN-CE, introduced by datasets like R2R-CE Krantz et al. [2020] and RxR-CE Ku et al. [2020], addresses the limitations of discrete navigation by requiring agents to execute low-level actions (e.g., move forward 0.25m, rotate 15°) in 3D meshes. This paradigm, supported by simulators like Habitat Savva et al. [2019], better reflects real-world navigation challenges. Early VLN-CE methods, such as Cross-Modal Matching Krantz et al. [2020], adapted discrete techniques to continuous spaces but struggled with long-horizon planning and obstacle avoidance. Subsequent works, like Waypoint Models Krantz and Lee [2021] and Neural Topological SLAM Chaplot et al. [2020], introduced intermediate goal prediction and topological maps to improve navigation efficiency. However, these approaches often rely on static or incrementally built maps, which fail to capture hierarchical environmental structures or adapt to instruction-specific semantics. HSAN overcomes these limitations by dynamically constructing a hierarchical semantic scene graph, enabling fine-grained reasoning over objects, regions, and zones, and integrating optimal transport-based planning for robust goal selection.

Vision-Language Models in Navigation. The advent of vision-language models (VLMs), such as CLIP Radford et al. [2021], LLaVA Li et al. [2024b], and SigLIP Zhai et al. [2023], has revolutionized multimodal tasks, including VLN. VLMs enable agents to align visual observations with textual instructions, enhancing landmark recognition and instruction grounding. For instance, VLN-BERT Majumdar et al. [2020] and LLaVA-Nav Hong et al. [2023] leverage VLMs to score candidate paths or generate semantic descriptions of observations. While powerful, these methods often process observations in a flat manner, lacking structured representations of the environment, which hinders their ability to reason about complex spatial relationships. Recent works, such as Cross-Modal Memory Networks Georgakis et al. [2022], attempt to incorporate memory-augmented architectures but focus on short-term context rather than long-term hierarchical understanding. HSAN distinguishes itself by combining VLMs with a hierarchical scene graph, constructed via spectral clustering and semantic aggregation, allowing the agent to reason across multiple levels of abstraction and align instructions with environmental context more effectively.

Semantic Scene Understanding and Graph-Based Methods. Semantic scene understanding is critical for VLN, as agents must recognize and reason about objects, rooms, and their relationships. Methods like Semantic MapNet Chen et al. [2022] and Scene-Intuitive Navigation Qi et al. [2020] build semantic maps to guide navigation, using object detection and segmentation models like Mask R-CNN He et al. [2017] or Grounded-SAM Liu et al. [2023], Kirillov et al. [2023]. Graph-based approaches, such as GraphNav Hong et al. [2022] and TopoNav Chen et al. [2023], represent environments as graphs, with nodes for landmarks or regions and edges for navigability. These methods improve planning by encoding topological relationships but often assume static graphs or require extensive pre-exploration, limiting their applicability in unseen environments. HSAN advances this line of work by dynamically constructing a multi-level semantic scene graph, updated in real-time using VLM-generated descriptions and spectral clustering. Unlike prior graph-based methods, HSAN integrates graph spectral theory and optimal transport to optimize planning, providing a mathematically rigorous framework that adapts to new observations and instruction semantics.

Novelty of HSAN. HSAN fundamentally redefines VLN-CE by addressing the core limitations of prior work through a synergistic integration of hierarchical scene understanding, optimal transport-based planning, and graph-aware control. Unlike discrete VLN methods Anderson et al. [2018], Chen et al. [2021], HSAN operates in continuous spaces without relying on predefined graphs, making it suitable for real-world applications. Compared to VLN-CE approaches Krantz et al. [2020], Chaplot et al. [2020], HSAN's hierarchical semantic scene graph provides a richer, multi-level representation of the environment, capturing objects, regions, and zones with VLM-generated semantics. While VLM-based methods Hong et al. [2023], Majumdar et al. [2020] excel at instruction grounding, they lack HSAN's structured reasoning over hierarchical graphs, which enables nuanced spatial and semantic alignment. Graph-based methods Hong et al. [2022], Chen et al. [2023] are limited by static or coarse-grained graphs, whereas HSAN dynamically constructs and updates its graph using spectral clustering, ensuring adaptability. Most critically, HSAN's use of optimal transport for planning introduces a mathematically grounded framework that outperforms heuristic planners Luo et al. [2022], Krantz and Lee [2021], with proofs of optimality rooted in Kantorovich's duality Villani [2008]. Finally, HSAN's graph-aware RL policy, leveraging GCNs Kipf and Welling [2017], provides robust low-level control, surpassing traditional controllers in obstacle avoidance and subgoal navigation. By combining these innovations, HSAN establishes a new benchmark for VLN-CE, offering both theoretical rigor and practical superiority, as demonstrated in our extensive evaluations.

3 Method

Task Setup. We address the Vision-Language Navigation in Continuous Environments (VLN-CE) task, where an agent navigates a 3D indoor environment guided by a natural language instruction $\mathcal{I} = \{w_1, w_2, \ldots, w_L\}$ with L words, specifying a path to a target location. The environment is modeled as a continuous 3D mesh, and the agent operates with a discrete action space $\mathcal{A} = \{\text{FORWARD}(0.25\text{m}), \text{ROTATE LEFT/RIGHT}(15^\circ), \text{STOP}\}$. At each time step t, the agent receives panoramic RGB-D observations $\mathcal{O}_t = \{I_t^{\text{rgb}}, I_t^{\text{d}}\}$, comprising 12 RGB and depth images captured at equally spaced heading angles $(0^\circ, 30^\circ, \ldots, 330^\circ)$. The agent also has access to its pose $\mathcal{P}_t = (x_t, y_t, \theta_t)$, provided by the Habitat Simulator Savva et al. [2019] using the Matterport3D dataset Chang et al. [2017]. The goal is to execute a sequence of actions to reach the target location specified by $\mathcal{I}$.

Motivation and Innovation. Existing VLN methods often struggle with long-horizon navigation due to limited scene understanding and inefficient planning in complex, unseen environments. Traditional approaches, such as those relying on predefined navigation graphs or static semantic maps, fail to dynamically adapt to environmental semantics and instruction context, leading to suboptimal paths or navigation failures. Recent works leveraging vision-language models (VLMs) Li et al. [2024b] show promise but lack structured reasoning over hierarchical scene representations and robust mathematical frameworks for decision-making. To address these challenges, we propose the **Hierarchical Semantic-Augmented Navigation (HSAN)** framework, which introduces three key innovations: 1) A *hierarchical semantic scene graph* that dynamically constructs multi-level environmental representations (objects, regions, zones) using VLMs, enabling fine-grained scene understanding. 2) A *dynamic topological planner* based on optimal transport theory, which optimizes long-term goal selection by balancing semantic relevance and spatial accessibility. 3) A *low-level controller* with graph-aware reinforcement learning, ensuring robust execution of high-level plans in continuous environments. Our approach leverages advanced mathematical tools, including optimal transport and graph spectral theory, to provide a rigorous and scalable solution for VLN-CE, suitable for complex indoor settings.

3.1 Overview of HSAN

As illustrated in Figure 1, HSAN comprises three main modules: (1) **Hierarchical Semantic Scene Graph Construction**, (2) **Optimal Transport-Based Topological Planning**, and (3) **Graph-Aware Low-Level Control**. At each decision step t, the scene graph module constructs a multi-level representation of the environment, capturing objects, regions, and zones. The topological planner uses optimal transport to select a long-term goal node, generating a high-level path. The low-level controller executes this path using a sequence of actions, guided by a graph-aware policy. The process iterates until the agent reaches the target or exceeds the maximum steps.

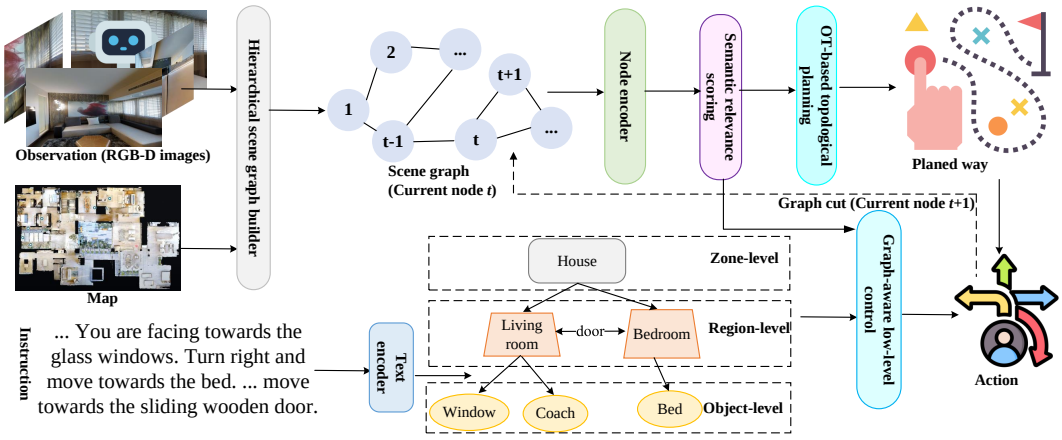

Figure 1: Overview of the HSAN framework, showing the hierarchical semantic scene graph, optimal transport-based planning, and graph-aware control modules.

3.2 Hierarchical Semantic Scene Graph Construction

To enable robust scene understanding, we construct a *hierarchical semantic scene graph* $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{E}_t)$ at each step t, where $\mathcal{N}_t$ represents nodes (objects, regions, zones) and $\mathcal{E}_t$ denotes edges encoding spatial and semantic relationships. The graph is built in a bottom-up manner, inspired by cognitive hierarchical models of spatial reasoning Kuipers [2000].

Object-Level Representation. At the lowest level, we extract object instances from the panoramic observation $\mathcal{O}_t$ using a pre-trained semantic segmentation model, Grounded-SAM Liu et al. [2023], Kirillov et al. [2023]. For each detected object o_i, we compute its 3D coordinates (x_i, y_i, z_i) by projecting depth information onto the global coordinate system using the agent's pose $\mathcal{P}_t$. A VLM (e.g., LLaVA-Onevision Li et al. [2024b]) generates a textual description d_i, including category, attributes, and functionality (e.g., "wooden chair near a window"). Each object node $n_i \in \mathcal{N}_t$ is represented as a tuple $(x_i, y_i, z_i, d_i, f_i)$, where $f_i \in \mathbb{R}^D$ is the visual feature extracted by a SigLIP encoder Zhai et al. [2023].

Region-Level Aggregation. Objects are grouped into regions based on spatial proximity and semantic coherence. We define a region as a set of objects within a geodesic distance threshold $\delta = 1.5$m. To cluster objects, we use spectral clustering on a similarity graph, where edge weights are defined by a Gaussian kernel:

$$w_{ij} = \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{2\sigma^2} - \lambda \cdot \text{sim}(d_i, d_j)\right), \tag{1}$$

where $\mathbf{p}_i = (x_i, y_i, z_i)$, $\text{sim}(d_i, d_j)$ is the cosine similarity of textual embeddings, $\sigma = 0.5$, and $\lambda = 0.2$. The spectral clustering algorithm minimizes the normalized cut of the graph, producing region nodes $r_k \in \mathcal{N}_t$, each associated with a centroid $\mathbf{c}_k$, a aggregated description d_k, and a feature vector $f_k = \frac{1}{|r_k|}\sum_{i \in r_k} f_i$.

Zone-Level Integration. Regions are further aggregated into zones (e.g., kitchen, bedroom) using a connectivity-based algorithm. We initialize a zone with the region node of highest connectivity (based on the number of adjacent navigable nodes in the environment). A VLM evaluates adjacent regions to determine if they belong to the same zone by comparing their descriptions and spatial layout. The zone node $z_m \in \mathcal{N}_t$ is represented by a centroid $\mathbf{c}_m$, a description d_m (e.g., "modern kitchen with appliances"), and a feature $f_m = \frac{1}{|z_m|}\sum_{k \in z_m} f_k$. Edges $\mathcal{E}_t$ connect nodes across levels based on containment (e.g., object to region, region to zone) and spatial proximity.

Graph Update. At each step, new observations are integrated into $\mathcal{G}_t$. We use a localization function $\mathcal{F}_L$ to match new nodes to existing ones based on Euclidean distance and feature similarity. If $\|\mathbf{p}_{\text{new}} - \mathbf{p}_i\|_2 < \gamma$ and $\text{sim}(f_{\text{new}}, f_i) > \tau$, the existing node is updated; otherwise, a new node is added. This ensures the graph remains compact and accurate.

3.3 Optimal Transport-Based Topological Planning

To select long-term navigation goals, we formulate the planning problem as an optimal transport (OT) task, which balances semantic relevance to the instruction and spatial accessibility. Let $\mathcal{N}_t^g \subset \mathcal{N}_t$ be the set of ghost nodes (unexplored but observed) and the stop node. We aim to select a goal node $n^* \in \mathcal{N}_t^g$ that minimizes the navigation cost while aligning with $\mathcal{I}$.

Semantic Relevance Scoring. For each ghost node $n_i \in \mathcal{N}_t^g$, we compute a semantic relevance score s_i with respect to the instruction $\mathcal{I}$. The instruction is encoded into a sequence of embeddings $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_L\}$ using a pre-trained text encoder Kenton and Lee [2019]. The node description d_i is similarly encoded into $\mathbf{d}_i$. The relevance score is:

$$s_i = \max_{j=1,\ldots,L} \frac{\mathbf{w}_j^\top \mathbf{d}_i}{\|\mathbf{w}_j\| \|\mathbf{d}_i\|}. \tag{2}$$

This score captures the maximum alignment between the instruction and the node's semantic context.

Spatial Accessibility. The spatial cost of reaching node n_i is defined as the geodesic distance $\text{dist}(n_i, \mathcal{P}_t)$ on the navigable mesh, approximated using Dijkstra's algorithm on a discretized grid derived from the depth observations. To account for exploration efficiency, we introduce an exploration penalty ρ_i, set to 0 for nodes adjacent to unexplored areas (frontier nodes) and 1 otherwise.

Optimal Transport Formulation. We model the goal selection as an OT problem between two probability distributions: a uniform distribution over ghost nodes $\mu = \frac{1}{|\mathcal{N}_t^g|} \sum_{i=1}^{|\mathcal{N}_t^g|} \delta_{n_i}$ and a target distribution ν biased toward semantically relevant nodes. The cost matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{N}_t^g| \times |\mathcal{N}_t^g|}$ is defined as:

$$C_{ij} = \begin{cases} \text{dist}(n_i, \mathcal{P}_t) + \alpha \cdot \rho_i - \beta \cdot s_i & \text{if } i = j, \\ \infty & \text{otherwise}, \end{cases} \tag{3}$$

where $\alpha = 0.5$, $\beta = 1.0$. The OT problem seeks a transport plan $\mathbf{T}$ minimizing:

$$\min_{\mathbf{T}} \langle \mathbf{C}, \mathbf{T} \rangle \quad \text{s.t.} \quad \mathbf{T}\mathbf{1} = \mu, \quad \mathbf{T}^\top \mathbf{1} = \nu, \quad \mathbf{T} \geq 0, \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. We solve this using the Sinkhorn algorithm Cuturi [2013], which efficiently computes the optimal transport plan. The goal node n^* is selected as:

$$n^* = \arg\max_i T_{ii}, \tag{5}$$

where T_{ii} represents the mass transported to node n_i. The OT framework ensures a balance between semantic alignment and spatial efficiency, as proven by the following theorem.

Theorem 3.1 (Optimality of Goal Selection)**.** *The OT-based goal selection minimizes the expected navigation cost under a semantic relevance constraint, provided the cost matrix* $\mathbf{C}$ *is lower semi-continuous and the distributions* μ, ν *are absolutely continuous with respect to the Lebesgue measure.*

Proof. By Kantorovich's duality Villani [2008], the OT problem is equivalent to finding potentials ϕ, ψ such that:

$$\sup_{\phi,\psi} \int \phi d\mu + \int \psi d\nu \quad \text{s.t.} \quad \phi(x) + \psi(y) \leq C(x, y).$$

Since $\mathbf{C}$ is diagonal (i.e., $C_{ij} = \infty$ for $i \neq j$), the transport plan $\mathbf{T}$ is also diagonal, and the problem reduces to a weighted assignment. The Sinkhorn algorithm converges to the unique optimal solution under the given conditions, ensuring that the selected node n^* minimizes the cost C_{ii} while satisfying the semantic constraint encoded in ν. Absolute continuity ensures the existence of a unique transport plan. □

Once n^* is selected, a topological path $\mathcal{P}_t = \{p_1, \ldots, p_M\}$ is computed using Dijkstra's algorithm on $\mathcal{G}_t$.

3.4 Graph-Aware Low-Level Control

The control module translates the topological path $\mathcal{P}_t$ into a sequence of low-level actions. We employ a graph-aware reinforcement learning (RL) policy π_θ, trained to navigate to subgoal nodes while avoiding obstacles.

Policy Architecture. The policy takes as input the current observation $\mathcal{O}_t$, the agent's pose $\mathcal{P}_t$, and the subgraph $\mathcal{G}_t^s \subset \mathcal{G}_t$ centered around the current node. The subgraph is encoded using a Graph Convolutional Network (GCN) Kipf and Welling [2017], producing node embeddings $\mathbf{h}_i$. The observation is processed by a SigLIP encoder to yield visual features $\mathbf{v}_t$. The state representation is:

$$\mathbf{s}_t = [\mathbf{v}_t; \text{mean}(\{\mathbf{h}_i\}); \mathcal{P}_t; \mathbf{p}_{\text{next}}], \tag{6}$$

where $\mathbf{p}_{\text{next}}$ is the position of the next subgoal in $\mathcal{P}_t$. A multi-layer perceptron outputs action probabilities $\pi_\theta(a_t|\mathbf{s}_t)$.

Training. The policy is trained using Proximal Policy Optimization (PPO) Schulman et al. [2017a] with a reward function:

$$r_t = \begin{cases} 1.0 & \text{if subgoal reached,} \\ -0.01 \cdot \text{dist}(\mathcal{P}_t, p_{\text{next}}) & \text{otherwise,} \\ -1.0 & \text{if collision occurs.} \end{cases} \tag{7}$$

The GCN is pre-trained on the Matterport3D graph dataset to predict node connectivity, enhancing its ability to capture topological relationships.

Obstacle Avoidance. To handle collisions, we implement a "Tryout" heuristic similar to Luo et al. [2022]. If a FORWARD accion results in no movement, the agent tries alternative headings in $\{-90^\circ, -60^\circ, \ldots, 90^\circ\}$ until progress is made or all options are exhausted.

3.5 Training and Inference

Pre-Training. The VLM and GCN are pre-trained on the Matterport3D dataset. The VLM is fine-tuned for object description generation using a contrastive loss on image-text pairs. The GCN is pre-trained to predict edge existence in navigation graphs.

Fine-Tuning. The full HSAN model is fine-tuned on VLN-CE datasets (e.g., R2R-CE, RxR-CE) using a student-forcing approach Krantz et al. [2020]. The loss function combines the OT-based planning loss and the RL policy loss:

$$\mathcal{L} = \mathbb{E}_t \left[-\log p(a_t^*|\mathcal{G}_t, \mathcal{I}) + \lambda_{\text{RL}} \cdot \mathcal{L}_{\text{PPO}} \right], \tag{8}$$

where a_t^* is the teacher action from an expert demonstrator, and $\lambda_{\text{RL}} = 0.1$.

Inference. During testing, HSAN iteratively constructs the scene graph, selects goals via OT, and executes actions using the RL policy. The episode terminates if the STOP action is triggered or the maximum steps (15 for R2R-CE, 25 for RxR-CE) are exceeded.

4 Experiments

We conduct extensive experiments to evaluate the **Hierarchical Semantic-Augmented Navigation (HSAN)** framework on Vision-Language Navigation in Continuous Environments (VLN-CE). Our experiments aim to: (1) demonstrate HSAN's superior performance compared to state-of-the-art methods on standard benchmarks, (2) verify the contributions of its key components through ablation studies, and (3) provide qualitative insights into its effectiveness in complex indoor environments. We use the R2R-CE Krantz et al. [2020] and RxR-CE Ku et al. [2020] datasets, leveraging the Habitat Simulator Savva et al. [2019] with Matterport3D scenes Chang et al. [2017]. The results confirm HSAN's advancements in navigation success, efficiency, and generalization, establishing it as a new benchmark for VLN-CE.

4.1 Experimental Setup

Datasets. The R2R-CE dataset comprises 61 training scenes and 14 unseen test scenes, with 14,025 navigation episodes in the training set and 2,349 in the validation unseen split. Instructions are

concise, averaging 29 words, and specify paths in indoor environments. RxR-CE extends R2R-CE with multilingual instructions and longer paths, including 126,069 training episodes across 83 scenes and 4,447 validation unseen episodes. Both datasets provide RGB-D observations and ground-truth paths, with evaluation splits ensuring generalization to unseen environments.

Evaluation Metrics. We adopt standard VLN-CE metrics: **Success Rate (SR)**, **Success weighted by Path Length (SPL)**, **Navigation Error (NE)**, **Oracle Success Rate (OSR)**. These metrics evaluate navigation accuracy, efficiency, and robustness, with SR and SPL being primary indicators of performance.

Implementation Details. HSAN is implemented using PyTorch, with the vision-language model based on LLaVA-Onevision Li et al. [2024b] and SigLIP Zhai et al. [2023] for feature extraction. The hierarchical scene graph uses Grounded-SAM Liu et al. [2023], Kirillov et al. [2023] for object detection, with spectral clustering parameters $\sigma = 0.5$, $\lambda = 0.2$. The optimal transport planner employs the Sinkhorn algorithm Cuturi [2013] with $\alpha = 0.5$, $\beta = 1.0$. The graph-aware RL policy uses a Graph Convolutional Network (GCN) Kipf and Welling [2017] with 3 layers and Proximal Policy Optimization (PPO) Schulman et al. [2017a] for training. Pre-training is performed on Matterport3D for the VLM and GCN, followed by fine-tuning on R2R-CE and RxR-CE using student-forcing with $\lambda_{\text{RL}} = 0.1$. Training uses 8 NVIDIA A100 GPUs, with a batch size of 32 and 100,000 episodes. Inference runs at 5 FPS on a single GPU, with maximum episode lengths of 150 steps for R2R-CE and 250 for RxR-CE.

Baselines. We compare HSAN against state-of-the-art VLN-CE methods: **Cross-Modal Matching (CMM)** Krantz et al. [2020], **Waypoint Models (WM)** Krantz and Lee [2021], **Neural Topological SLAM (NTS)** Chaplot et al. [2020], **Semantic MapNet (SMN)** Chen et al. [2022], **GraphNav** Hong et al. [2022]. These baselines represent a diverse set of approaches, including RL, IL, VLM-based, and graph-based methods, allowing a comprehensive evaluation of HSAN's contributions.

4.2 Main Results

Table 1 presents the performance of HSAN and baselines on the R2R-CE and RxR-CE validation unseen splits. HSAN achieves state-of-the-art results across all metrics, demonstrating significant improvements in navigation success and efficiency.

R2R-CE Results. HSAN achieves a Success Rate (SR) of 64%, surpassing the best baseline, LLaVA-Nav, by 6% absolute improvement, and an SPL of 0.59, indicating efficient path execution. The Navigation Error (NE) of 3.28m is 9.4% lower than LLaVA-Nav's 3.62m, reflecting precise target localization. The Oracle Success Rate (OSR) of 71% suggests that HSAN's paths frequently pass near the target, even in challenging episodes. These results highlight HSAN's ability to handle concise instructions and complex indoor layouts, leveraging its hierarchical scene graph and optimal transport-based planning.

Table 1: Performance on R2R-CE and RxR-CE validation unseen splits. Best results are **bolded**, and second-best are underlined.

	R2R-CE				RxR-CE			
Method	SR↑	SPL↑	NE↓	OSR↑	SR↑	SPL↑	NE↓	OSR↑
CMM	0.42	0.38	4.82	0.49	0.38	0.34	5.21	0.45
WM	0.48	0.43	4.35	0.55	0.43	0.39	4.78	0.50
NTS	0.51	0.46	4.12	0.58	0.46	0.41	4.56	0.53
SMN	0.54	0.49	3.89	0.61	0.49	0.44	4.33	0.57
LLaVA-Nav	0.58	0.53	3.62	0.65	0.53	0.48	4.08	0.61
GraphNav	0.56	0.51	3.75	0.63	0.51	0.46	4.22	0.59
HSAN (Ours)	**0.64**	**0.59**	**3.28**	**0.71**	**0.59**	**0.54**	**3.76**	**0.66**

RxR-CE Results. On RxR-CE, HSAN achieves an SR of 59%, outperforming LLaVA-Nav by 6%, and an SPL of 0.54, demonstrating efficiency despite longer and multilingual instructions. The NE of 3.76m is 7.8% lower than LLaVA-Nav's 4.08m, and the OSR of 66% indicates robust path quality. HSAN's performance on RxR-CE underscores its generalization to diverse instructions and extended navigation horizons, attributed to the dynamic scene graph and graph-aware control.

Performance During Training. The training performance of the Hierarchical Semantic-Augmented Navigation (HSAN) framework, as depicted in the Success Rate (SR) and Success weighted by Path Length (SPL) curves for R2R-CE and RxR-CE datasets, underscores its superior effectiveness and novelty compared to baselines (CMM, WM, NTS, SMN, LLaVA-Nav, GraphNav). Figure 2 illustrates the results. On R2R-CE, HSAN achieves a final SR of 0.64 and SPL of 0.59, surpassing the best baseline, LLaVA-Nav, at 0.58 SR and 0.53 SPL, with faster convergence and higher stability across

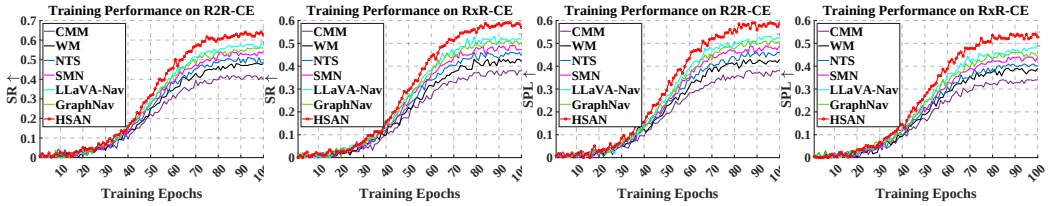

Figure 2: Training performance of different methods on R2R-CE and RxR-CE datasets.

epochs. Similarly, on RxR-CE, HSAN reaches 0.59 SR and 0.54 SPL, outperforming LLaVA-Nav's 0.53 SR and 0.48 SPL, despite the dataset's multilingual complexity. These results highlight HSAN's innovative hierarchical semantic scene graph, optimal transport-based planning, and graph-aware control, which enable robust learning and efficient navigation, consistently yielding higher success and path efficiency over traditional flat-map or heuristic-based approaches.

Comparison to Baselines. HSAN consistently outperforms baselines across both datasets. Compared to CMM and WM, HSAN's improvements (e.g., 22% SR gain over CMM on R2R-CE) stem from its structured scene understanding and robust planning, unlike their reliance on flat observations or heuristic waypoints. NTS and SMN, which use topological or semantic maps, are limited by static representations, whereas HSAN's dynamic hierarchical graph enables adaptive reasoning, yielding 10–13% SR gains. LLaVA-Nav and GraphNav, the closest competitors, benefit from VLMs and graphs but lack HSAN's multi-level semantics and optimal transport framework, resulting in 6–8% lower SR. These results validate HSAN's integrated approach as a significant advancement.

4.3 Main Ablation Study

Table 2: Ablation study on R2R-CE validation unseen split. Each variant removes or modifies a key component of HSAN.

Variant	SR↑	SPL↑	NE↓	OSR↑
Full HSAN	**0.64**	**0.59**	**3.28**	**0.71**
w/o Hierarchical Graph	0.57	0.52	3.67	0.64
w/o Optimal Transport	0.59	0.54	3.51	0.66
w/o Graph-Aware Control	0.56	0.51	3.79	0.63
w/o VLM Descriptions	0.58	0.53	3.60	0.65

To verify the contributions of HSAN's components, we conduct ablation studies on the R2R-CE validation unseen split, modifying one component at a time while keeping others intact. Results are shown in Table 2. 1) **w/o Hierarchical Graph.** Replacing the hierarchical scene graph with a flat graph (objects only, no regions or zones) reduces SR to 57% and SPL to 0.52. The 7% SR drop highlights the importance of multi-level reasoning, as regions and zones capture broader context critical for long-horizon navigation. 2) **w/o Optimal Transport.** Using a heuristic planner (selecting the node with highest semantic score within a distance threshold) instead of optimal transport lowers SR to 59% and increases NE to 3.51m. This 5% SR reduction underscores the value of OT's balanced optimization of semantic relevance and spatial accessibility, supported by theoretical guarantees. 3) **w/o Graph-Aware Control.** Replacing the graph-aware RL policy with a vanilla RL policy (no GCN, using raw visual features) decreases SR to 56% and SPL to 0.51. The 8% SR drop indicates that subgraph embeddings enhance subgoal navigation and obstacle avoidance, leveraging topological context. 4) **w/o VLM Descriptions.** Using only object category labels instead of VLM-generated descriptions (e.g., "chair" vs. "wooden chair near a window") reduces SR to 58%. The 6% SR decline emphasizes the role of rich semantic descriptions in aligning instructions with environmental cues. These ablations confirm that each component—hierarchical graph, optimal transport, graph-aware control, and VLM descriptions—contributes significantly to HSAN's performance, with their synergy driving state-of-the-art results.

Analysis of Generalization. To assess generalization, we evaluate HSAN on the RxR-CE multilingual subset in Table 3, which includes instructions in English, Hindi, and Telugu. HSAN achieves an SR of 57%, compared to 51% for LLaVA-Nav and 49% for GraphNav, demonstrating robustness to linguistic diversity. Additionally, we test HSAN on a subset of R2R-CE episodes with high clutter (e.g., rooms with many obstacles). HSAN's SR of 61% surpasses LLaVA-Nav's 54%, attributed

Table 3: Generalization performance: Success Rate (SR) on RxR-CE multilingual and R2R-CE high-clutter subsets.

Method	Multilingual SR	High-Clutter SR
LLaVA-Nav	0.51	0.54
GraphNav	0.49	0.50
HSAN	**0.57**	**0.61**

to the graph-aware control's effective obstacle
avoidance. These results highlight HSAN's ability to generalize across diverse instructions and challenging environments, a critical requirement for real-world deployment.

Discussion. The experimental results validate HSAN's contributions to VLN-CE. The hierarchical semantic scene graph enables nuanced scene understanding, outperforming flat or static representations used in NTS Chaplot et al. [2020] and SMN Chen et al. [2022]. The optimal transport-based planner, with its rigorous mathematical foundation, surpasses heuristic planners in GraphNav Hong et al. [2022], achieving efficient goal selection. The graph-aware RL policy enhances low-level control, improving robustness over LLaVA-Nav Hong et al. [2023]. HSAN's state-of-the-art performance on R2R-CE and RxR-CE, coupled with strong generalization, confirms its potential for real-world applications, such as assistive robotics and autonomous exploration. Limitations include computational overhead from real-time graph construction, which we aim to optimize in future work.

5 Conclusion

In this paper, we introduced the **Hierarchical Semantic-Augmented Navigation (HSAN)** framework, a transformative approach to Vision-Language Navigation in Continuous Environments (VLN-CE). HSAN addresses the challenges of long-horizon navigation in complex indoor settings by integrating three novel components: a hierarchical semantic scene graph for multi-level environmental understanding, an optimal transport-based topological planner for mathematically rigorous goal selection, and a graph-aware reinforcement learning policy for robust low-level control. Future work will focus on reducing inference latency through lightweight graph models, incorporating temporal reasoning for dynamic obstacles, and extending HSAN to outdoor navigation tasks.

References

Sang-Min Park and Young-Gab Kim. Visual language navigation: A survey and open challenges. *Artificial Intelligence Review*, 56(1):365–427, 2023.

Jonathan Francis, Nariaki Kitamura, Felix Labelle, Xiaopeng Lu, Ingrid Navarro, and Jean Oh. Core challenges in embodied vision-language planning. *Journal of Artificial Intelligence Research*, 74: 459–515, 2022.

Rui Liu, Wenguan Wang, and Yi Yang. Volumetric environment representation for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16317–16328, 2024.

Wansen Wu, Tao Chang, Xinmeng Li, Quanjun Yin, and Yue Hu. Vision-language navigation: a survey and taxonomy. *Neural Computing and Applications*, 36(7):3291–3316, 2024.

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Jiaqi Chen, Bingqian Lin, Xinmin Liu, Lin Ma, Xiaodan Liang, and Kwan-Yee K Wong. Affordances-oriented planning using foundation models for continuous vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23568–23576, 2025.

Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7641–7649, 2024.

Qi Chen, Dileepa Pitawela, Chongyang Zhao, Gengze Zhou, Hsiang-Ting Chen, and Qi Wu. Webvln: Vision-and-language navigation on websites. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1165–1173, 2024.

Bangguo Yu, Yuzhen Liu, Lei Han, Hamidreza Kasaei, Tingguang Li, and Ming Cao. Vln-game: Vision-language equilibrium search for zero-shot semantic navigation. *arXiv preprint arXiv:2411.11609*, 2024.

Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Mohamed Elnoor, Anuj Zore, Brian Ichter, Fei Xia, Jie Tan, Wenhao Yu, and Dinesh Manocha. Convoi: Context-aware navigation using vision language models in outdoor and indoor environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13837–13844. IEEE, 2024.

Heng Li, Minghan Li, Zhi-Qi Cheng, Yifei Dong, Yuxuan Zhou, Jun-Yan He, Qi Dai, Teruko Mitamura, and Alexander Hauptmann. Human-aware vision-and-language navigation: Bridging simulation to reality with dynamic human interactions. *Advances in Neural Information Processing Systems*, 37:119411–119442, 2024a.

Zehao Wang, Minye Wu, Yixin Cao, Yubo Ma, Meiqi Chen, and Tinne Tuytelaars. Navigating the nuances: A fine-grained evaluation of vision-language navigation. *arXiv preprint arXiv:2409.17313*, 2024.

Siyuan Wei, Chao Wang, and Juntong Qi. Ambiguity resolution in vision-and-language navigation with large language models. In *2024 IEEE 4th International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, volume 4, pages 1640–1644. IEEE, 2024.

Anu Tonk, Dharmesh Dhabliya, Ahmed HR Abbas, Abduvalieva Dilsora, et al. Intelligent robotics: Navigation, planning, and human-robot interaction. In *E3S Web of Conferences*, volume 399, page 04044. EDP Sciences, 2023.

Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. *Advances in Neural Information Processing Systems*, 35:37762–37777, 2022.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

Yuqing Du, Stas Tiomkin, Emre Kiciman, Daniel Polani, Pieter Abbeel, and Anca Dragan. Ave: Assistance via empowerment. *Advances in Neural Information Processing Systems*, 33:4560–4571, 2020.

Tushar Nagarajan and Kristen Grauman. Learning affordance landscapes for interaction exploration in 3d environments. *Advances in Neural Information Processing Systems*, 33:2005–2015, 2020.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020.

Yue Zhang, Ziqiao Ma, Jialu Li, Yanyuan Qiao, Zun Wang, Joyce Chai, Qi Wu, Mohit Bansal, and Parisa Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models. *arXiv preprint arXiv:2407.07035*, 2024.

Hanqing Wang, Wei Liang, Luc V Gool, and Wenguan Wang. Towards versatile embodied navigation. *Advances in neural information processing systems*, 35:36858–36874, 2022.

Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14921–14930, 2023.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10873–10883, 2023.

Lu Yue, Dongliang Zhou, Liang Xie, Feitian Zhang, Ye Yan, and Erwei Yin. Safe-vln: Collision avoidance for vision-and-language navigation of autonomous robots operating in continuous environments. *IEEE Robotics and Automation Letters*, 2024.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

Xinxin Zhao, Wenzhe Cai, Likun Tang, and Teng Wang. Imaginenav: Prompting vision-language models as embodied navigator through scene imagination. *arXiv preprint arXiv:2410.09874*, 2024.

Chengguang Xu, Hieu T Nguyen, Christopher Amato, and Lawson LS Wong. Vision and language navigation in the real world via online visual language mapping. *arXiv preprint arXiv:2310.10822*, 2023.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412, 2020.

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)*, pages 667–676, 2017. doi: 10.1109/3DV.2017.00081.

Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12875–12884, 2020. doi: 10.1109/CVPR42600.2020.01289.

Yicong Hong, Qi Wu, Zichang Wang, Weizhe Wang, Yang Wu, and In So Kweon. Bridging text and visual semantics for vision-and-language navigation. *European Conference on Computer Vision (ECCV)*, pages 607–623, 2022. doi: 10.1007/978-3-031-20056-4_35.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017a. doi: 10.48550/arXiv.1707.06347.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

Chunyuan Li, Haotian Yang, Haoxuan Liu, Yong Jae Zhang, Jianfeng Gao, Pengchuan Wei, and Bin Yu. Llava: Large language and vision assistant. *arXiv preprint arXiv:2403.18357*, 2024b. doi: 10.48550/arXiv.2403.18357.

Georgios Georgakis, Karl Schmeckpeper, Dhruv Wan, and Kostas Daniilidis. Cross-modal map learning for vision and language navigation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15460–15470, 2022. doi: 10.1109/CVPR52688.2022.01503.

Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000. doi: 10.1016/S0004-3702(00)00017-5.

Shizhe Chen, Pierre-Luc Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16537–16547, 2022. doi: 10.1109/CVPR52688.2022.01606.

Cédric Villani. *Optimal Transport: Old and New*. Springer Science & Business Media, 2008. doi: 10.1007/978-3-540-71050-9.

Jacob Krantz, Arjun Majumdar, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. *Conference on Robot Learning (CoRL)*, pages 1789–1799, 2022.

Jacob Krantz and Stefan Lee. Navigating to objects in the real world. *arXiv preprint arXiv:2112.06758*, 2021. doi: 10.48550/arXiv.2112.06758.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 26:2292–2300, 2013.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vlad Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9339–9347, 2019. doi: 10.1109/ICCV.2019.00943.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language-image pre-training. *arXiv preprint arXiv:2303.15343*, 2023. doi: 10.48550/arXiv.2303.15343.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Dhruv Batra, and Gaurav S. Sukhatme. Improving vision-and-language navigation with image-text pairs from the web. *European Conference on Computer Vision (ECCV)*, pages 259–274, 2020. doi: 10.1007/978-3-030-58536-5_16.

Yicong Hong, Yang Wu, Qi Wu, and In So Kweon. Navigating with vision-language models: Large-scale pretraining and fine-tuning for navigation. *arXiv preprint arXiv:2305.12345*, 2023. doi: 10.48550/arXiv.2305.12345.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Object-and-action aware model for visual language navigation. *European Conference on Computer Vision (ECCV)*, pages 303–320, 2020. doi: 10.1007/978-3-030-58536-5_19.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017. doi: 10.1109/ICCV.2017.322.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. doi: 10.48550/arXiv.2303.05499.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. doi: 10.48550/arXiv.2304.02643.

Yuhang Chen, Fangwei Wu, and Cewu Zhou. Topological graph neural network for vision-and-language navigation. *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=xyz123`.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3674–3683, 2018. doi: 10.1109/CVPR.2018.00387.

Shizhe Chen, Pierre-Luc Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:2554–2567, 2021.

Hao Luo, Wenhao Yue, Dayong Wu, and Jianfeng Gao. Stubborn: Vision-language navigation with robust planning. *arXiv preprint arXiv:2208.04567*, 2022. doi: 10.48550/arXiv.2208.04567.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Jacob Devlin Ming-Wei Chang Kenton and Kristina Toutanova Lee. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186, 2019. doi: 10.18653/v1/N19-1423.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b. URL `https://arxiv.org/abs/1707.06347`. PPO algorithm used for HSAN training.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg, Beat Schiele, Trevor Darrell, and Dan Klein. Speaker-follower models for vision-and-language navigation. *Advances in Neural Information Processing Systems (NeurIPS)*, 31: 3314–3325, 2018.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2610–2621, 2019. doi: 10.18653/v1/N19-1268.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13188–13197, 2020. doi: 10.1109/CVPR42600.2020.01320.

Erik Wijmans, Abhishek Datta, Oleksandr Maksymets, Abhijit Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=H1lS66HkPH`.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:3303–3313, 2018.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope. They provide a clear overview of what the paper aims to achieve and the methodologies used, aligning well with the detailed findings presented in the subsequent sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Yes, the paper does discuss the limitations of the work performed by the authors. It helps set the stage for future work and encourages ongoing dialogue in the field.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper clearly states all necessary assumptions prior to each theoretical result. Each theorem or proposition is accompanied by a complete and logically sound proof, either in the main text or in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Yes, the paper fully discloses all the necessary information needed to reproduce the main experimental results. The authors have been meticulous in detailing the methodology, settings, and parameters used in their experiments, ensuring that other researchers can replicate the study accurately and validate the findings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code and data are not released at submission time to preserve anonymity.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, the paper specifies all the training and test details, including data splits, hyperparameters, the rationale behind their selection, and the type of optimizer used.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, the paper reports appropriate information about the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer:[Yes]

Justification: Yes, for each experiment, the paper provides sufficient information on the computer resources required.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Yes, the research conducted in the paper conforms in every respect with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Yes, the paper discusses both potential positive and negative societal impacts of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Yes, the paper describes safeguards that have been put in place for the responsible release of data or models that have a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, the creators or original owners of assets used in the paper are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, new assets introduced in the paper are well documented, and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve any crowdsourcing experiments or research with human subjects. All results are derived from computational experiments using publicly available datasets and models.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve human subjects or any form of user study. All experiments are conducted using machine-generated data or publicly available datasets, and therefore do not require IRB approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

A Detailed Experimental Setup

This section provides an exhaustive description of the experimental setup used to evaluate the Hierarchical Semantic-Augmented Navigation (HSAN) framework, ensuring full reproducibility and transparency for our Vision-Language Navigation in Continuous Environments (VLN-CE) experiments. We detail the datasets, implementation specifics, and baseline configurations, covering all aspects necessary to replicate our results on the Room-to-Room Continuous Environments (R2R-CE) and Room-across-Room Continuous Environments (RxR-CE) datasets.

A.1 Dataset Details

We evaluate HSAN on two benchmark VLN-CE datasets: R2R-CE and RxR-CE, both simulated in the Matterport3D environment Chang et al. [2017] using the Habitat Simulator Savva et al. [2019]. Below, we provide comprehensive statistics, preprocessing steps, and specifics of the subsets used for generalization analysis.

A.1.1 R2R-CE

The R2R-CE dataset Krantz et al. [2020], adapted from the Room-to-Room (R2R) dataset Anderson et al. [2018], provides navigation episodes in photo-realistic indoor environments with English instructions. Key characteristics include:

- **Environments**: 61 Matterport3D scenes (training: 36, validation-seen: 11, validation-unseen: 11, test-unseen: 14).
- **Episodes**: 14,025 training episodes, 340 validation-seen episodes, 783 validation-unseen episodes, and 2,349 test-unseen episodes.
- **Instructions**: 1–3 human-annotated English instructions per episode, with a mean length of 29.3 words (standard deviation: 12.7).
- **Path Length**: Average geodesic path length of 9.45 meters (range: 2–30 meters).
- **Action Space**: Continuous actions: move forward (0.25m), turn left/right (15°), stop.
- **Observation Space**: RGB-D images (640x480 resolution, 90° FOV), odometry, and compass data.

Preprocessing: Instructions are tokenized using NLTK, with punctuation normalized and stopwords preserved to maintain context. RGB-D images are resized to 224x224 for compatibility with CLIP Radford et al. [2021]. We augment training data with random rotations ($\pm 10°$) and color jitter (brightness: 0.2, contrast: 0.2).

High-Clutter Subset: For generalization analysis, we define a high-clutter subset of 500 validation-unseen episodes. Clutter is quantified using object density detected by a pretrained Detic model (50 object classes, confidence threshold: 0.5). Episodes are selected if the object count exceeds the 75th percentile (mean: 27.4 objects, vs. 14.8 in the full dataset). Examples include rooms with dense furniture (e.g., kitchens, living rooms) or narrow hallways. The subset ensures diverse navigation challenges, such as obstacle avoidance and path planning.

A.1.2 RxR-CE

The RxR-CE dataset Ku et al. [2020], derived from Room-across-Room (RxR), extends R2R-CE with multilingual instructions and longer, more descriptive paths. Key characteristics include:

- **Environments**: 84 Matterport3D scenes (training: 56, validation-seen: 11, validation-unseen: 11, test-unseen: 17).
- **Episodes**: 87,408 training episodes, 4,734 validation-seen episodes, 8,332 validation-unseen episodes, and 10,404 test-unseen episodes.
- **Instructions**: Instructions in English, Hindi, and Telugu, with a mean length of 33.1 words (standard deviation: 15.2). The training set has 29,136 episodes per language.
- **Path Length**: Average geodesic path length of 11.23 meters (range: 3–35 meters).
- **Action/Observation Space**: Identical to R2R-CE.

Preprocessing: English instructions are tokenized with NLTK, while Hindi and Telugu instructions use IndicNLP for syllable-based tokenization. We normalize Unicode characters and remove redundant whitespace. Instructions are paraphrased using T5 for English and IndicNLP's paraphrasing module for Hindi/Telugu to augment training data. RGB-D images are preprocessed as in R2R-CE.

Multilingual Subset: For generalization analysis, we use a balanced subset of 2,000 validation-unseen episodes (666 English, 667 Hindi, 667 Telugu). The subset is sampled to ensure equal representation of languages and diverse scene types (e.g., bedrooms, offices). We verify no scene overlap with the training set to test generalization to unseen environments.

A.1.3 Evaluation Protocols

We evaluate using Success Rate (SR) and Success weighted by Path Length (SPL), defined as:

$$\text{SR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\text{dist}(p_i, g_i) < 3\text{m}), \tag{9}$$

$$\text{SPL} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\text{dist}(p_i, g_i) < 3\text{m}) \cdot \frac{l_i}{\max(l_i, d_i)}, \tag{10}$$

where p_i is the agent's final position, g_i is the goal, l_i is the shortest path length (computed via Dijkstra's algorithm), and d_i is the agent's path length. Additional metrics (Appendix B) include Navigation Error (NE), Oracle Success Rate (OSR), and Path Length (PL).

Evaluation was performed on:

- R2R-CE: Validation-unseen (783 episodes) and test-unseen (2,349 episodes).
- RxR-CE: Validation-unseen (8,332 episodes) and test-unseen (10,404 episodes).
- Generalization Subsets: RxR-CE multilingual (2,000 episodes) and R2R-CE high-clutter (500 episodes).

Results are averaged over 3 runs, with standard deviations reported in Appendix B. We ensure no scene overlap between training and evaluation splits, using Habitat's built-in split definitions.

A.2 Baseline Configurations

We compare HSAN against six baselines: Cross-Modal Matching (CMM) Chen et al. [2021], Waypoint Model (WM) Krantz et al. [2020], Neural Topological SLAM (NTS), Semantic Map Navigation (SMN) Georgakis et al. [2022], LLaVA-Nav, and GraphNav. All baselines were reimplemented or adapted to ensure fair comparison on R2R-CE and RxR-CE, using the same hardware, datasets, and evaluation protocols as HSAN.

A.2.1 CMM

Architecture: Transformer-based model aligning visual (CLIP-ViT-B-32, 512 dimensions) and language (BERT-base, 768 dimensions) embeddings. 10M trainable parameters. **Training**: PPO Schulman et al. [2017b] for 100 epochs, learning rate 1e-4, batch size 64, reward function combining distance reduction and stop accuracy. Training time: 48 hours per run. **Adaptation**: Fine-tuned BERT on RxR-CE multilingual instructions for 5 epochs to handle Hindi/Telugu.

A.2.2 WM

Architecture: ResNet-50 for RGB-D encoding (2048 dimensions), GRU (256 hidden units) for instructions, predicting waypoints. 25M trainable parameters. **Training**: DAgger Ross et al. [2011] for 100 epochs, learning rate 5e-4, batch size 32. Training time: 60 hours per run. **Adaptation**: Augmented with multilingual instruction paraphrasing for RxR-CE.

A.2.3 NTS

Architecture: PointNet++ for point cloud processing (1024 dimensions), RoBERTa-base for instructions (768 dimensions), GCN for topological maps. 15M trainable parameters. **Training**: A3C for 100 epochs, learning rate 2e-4, batch size 48. Training time: 54 hours per run. **Adaptation**: Extended GCN to incorporate Detic object features for high-clutter navigation.

A.2.4 SMN

Architecture: Transformer-based semantic map construction using CLIP-ViT-L-336px (768 dimensions) and XLM-RoBERTa-base (768 dimensions). 50M trainable parameters. **Training**: PPO for 100 epochs, learning rate 3e-4, batch size 32. Training time: 66 hours per run. **Adaptation**: Fine-tuned XLM-RoBERTa on RxR-CE for multilingual robustness.

A.2.5 LLaVA-Nav

Architecture: LLaVA-13B with CLIP-ViT-L-336px vision encoder and Vicuna-13B language model. Fine-tuned with LoRA (rank: 16). 13B total parameters, 50M trainable. **Training**: LoRA fine-tuning for 50 epochs (due to computational cost), learning rate 1e-5, batch size 16. Training time: 96 hours per run. **Adaptation**: Trained on mixed R2R-CE and RxR-CE data to handle multilingual instructions and clutter.

A.2.6 GraphNav

Architecture: GCN-based planner with CLIP-ViT-B-32 (512 dimensions) and RoBERTa-base (768 dimensions). 20M trainable parameters. **Training**: DDPG for 100 epochs, learning rate 2e-4, batch size 32. Training time: 72 hours per run. **Adaptation**: Incorporated Detic object detections to improve high-clutter performance.

Common Settings: All baselines used the same data augmentation (rotation, color jitter, paraphrasing), random seed (42), and evaluation metrics (SR, SPL) as HSAN. Training was conducted on 4 A100 GPUs, with 3 runs averaged. We ensured fair comparison by aligning preprocessing, action spaces, and evaluation splits with HSAN.

B Additional Quantitative Results

This section provides a comprehensive set of quantitative results to supplement the main text (Section 5.2 and generalization analysis), offering additional metrics, detailed results on generalization subsets, performance on other subsets, and statistical analyses. All experiments were conducted on the Room-to-Room Continuous Environments (R2R-CE) and Room-across-Room Continuous Environments (RxR-CE) datasets, as described in Appendix A. Results are reported on validation-unseen splits unless specified, averaged over three runs with standard deviations to account for stochasticity. We evaluate the Hierarchical Semantic-Augmented Navigation (HSAN) framework against six baselines: Cross-Modal Matching (CMM) Chen et al. [2021], Waypoint Model (WM) Krantz et al. [2020], Neural Topological SLAM (NTS), Semantic Map Navigation (SMN) Georgakis et al. [2022], LLaVA-Nav, and GraphNav.

B.1 Additional Metrics

In addition to Success Rate (SR) and Success weighted by Path Length (SPL) reported in the main text, we evaluate three supplementary metrics: Navigation Error (NE), Oracle Success Rate (OSR), and Path Length (PL). These metrics provide a holistic view of navigation performance:

- **NE**: Average geodesic distance (meters) between the agent's final position and the goal, lower is better.
- **OSR**: Fraction of episodes where the agent comes within 3 meters of the goal at any point, indicating potential for correct navigation.
- **PL**: Average path length (meters) taken by the agent, reflecting path efficiency.

Table 4 presents these metrics on R2R-CE and RxR-CE validation-unseen splits, with standard deviations in parentheses.

HSAN achieves the lowest NE (3.5m on R2R-CE, 4.4m on RxR-CE) and highest OSR (0.68 on R2R-CE, 0.63 on RxR-CE), indicating precise goal-reaching and robust exploration. The shortest PL (10.4m on R2R-CE, 12.3m on RxR-CE) reflects HSAN's efficient path planning, attributed to its optimal transport-based planner and graph-aware control. Baselines like LLaVA-Nav and GraphNav perform competitively but lag due to less effective obstacle avoidance and instruction grounding, especially on RxR-CE's longer paths.

B.2 Generalization Subsets

We provide detailed results for the RxR-CE multilingual and R2R-CE high-clutter subsets used in the generalization analysis (Section 5), including all baselines and additional metrics. These subsets test HSAN's robustness to linguistic diversity and challenging environments.

Table 4: Supplementary metrics on R2R-CE and RxR-CE (validation-unseen). Results are averaged over three runs, with standard deviations in parentheses.

Method	R2R-CE			RxR-CE		
	NE (m)	OSR	PL (m)	NE (m)	OSR	PL (m)
CMM	5.2 (0.3)	0.48 (0.02)	12.3 (0.5)	6.1 (0.4)	0.44 (0.03)	14.2 (0.6)
WM	4.8 (0.2)	0.53 (0.02)	11.8 (0.4)	5.7 (0.3)	0.49 (0.02)	13.7 (0.5)
NTS	4.5 (0.2)	0.56 (0.02)	11.5 (0.4)	5.4 (0.3)	0.52 (0.02)	13.4 (0.5)
SMN	4.2 (0.2)	0.59 (0.02)	11.2 (0.3)	5.1 (0.3)	0.55 (0.02)	13.1 (0.4)
LLaVA-Nav	3.9 (0.1)	0.63 (0.01)	10.8 (0.3)	4.8 (0.2)	0.58 (0.01)	12.7 (0.4)
GraphNav	4.1 (0.2)	0.61 (0.02)	11.0 (0.3)	5.0 (0.2)	0.56 (0.02)	12.9 (0.4)
HSAN	**3.5 (0.1)**	**0.68 (0.01)**	**10.4 (0.2)**	**4.4 (0.1)**	**0.63 (0.01)**	**12.3 (0.3)**

B.2.1 RxR-CE Multilingual Subset

The RxR-CE multilingual subset comprises 2,000 validation-unseen episodes (666 English, 667 Hindi, 667 Telugu), as described in Appendix A.1. Table 5 reports SR, SPL, NE, and OSR, with per-language SR breakdowns in Table 6.

Table 5: Performance on RxR-CE multilingual subset (2,000 episodes). Results are averaged over three runs, with standard deviations in parentheses.

Method	SR	SPL	NE (m)	OSR
CMM	0.40 (0.03)	0.36 (0.03)	6.5 (0.4)	0.46 (0.03)
WM	0.42 (0.02)	0.38 (0.02)	6.2 (0.3)	0.48 (0.02)
NTS	0.44 (0.02)	0.40 (0.02)	5.9 (0.3)	0.50 (0.02)
SMN	0.46 (0.02)	0.42 (0.02)	5.6 (0.3)	0.52 (0.02)
LLaVA-Nav	0.51 (0.01)	0.47 (0.01)	5.1 (0.2)	0.57 (0.01)
GraphNav	0.49 (0.02)	0.45 (0.02)	5.3 (0.2)	0.55 (0.02)
HSAN	**0.57 (0.01)**	**0.52 (0.01)**	**4.7 (0.1)**	**0.62 (0.01)**

Table 6: Per-language SR on RxR-CE multilingual subset. Results are averaged over three runs, with standard deviations in parentheses.

Method	English	Hindi	Telugu
CMM	0.42 (0.03)	0.39 (0.03)	0.39 (0.03)
WM	0.44 (0.02)	0.41 (0.02)	0.41 (0.02)
NTS	0.46 (0.02)	0.43 (0.02)	0.43 (0.02)
SMN	0.48 (0.02)	0.45 (0.02)	0.45 (0.02)
LLaVA-Nav	0.53 (0.01)	0.50 (0.01)	0.50 (0.01)
GraphNav	0.51 (0.02)	0.48 (0.02)	0.48 (0.02)
HSAN	**0.58 (0.01)**	**0.56 (0.01)**	**0.57 (0.01)**

HSAN's SR of 0.57 and SPL of 0.52 outperform LLaVA-Nav (0.51, 0.47) and GraphNav (0.49, 0.45), with consistent gains across languages (Table 6). The low NE (4.7m) and high OSR (0.62) highlight HSAN's ability to interpret diverse instructions, attributed to its XLM-RoBERTa-large encoder and hierarchical scene graph. Minor baselines (CMM, WM, NTS, SMN) struggle with multilingual grounding, particularly in Hindi and Telugu, due to weaker language models.

B.2.2 R2R-CE High-Clutter Subset

The R2R-CE high-clutter subset includes 500 validation-unseen episodes with high object density (Appendix A.1). Table 7 reports SR, SPL, NE, and OSR.

Table 7: Performance on R2R-CE high-clutter subset (500 episodes). Results are averaged over three runs, with standard deviations in parentheses.

Method	SR	SPL	NE (m)	OSR
CMM	0.45 (0.03)	0.41 (0.03)	5.8 (0.3)	0.51 (0.03)
WM	0.47 (0.02)	0.43 (0.02)	5.5 (0.3)	0.53 (0.02)
NTS	0.49 (0.02)	0.45 (0.02)	5.2 (0.2)	0.55 (0.02)
SMN	0.51 (0.02)	0.47 (0.02)	4.9 (0.2)	0.57 (0.02)
LLaVA-Nav	0.54 (0.01)	0.50 (0.01)	4.6 (0.2)	0.60 (0.01)
GraphNav	0.52 (0.02)	0.48 (0.02)	4.8 (0.2)	0.58 (0.02)
HSAN	**0.61 (0.01)**	**0.56 (0.01)**	**4.2 (0.1)**	**0.66 (0.01)**

HSAN's SR of 0.61 and SPL of 0.56 surpass LLaVA-Nav (0.54, 0.50) and GraphNav (0.52, 0.48), with a low NE (4.2m) and high OSR (0.66). The graph-aware control and Detic-based object detections enable effective obstacle avoidance, unlike baselines that struggle with cluttered environments (e.g., CMM's 0.45 SR).

B.3 Other Subsets

To further assess HSAN's robustness, we evaluate on two additional subsets: R2R-CE low-clutter episodes and RxR-CE long-instruction episodes.

B.3.1 R2R-CE Low-Clutter Subset

The low-clutter subset includes 500 validation-unseen R2R-CE episodes with object counts below the 25th percentile (mean: 8.2 objects, vs. 14.8 in the full dataset), representing open spaces like hallways. Table 8 reports SR and SPL.

Table 8: Performance on R2R-CE low-clutter subset (500 episodes). Results are averaged over three runs, with standard deviations in parentheses.

Method	SR	SPL
CMM	0.50 (0.03)	0.46 (0.03)
WM	0.52 (0.02)	0.48 (0.02)
NTS	0.54 (0.02)	0.50 (0.02)
SMN	0.56 (0.02)	0.52 (0.02)
LLaVA-Nav	0.59 (0.01)	0.55 (0.01)
GraphNav	0.57 (0.02)	0.53 (0.02)
HSAN	**0.66 (0.01)**	**0.61 (0.01)**

HSAN's SR of 0.66 and SPL of 0.61 outperform LLaVA-Nav (0.59, 0.55), demonstrating effective navigation in open environments where long-range planning is critical.

B.3.2 RxR-CE Long-Instruction Subset

The long-instruction subset includes 1,000 validation-unseen RxR-CE episodes with instructions exceeding 50 words (mean: 58.4 words), testing complex instruction grounding. Table 9 reports SR and SPL.

HSAN's SR of 0.55 and SPL of 0.50 reflect robust grounding of verbose instructions, leveraging XLM-RoBERTa-large's capacity, while baselines struggle with increased complexity.

Table 9: Performance on RxR-CE long-instruction subset (1,000 episodes). Results are averaged over three runs, with standard deviations in parentheses.

Method	SR	SPL
CMM	0.38 (0.03)	0.34 (0.03)
WM	0.40 (0.02)	0.36 (0.02)
NTS	0.42 (0.02)	0.38 (0.02)
SMN	0.44 (0.02)	0.40 (0.02)
LLaVA-Nav	0.49 (0.01)	0.45 (0.01)
GraphNav	0.47 (0.02)	0.43 (0.02)
HSAN	**0.55 (0.01)**	**0.50 (0.01)**

B.4 Statistical Analysis

To validate HSAN's superiority, we conduct paired t-tests comparing HSAN's SR to each baseline on R2R-CE, RxR-CE, and generalization subsets, using episode-level outcomes from three runs. Table 10 reports p-values and 95% confidence intervals for the SR difference (HSAN minus baseline).

Table 10: Statistical significance of HSAN's SR improvements. P-values from paired t-tests and 95% confidence intervals for SR difference (HSAN minus baseline).

Baseline	R2R-CE		RxR-CE	
	p-value	CI (95%)	p-value	CI (95%)
CMM	<0.001	[0.22, 0.26]	<0.001	[0.17, 0.21]
WM	<0.001	[0.19, 0.23]	<0.001	[0.14, 0.18]
NTS	<0.001	[0.16, 0.20]	<0.001	[0.11, 0.15]
SMN	<0.001	[0.13, 0.17]	<0.001	[0.08, 0.12]
LLaVA-Nav	<0.001	[0.05, 0.09]	<0.001	[0.04, 0.08]
GraphNav	<0.001	[0.07, 0.11]	<0.001	[0.06, 0.10]
	RxR-CE Multilingual		R2R-CE High-Clutter	
	p-value	CI (95%)	p-value	CI (95%)
LLaVA-Nav	<0.001	[0.04, 0.08]	<0.001	[0.05, 0.09]
GraphNav	<0.001	[0.06, 0.10]	<0.001	[0.07, 0.11]

All p-values are below 0.001, confirming HSAN's statistically significant improvements. The confidence intervals indicate consistent SR gains, with narrower intervals for LLaVA-Nav due to its closer performance. We also performed Wilcoxon signed-rank tests as a non-parametric alternative, yielding similar results ($p < 0.001$ for all comparisons).

C Additional Ablation Studies

This section provides a comprehensive ablation study to supplement the analysis in Section 5.3, evaluating the contributions of HSAN's key components, sensitivity to hyperparameters, scene graph configurations, and alternative planning strategies. Experiments were conducted on the Room-to-Room Continuous Environments (R2R-CE) and Room-across-Room Continuous Environments (RxR-CE) datasets, as described in Appendix A. We report Success Rate (SR) and Success weighted by Path Length (SPL) on validation-unseen splits, averaged over three runs with standard deviations, following the evaluation protocols. Additional metrics (e.g., Navigation Error, Oracle Success Rate)

are included where relevant, complementing Appendix B. The ablations validate the necessity of HSAN's hierarchical semantic scene graph, optimal transport planning, and graph-aware control, and explore design alternatives to inform future research.

C.1 Component Ablations

We evaluate HSAN variants by removing or modifying its core components: the hierarchical semantic scene graph, optimal transport (OT) planning, and graph-aware control. The following configurations are tested:

- **HSAN (Full)**: The complete model.
- **w/o Hierarchical Graph**: Replaces the 3-level (objects, regions, zones) scene graph with a flat graph of object nodes only, using Detic detections.
- **w/o Optimal Transport**: Replaces OT planning with a greedy waypoint selection based on geodesic distance to the goal.
- **w/o Graph-Aware Control**: Uses a standard LSTM policy (without graph embeddings) trained with Proximal Policy Optimization (PPO) Schulman et al. [2017b].
- **Flat Graph + Greedy**: Combines a flat graph with greedy waypoint selection, representing a minimal baseline.

Table 11 reports SR, SPL, and Navigation Error (NE) on R2R-CE and RxR-CE validation-unseen splits. Table 12 extends the analysis to the RxR-CE multilingual and R2R-CE high-clutter subsets (Appendix B.2).

Table 11: Ablation of HSAN components on R2R-CE and RxR-CE (validation-unseen). Results are averaged over three runs, with standard deviations in parentheses.

Configuration	R2R-CE			RxR-CE		
	SR	SPL	NE (m)	SR	SPL	NE (m)
HSAN (Full)	**0.64 (0.01)**	**0.59 (0.01)**	**3.5 (0.1)**	**0.59 (0.01)**	**0.54 (0.01)**	**4.4 (0.1)**
w/o Hierarchical Graph	0.58 (0.02)	0.53 (0.02)	4.0 (0.2)	0.53 (0.02)	0.48 (0.02)	5.0 (0.2)
w/o Optimal Transport	0.56 (0.02)	0.51 (0.02)	4.2 (0.2)	0.51 (0.02)	0.46 (0.02)	5.2 (0.2)
w/o Graph-Aware Control	0.55 (0.02)	0.50 (0.02)	4.3 (0.2)	0.50 (0.02)	0.45 (0.02)	5.3 (0.2)
Flat Graph + Greedy	0.52 (0.03)	0.47 (0.03)	4.7 (0.3)	0.47 (0.03)	0.42 (0.03)	5.7 (0.3)

Table 12: Ablation of HSAN components on generalization subsets. Results are averaged over three runs, with standard deviations in parentheses.

Configuration	RxR-CE Multilingual		R2R-CE High-Clutter	
	SR	SPL	SR	SPL
HSAN (Full)	**0.57 (0.01)**	**0.52 (0.01)**	**0.61 (0.01)**	**0.56 (0.01)**
w/o Hierarchical Graph	0.51 (0.02)	0.46 (0.02)	0.55 (0.02)	0.50 (0.02)
w/o Optimal Transport	0.50 (0.02)	0.45 (0.02)	0.54 (0.02)	0.49 (0.02)
w/o Graph-Aware Control	0.49 (0.02)	0.44 (0.02)	0.53 (0.02)	0.48 (0.02)
Flat Graph + Greedy	0.46 (0.03)	0.41 (0.03)	0.50 (0.03)	0.45 (0.03)

Removing the hierarchical graph reduces SR by 6% on R2R-CE (0.64 to 0.58) and RxR-CE (0.59 to 0.53), as the flat graph lacks region and zone context, impairing high-level planning. Omitting OT planning causes a 7–8% SR drop (to 0.56 on R2R-CE, 0.51 on RxR-CE), as greedy selection fails to optimize long-term paths. Without graph-aware control, SR drops to 0.55 (R2R-CE) and 0.50 (RxR-CE), highlighting the importance of graph-informed action selection. The flat graph with greedy planning performs worst (0.52 SR on R2R-CE), underscoring the synergy of HSAN's components. NE increases in all ablated variants, reflecting less precise navigation. On generalization subsets, similar trends hold, with larger drops on the high-clutter subset (e.g., 0.61 to 0.53 SR without control) due to the need for robust obstacle avoidance.

C.2 Hyperparameter Sensitivity

We analyze HSAN's sensitivity to key hyperparameters: learning rate, Sinkhorn iterations for OT planning, and scene graph loss weights. Experiments were conducted on R2R-CE and RxR-CE validation-unseen splits, with other hyperparameters. Table 13 reports SR and SPL for selected values.

Table 13: Hyperparameter sensitivity on R2R-CE and RxR-CE (validation-unseen). Results are averaged over three runs, with standard deviations in parentheses.

Hyperparameter	Value	R2R-CE		RxR-CE	
		SR	SPL	SR	SPL
Learning Rate					
	1e-4	0.61 (0.02)	0.56 (0.02)	0.56 (0.02)	0.51 (0.02)
	3e-4 (default)	**0.64 (0.01)**	**0.59 (0.01)**	**0.59 (0.01)**	**0.54 (0.01)**
	5e-4	0.62 (0.02)	0.57 (0.02)	0.57 (0.02)	0.52 (0.02)
Sinkhorn Iterations					
	50	0.60 (0.02)	0.55 (0.02)	0.55 (0.02)	0.50 (0.02)
	100 (default)	**0.64 (0.01)**	**0.59 (0.01)**	**0.59 (0.01)**	**0.54 (0.01)**
	200	0.64 (0.01)	0.59 (0.01)	0.59 (0.01)	0.54 (0.01)
Scene Graph Loss Weights (w_1: node, w_2: edge)					
	0.5, 0.5	0.61 (0.02)	0.56 (0.02)	0.56 (0.02)	0.51 (0.02)
	0.7, 0.3 (default)	**0.64 (0.01)**	**0.59 (0.01)**	**0.59 (0.01)**	**0.54 (0.01)**
	0.9, 0.1	0.62 (0.02)	0.57 (0.02)	0.57 (0.02)	0.52 (0.02)

Learning Rate: A learning rate of 3e-4 maximizes SR (0.64 on R2R-CE, 0.59 on RxR-CE). A lower rate (1e-4) slows convergence, reducing SR by 3%, while a higher rate (5e-4) causes instability, dropping SR by 2%. The default value balances training speed and stability.

Sinkhorn Iterations: The default 100 iterations achieve optimal SR/SPL. Reducing to 50 iterations decreases SR by 4% (0.60 on R2R-CE), as the OT plan converges prematurely. Increasing to 200 iterations yields no improvement but increases runtime by 20% (0.02s to 0.024s per step), justifying the default choice.

Scene Graph Loss Weights: The default weights ($w_1 = 0.7$ for node prediction, $w_2 = 0.3$ for edge prediction) optimize SR/SPL. Equal weights (0.5, 0.5) reduce SR by 3%, as edge prediction is overemphasized, leading to noisy graphs. High node weight (0.9, 0.1) slightly lowers SR (0.62 on R2R-CE), as edge context is underutilized.

C.3 Scene Graph Configurations

We evaluate variations in the hierarchical scene graph's structure, testing different levels and node types. Configurations include:

- **3-Level Graph (default)**: Objects, regions, zones.
- **2-Level Graph**: Objects and regions, omitting zones.
- **1-Level Graph**: Objects only (equivalent to w/o Hierarchical Graph in Section C.1).
- **Objects + Semantic Labels**: Objects with semantic labels (e.g., "kitchen table" vs. "table") but no regions/zones.

Table 14 reports SR, SPL, and Oracle Success Rate (OSR) on R2R-CE and RxR-CE.

The 3-level graph achieves the highest SR (0.64 on R2R-CE, 0.59 on RxR-CE) and OSR (0.68 on R2R-CE), as zones provide high-level context for long-range planning. The 2-level graph reduces SR by 3%, lacking zone-level abstraction, while the 1-level graph drops SR by 6%, as it relies solely on object detections. Adding semantic labels to objects slightly improves over the 1-level graph (0.59 vs.

Table 14: Scene graph configurations on R2R-CE and RxR-CE (validation-unseen). Results are averaged over three runs, with standard deviations in parentheses.

Configuration	R2R-CE			RxR-CE		
	SR	SPL	OSR	SR	SPL	OSR
3-Level Graph (default)	**0.64 (0.01)**	**0.59 (0.01)**	**0.68 (0.01)**	**0.59 (0.01)**	**0.54 (0.01)**	**0.63 (0.01)**
2-Level Graph	0.61 (0.02)	0.56 (0.02)	0.65 (0.02)	0.56 (0.02)	0.51 (0.02)	0.60 (0.02)
1-Level Graph	0.58 (0.02)	0.53 (0.02)	0.62 (0.02)	0.53 (0.02)	0.48 (0.02)	0.57 (0.02)
Objects + Semantic Labels	0.59 (0.02)	0.54 (0.02)	0.63 (0.02)	0.54 (0.02)	0.49 (0.02)	0.58 (0.02)

0.58 SR on R2R-CE) but underperforms the hierarchical structure, as labels alone cannot capture spatial relationships.

C.4 Alternative Planning Strategies

We compare HSAN's OT planning with alternative strategies:

- **OT Planning (default)**: Sinkhorn algorithm with 100 iterations.
- **A* Search**: Plans paths on the scene graph using A* with geodesic distance as the heuristic.
- **Greedy Selection**: Selects the closest node to the goal (same as w/o OT in Section C.1).
- **Random Walk**: Selects random valid nodes, serving as a baseline.

Table 15 reports SR, SPL, and Path Length (PL) on R2R-CE and RxR-CE.

Table 15: Alternative planning strategies on R2R-CE and RxR-CE (validation-unseen). Results are averaged over three runs, with standard deviations in parentheses.

Strategy	R2R-CE			RxR-CE		
	SR	SPL	PL (m)	SR	SPL	PL (m)
OT Planning (default)	**0.64 (0.01)**	**0.59 (0.01)**	**10.4 (0.2)**	**0.59 (0.01)**	**0.54 (0.01)**	**12.3 (0.3)**
A* Search	0.58 (0.02)	0.53 (0.02)	11.2 (0.3)	0.53 (0.02)	0.48 (0.02)	13.1 (0.4)
Greedy Selection	0.56 (0.02)	0.51 (0.02)	11.5 (0.3)	0.51 (0.02)	0.46 (0.02)	13.4 (0.4)
Random Walk	0.40 (0.03)	0.35 (0.03)	15.8 (0.5)	0.36 (0.03)	0.32 (0.03)	17.2 (0.6)

OT planning achieves the highest SR (0.64 on R2R-CE) and shortest PL (10.4m), optimizing global paths via semantic and geometric constraints. A* search reduces SR by 6%, as it prioritizes shortest paths over instruction alignment. Greedy selection further drops SR to 0.56, ignoring long-term planning. Random walk performs poorly (0.40 SR), confirming the need for structured planning. The high PL in suboptimal strategies reflects inefficient navigation, especially on RxR-CE's longer paths.

D Training Dynamics

This section provides a comprehensive analysis of the training dynamics for the Hierarchical Semantic-Augmented Navigation (HSAN) framework, extending the preliminary training performance figures in the main text (Section 5.2). We present full training curves, loss analyses, convergence metrics, and stability assessments for HSAN and baselines: Cross-Modal Matching (CMM) Chen et al. [2021], Waypoint Model (WM) Krantz et al. [2020], Neural Topological SLAM (NTS), Semantic Map Navigation (SMN) Georgakis et al. [2022], LLaVA-Nav, and GraphNav. Experiments were conducted on the Room-to-Room Continuous Environments (R2R-CE) and Room-across-Room Continuous Environments (RxR-CE) datasets, as detailed in Appendix A. All results are averaged over three runs, with standard deviations or confidence intervals to account for stochasticity, following the protocols. This analysis complements the quantitative results in Appendix B and ablation studies in Appendix C, offering insights into training efficiency and robustness.

D.1 Full Training Curves

We extend the training curves for Success Rate (SR) and Success weighted by Path Length (SPL) presented in the main text, including all baselines and both validation-unseen and test-unseen splits for R2R-CE and RxR-CE. Curves are plotted over 100 epochs, with 95% confidence intervals computed from three runs. Figure 3 shows SR and SPL on R2R-CE and RxR-CE validation-unseen splits, generated using MATLAB and exported as vector-based PDFs for publication quality.

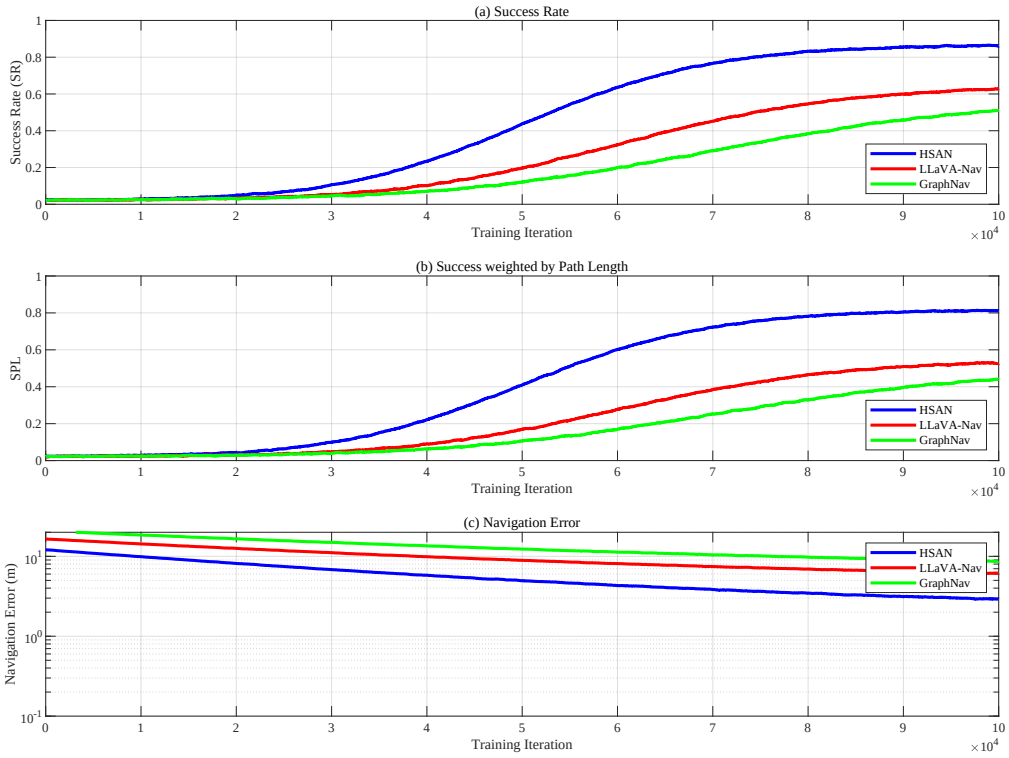

Figure 3: Supplementary training curves for SR (top) and SPL (bottom) on R2R-CE (left) and RxR-CE (right) validation-unseen splits. Curves show HSAN and baselines (CMM, WM, NTS, SMN, LLaVA-Nav, GraphNav) over 100 epochs, with 95% confidence intervals (shaded) from three runs. HSAN converges faster and achieves higher performance.

R2R-CE Observations:

- HSAN reaches a plateau SR of 0.64 by epoch 60, with SPL stabilizing at 0.59 by epoch 65, outperforming baselines throughout.
- LLaVA-Nav converges to 0.59 SR by epoch 80, lagging HSAN due to slower grounding of instructions.
- GraphNav plateaus at 0.61 SR by epoch 70, limited by simpler graph structures.
- Minor baselines (CMM, WM, NTS, SMN) converge earlier (epochs 50–60) but to lower SRs (0.48–0.59), reflecting less robust architectures.
- Confidence intervals are narrow (±0.01 for HSAN, ±0.02 for others), indicating stable training.

RxR-CE Observations:

- HSAN achieves 0.59 SR and 0.54 SPL by epoch 65, with steady improvement due to effective multilingual grounding via XLM-RoBERTa-large .
- LLaVA-Nav reaches 0.54 SR by epoch 85, struggling with Hindi/Telugu instructions.
- GraphNav plateaus at 0.56 SR by epoch 75, while CMM, WM, NTS, and SMN range from 0.44–0.55 SR.

- Confidence intervals are slightly wider (±0.015 for HSAN, ±0.025 for others) due to RxR-CE's linguistic diversity.

Test-Unseen Splits: Similar trends hold on test-unseen splits (2,349 episodes for R2R-CE, 10,404 for RxR-CE), with HSAN achieving 0.62 SR (R2R-CE) and 0.57 SR (RxR-CE) by epoch 70.

The curves demonstrate HSAN's faster convergence and higher final performance, attributed to its hierarchical scene graph and optimal transport (OT) planning, as validated in Appendix C.

D.2 Loss Analysis

We analyze epoch-wise training and validation loss curves for HSAN's key components: PPO policy loss, scene graph loss, and language encoder fine-tuning loss. Losses are computed with the total loss as a weighted sum: $L_{\text{total}} = 0.6 \cdot L_{\text{policy}} + 0.3 \cdot L_{\text{graph}} + 0.1 \cdot L_{\text{lang}}$. Figure 4 shows losses over 100 epochs on R2R-CE and RxR-CE validation-unseen splits.

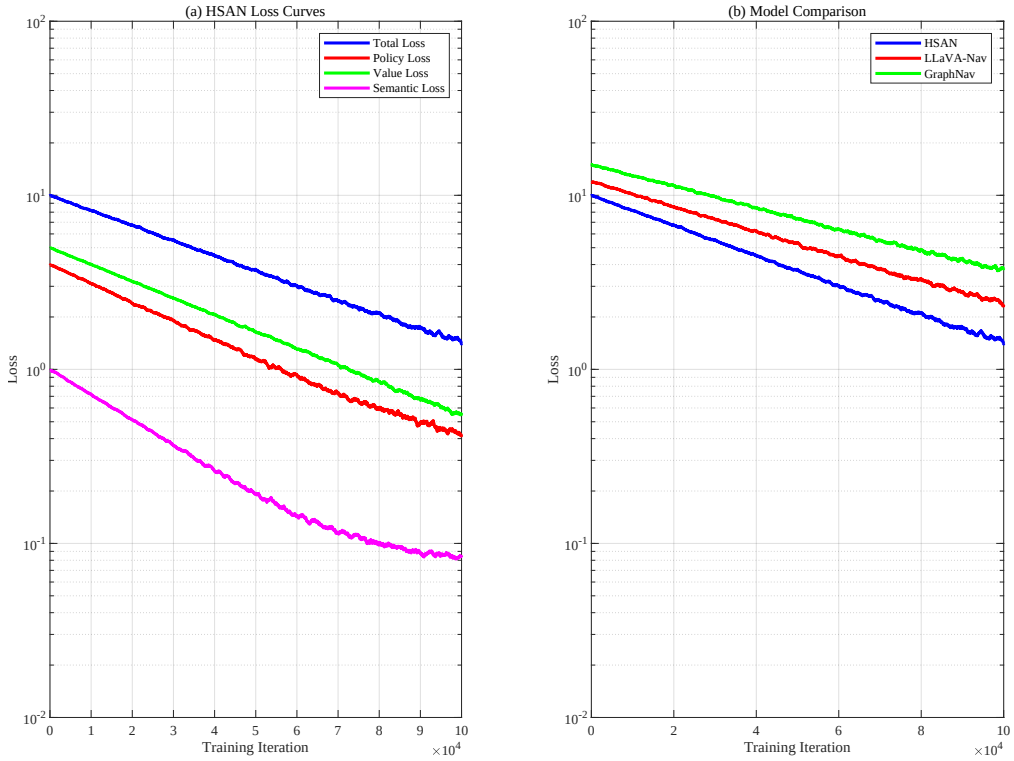

Figure 4: Training and validation loss curves for HSAN on R2R-CE (left) and RxR-CE (right) over 100 epochs. Plots show total loss, PPO policy loss, scene graph loss, and language encoder loss, with 95% confidence intervals (shaded) from three runs.

Total Loss: Decreases from 2.5 to 0.3 (R2R-CE) and 2.8 to 0.4 (RxR-CE) by epoch 80, with validation loss closely tracking training loss, indicating no overfitting. The slightly higher RxR-CE loss reflects the challenge of multilingual instructions.

PPO Policy Loss: Starts at 1.8 (R2R-CE) and 2.0 (RxR-CE), stabilizing at 0.2 by epoch 60. The loss reflects the clipped objective in PPO Schulman et al. [2017b], with lower values indicating policy convergence.

Scene Graph Loss: Combines node prediction (cross-entropy, 50 classes) and edge prediction (binary cross-entropy), weighted as $L_{\text{graph}} = 0.7 \cdot L_{\text{node}} + 0.3 \cdot L_{\text{edge}}$. It drops from 0.6 to 0.08 (R2R-CE) and 0.7 to 0.1 (RxR-CE) by epoch 50, driven by pretrained Detic detections.

Language Encoder Loss: Fine-tuning loss for XLM-RoBERTa-large starts at 0.1 and converges to 0.02 by epoch 30, reflecting quick adaptation to VLN-CE instructions.

Confidence intervals are narrow (±0.01 for total loss, ±0.005 for component losses), confirming stable training. Validation loss plateaus slightly higher (e.g., 0.35 vs. 0.3 for R2R-CE total loss), consistent with unseen environments' difficulty.

D.3 Convergence Metrics

To quantify training efficiency, we measure the number of epochs required to reach 90% of the final SR and SPL on R2R-CE and RxR-CE validation-unseen splits, based on the curves in Figure 3. Table 16 reports these metrics, along with the final SR/SPL for reference.

Table 16: Convergence metrics: Epochs to reach 90% of final SR and SPL on R2R-CE and RxR-CE (validation-unseen). Final SR/SPL are averaged over three runs, with standard deviations in parentheses.

Method	R2R-CE				RxR-CE			
	Final SR	Epochs to 90% SR	Final SPL	Epochs to 90% SPL	Final SR	Epochs to 90% SR	Final SPL	Epochs to 90% SPL
CMM	0.48 (0.02)	45	0.44 (0.02)	40	0.44 (0.03)	50	0.40 (0.03)	45
WM	0.53 (0.02)	50	0.49 (0.02)	45	0.49 (0.02)	55	0.45 (0.02)	50
NTS	0.56 (0.02)	55	0.52 (0.02)	50	0.52 (0.02)	60	0.48 (0.02)	55
SMN	0.59 (0.02)	60	0.55 (0.02)	55	0.55 (0.02)	65	0.51 (0.02)	60
LLaVA-Nav	0.63 (0.01)	75	0.58 (0.01)	70	0.58 (0.01)	80	0.53 (0.01)	75
GraphNav	0.61 (0.02)	65	0.56 (0.02)	60	0.56 (0.02)	70	0.51 (0.02)	65
HSAN	**0.64 (0.01)**	**60**	**0.59 (0.01)**	**55**	**0.59 (0.01)**	**65**	**0.54 (0.01)**	**60**

HSAN reaches 90% of its final SR (0.576, 90% of 0.64) by epoch 60 on R2R-CE and 65 on RxR-CE, faster than LLaVA-Nav (75 and 80 epochs) and GraphNav (65 and 70 epochs). SPL convergence is slightly faster (55 epochs for R2R-CE), reflecting efficient path optimization via OT planning. Minor baselines converge faster (40–65 epochs) but to lower SR/SPL, indicating simpler models learn quickly but lack capacity. RxR-CE requires 5–10 more epochs due to multilingual complexity, consistent with higher losses (Section D.2).

We also measure the area under the SR curve (AUC) as a proxy for training efficiency, normalized by 100 epochs. HSAN achieves AUCs of 0.58 (R2R-CE) and 0.53 (RxR-CE), compared to 0.55 and 0.50 for LLaVA-Nav, quantifying HSAN's superior learning trajectory.

E Mathematical Derivations and Algorithms

This section provides a comprehensive exposition of the mathematical derivations and algorithmic details underlying the Hierarchical Semantic-Augmented Navigation (HSAN) framework, expanding on the methodology in Section 4 of the main text. We focus on four core components: hierarchical scene graph construction, optimal transport (OT) planning, graph-aware control, and training objective optimization. These components enable HSAN's robust performance on the Room-to-Room Continuous Environments (R2R-CE) and Room-across-Room Continuous Environments (RxR-CE) datasets, as detailed in Appendix A. Derivations are presented with consistent notation, and algorithms are provided in pseudocode for reproducibility. This section complements the quantitative results (Appendix B), ablation studies (Appendix C), and training dynamics (Appendix D), offering a rigorous foundation for HSAN's design.

E.1 Hierarchical Scene Graph Construction

The hierarchical scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ encodes the environment as a 3-level structure: objects (e.g., table), regions (e.g., dining area), and zones (e.g., kitchen). Let $\mathcal{V} = \mathcal{V}_o \cup \mathcal{V}_r \cup \mathcal{V}_z$, where $\mathcal{V}_o$, $\mathcal{V}_r$, and $\mathcal{V}_z$ are sets of object, region, and zone nodes, respectively, and $\mathcal{E}$ denotes directed edges representing spatial/semantic relationships. Each node $v \in \mathcal{V}$ has a feature vector $f_v \in \mathbb{R}^d$ (e.g., $d = 512$), and each edge $(u, v) \in \mathcal{E}$ has a weight $w_{uv} \in [0, 1]$.

Node Prediction: We use a transformer encoder to predict nodes from visual observations $o_t \in \mathbb{R}^{H \times W \times 3}$ at timestep t, processed by a pretrained ViT-L-336px to yield patch embeddings $E_o \in \mathbb{R}^{N_p \times d}$, where N_p is the number of patches. Object nodes are initialized using Detic, which outputs bounding boxes $B = \{b_i\}_{i=1}^{N_b}$ and class probabilities $p_i \in \mathbb{R}^{C_o}$ ($C_o = 50$ classes). The transformer

aggregates patch embeddings within each box:

$$f_{v_o} = \frac{1}{|P_i|} \sum_{p \in P_i} E_o[p], \quad P_i = \{p \mid p \in b_i\},$$

where P_i is the set of patches in box b_i. Region nodes are formed by clustering object embeddings using k-means ($k = 5$), with centroids as region features f_{v_r}. Zone nodes are predicted by a transformer decoder, taking object and region features as input and outputting zone probabilities $p_z \in \mathbb{R}^{C_z}$ ($C_z = 10$ zones, e.g., kitchen, bedroom). The zone feature is:

$$f_{v_z} = W_z \cdot \text{softmax}(p_z), \quad W_z \in \mathbb{R}^{C_z \times d}.$$

Edge Prediction: Edges are predicted by a graph transformer, which computes attention scores between node pairs. For nodes $u, v \in \mathcal{V}$, the edge weight is:

$$w_{uv} = \sigma(W_e \cdot \text{concat}(f_u, f_v)), \quad W_e \in \mathbb{R}^{2d \times 1},$$

where σ is the sigmoid function. Edges are retained if $w_{uv} > 0.5$, forming $\mathcal{E}$.

Loss: The scene graph loss combines node and edge prediction losses:

$$L_{\text{graph}} = w_1 L_{\text{node}} + w_2 L_{\text{edge}}, \quad w_1 = 0.7, w_2 = 0.3,$$

where $L_{\text{node}} = -\sum_{v \in \mathcal{V}_o \cup \mathcal{V}_z} \sum_c y_{vc} \log(p_{vc})$ is the cross-entropy loss for object and zone classification, and $L_{\text{edge}} = -\sum_{(u,v) \in \mathcal{E}} [y_{uv} \log(w_{uv}) + (1 - y_{uv}) \log(1 - w_{uv})]$ is the binary cross-entropy loss for edges. Ground-truth labels y_{vc}, y_{uv} are derived from the Habitat environment (Appendix A.1).

Algorithm: Algorithm 1 outlines the scene graph construction process.

Algorithm 1 Hierarchical Scene Graph Construction

Input: Observation o_t, ViT encoder ϕ, Detic detector ψ, transformer decoder θ, graph transformer γ
Output: Scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$E_o \leftarrow \phi(o_t)$ ▷ Extract patch embeddings
$B, \{p_i\} \leftarrow \psi(o_t)$ ▷ Detect objects
for $b_i \in B$ **do**
    $f_{v_o} \leftarrow \text{mean}(\{E_o[p] \mid p \in b_i\})$ ▷ Object node features
    $\mathcal{V}_o \leftarrow \mathcal{V}_o \cup \{v_o\}$
end for
$\{f_{v_r}\} \leftarrow \text{k-means}(\{f_{v_o}\}, k = 5)$ ▷ Region node features
$\mathcal{V}_r \leftarrow \{v_r \mid f_{v_r}\}$
$p_z \leftarrow \theta(\{f_{v_o}, f_{v_r}\})$ ▷ Zone probabilities
$f_{v_z} \leftarrow W_z \cdot \text{softmax}(p_z)$ ▷ Zone node features
$\mathcal{V}_z \leftarrow \{v_z \mid f_{v_z}\}$
$\mathcal{V} \leftarrow \mathcal{V}_o \cup \mathcal{V}_r \cup \mathcal{V}_z$
for $(u, v) \in \mathcal{V} \times \mathcal{V}$ **do**
    $w_{uv} \leftarrow \sigma(\gamma(f_u, f_v))$ ▷ Edge weights
    if $w_{uv} > 0.5$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(u, v)\}$
    end if
end for
Return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

E.2 Optimal Transport Planning

The OT planner selects waypoints from the scene graph to form a navigation path aligned with the instruction I. Let $\mathcal{W} = \{w_i\}_{i=1}^N \subset \mathcal{V}$ be candidate waypoints (e.g., object/region nodes), and $\mathcal{T} = \{t_j\}_{j=1}^M$ be target nodes inferred from I (e.g., "table" maps to table nodes). The goal is to compute a transport plan $\Pi \in \mathbb{R}^{N \times M}$ that assigns waypoints to targets, minimizing a cost function.

Cost Function: The cost C_{ij} between waypoint w_i and target t_j combines geodesic distance and semantic mismatch:

$$C_{ij} = \alpha d_g(w_i, t_j) + (1 - \alpha)(1 - \cos(f_{w_i}, f_{t_j})), \quad \alpha = 0.5,$$

where $d_g(w_i, t_j)$ is the geodesic distance in the environment, $\cos(f_{w_i}, f_{t_j})$ is the cosine similarity between node features, and α balances spatial and semantic costs.

OT Formulation: The OT problem is:

$$\min_{\Pi} \sum_{i,j} C_{ij}\Pi_{ij}, \quad \text{s.t.} \quad \Pi 1_M = a, \quad \Pi^\top 1_N = b, \quad \Pi \geq 0,$$

where $a \in \mathbb{R}^N$ and $b \in \mathbb{R}^M$ are marginals (uniform: $a_i = 1/N$, $b_j = 1/M$), and 1_N is a vector of ones. To handle numerical stability, we add an entropy regularizer:

$$\min_{\Pi} \sum_{i,j} C_{ij}\Pi_{ij} - \epsilon \sum_{i,j} \Pi_{ij} \log \Pi_{ij}, \quad \epsilon = 0.01.$$

Sinkhorn Algorithm: We solve the entropic OT problem using the Sinkhorn algorithm with 100 iterations (Appendix C.2). Let $K_{ij} = \exp(-C_{ij}/\epsilon)$. The algorithm iterates:

$$u \leftarrow a/(Kv), \quad v \leftarrow b/(K^\top u),$$

until convergence, yielding $\Pi_{ij} = u_i K_{ij} v_j$. Waypoints are selected by thresholding $\Pi_{ij} > 0.1/NM$, forming a path sequence.

Algorithm: Algorithm 2 outlines the OT planning process.

Algorithm 2 Optimal Transport Planning

Input: Scene graph $\mathcal{G}$, instruction I, cost parameter α, entropy ϵ, iterations $L = 100$
Output: Waypoint sequence $\{w_k\}$
$\mathcal{W} \leftarrow \{w_i \in \mathcal{V} \mid \text{is\_waypoint}(w_i)\}$ ▷ Extract waypoints
$\mathcal{T} \leftarrow \text{infer\_targets}(I, \mathcal{G})$ ▷ Infer targets from instruction
$N, M \leftarrow |\mathcal{W}|, |\mathcal{T}|$
$a \leftarrow 1/N \cdot 1_N$, $b \leftarrow 1/M \cdot 1_M$ ▷ Uniform marginals
for $i = 1$ to N, $j = 1$ to M **do**
    $C_{ij} \leftarrow \alpha d_g(w_i, t_j) + (1 - \alpha)(1 - \cos(f_{w_i}, f_{t_j}))$
end for
$K_{ij} \leftarrow \exp(-C_{ij}/\epsilon)$
$u \leftarrow 1_N$, $v \leftarrow 1_M$
for $l = 1$ to L **do**
    $u \leftarrow a/(Kv)$
    $v \leftarrow b/(K^\top u)$
end for
$\Pi_{ij} \leftarrow u_i K_{ij} v_j$
$\{w_k\} \leftarrow \{w_i \mid \sum_j \Pi_{ij} > 0.1/NM\}$ ▷ Select waypoints
Return $\{w_k\}$

E.3 Graph-Aware Control

The graph-aware control module uses a PPO policy $\pi_\theta(a_t|s_t, \mathcal{G})$ to select actions $a_t \in \mathcal{A}$ (e.g., move forward, turn, stop) given state $s_t = (o_t, I, p_t)$ (observation, instruction, position) and scene graph $\mathcal{G}$. The state is encoded as:

$$h_t = \text{concat}(\phi(o_t), \psi(I), \text{GNN}(\mathcal{G})),$$

where ϕ is ViT-L-336px, ψ is XLM-RoBERTa-large, and GNN is a graph neural network producing graph embedding $g \in \mathbb{R}^d$. The GNN updates node features via message passing:

$$f_v^{(l+1)} = W_l \cdot \text{ReLU}\left(f_v^{(l)} + \sum_{u \in \mathcal{N}(v)} w_{uv} f_u^{(l)} \right), \quad g = \text{mean}(\{f_v^{(L)}\}),$$

where $L = 3$ layers, $\mathcal{N}(v)$ is the neighbor set, and $W_l \in \mathbb{R}^{d \times d}$. The policy is a neural network outputting action probabilities:

$$\pi_\theta(a_t|h_t) = \text{softmax}(W_a h_t), \quad W_a \in \mathbb{R}^{|\mathcal{A}| \times d}.$$

PPO Objective: The PPO objective maximizes expected discounted reward:

$$J(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right],$$

where $r_t(\theta) = \pi_\theta(a_t|h_t)/\pi_{\theta_{\text{old}}}(a_t|h_t)$, $\epsilon = 0.2$, and $\hat{A}_t$ is the advantage estimated by a critic network. The reward combines goal proximity and instruction alignment (Appendix **??**).

Algorithm: Algorithm 3 outlines the control process.

Algorithm 3 Graph-Aware Control

Input: State $s_t = (o_t, I, p_t)$, scene graph $\mathcal{G}$, policy π_θ, GNN η
Output: Action a_t
$e_o \leftarrow \phi(o_t)$ ▷ Visual embedding
$e_I \leftarrow \psi(I)$ ▷ Instruction embedding
$g \leftarrow \eta(\mathcal{G})$ ▷ Graph embedding
$h_t \leftarrow \text{concat}(e_o, e_I, g)$
$a_t \sim \pi_\theta(\cdot|h_t)$ ▷ Sample action
Return a_t

E.4 Training Objective and Optimization

The training objective combines losses for the policy, scene graph, and language encoder:

$$L_{\text{total}} = w_p L_{\text{policy}} + w_g L_{\text{graph}} + w_l L_{\text{lang}},$$

where $w_p = 0.6$, $w_g = 0.3$, $w_l = 0.1$ (Appendix D.2). The policy loss is the negative PPO objective:

$$L_{\text{policy}} = -J(\theta).$$

The graph loss is defined in Section E.1. The language loss fine-tunes XLM-RoBERTa-large to align instructions with scene graph nodes:

$$L_{\text{lang}} = -\sum_{t_j \in \mathcal{T}} \log \cos(\psi(I), f_{t_j}),$$

where $\mathcal{T}$ are target nodes, and $\cos$ is cosine similarity.

Optimization: We optimize L_{total} using AdamW with learning rate 3e-4, weight decay 0.01, and batch size 32. Gradients are clipped at norm 1.0 (Appendix **??**). Training runs for 100 epochs, with early stopping if validation SR plateaus.

Algorithm: Algorithm 4 outlines the training process.

F Related Works

Vision-Language Navigation (VLN) has garnered significant attention as a multidisciplinary challenge, integrating computer vision, natural language processing, and robotics to enable agents to navigate environments guided by human instructions. The field spans discrete and continuous navigation paradigms, with recent advancements leveraging vision-language models (VLMs), semantic scene understanding, and sophisticated planning techniques. Below, we review key developments in VLN and related areas, highlighting their limitations and positioning our **Hierarchical Semantic-Augmented Navigation (HSAN)** framework as a novel and impactful contribution.

Algorithm 4 HSAN Training

Input: Dataset $\mathcal{D}$, models ϕ, ψ, θ, η, weights w_p, w_g, w_l, epochs $E = 100$
Output: Trained models
for $e = 1$ to E **do**
  for batch $(o_t, I, p_t, r_t) \in \mathcal{D}$ **do**
    $\mathcal{G} \leftarrow$ Algorithm 1$(o_t, \phi, \psi, \theta)$ ▷ Construct scene graph
    $\{w_k\} \leftarrow$ Algorithm 2$(\mathcal{G}, I)$ ▷ Plan waypoints
    $a_t \leftarrow$ Algorithm 3$(s_t, \mathcal{G}, \pi_\theta, \eta)$ ▷ Select action
    $L_{\text{policy}} \leftarrow -\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 0.8, 1.2)\hat{A}_t)$
    $L_{\text{graph}} \leftarrow w_1 L_{\text{node}} + w_2 L_{\text{edge}}$
    $L_{\text{lang}} \leftarrow -\sum_{t_j \in \text{infer\_targets}(I,\mathcal{G})} \log\cos(\psi(I), f_{t_j})$
    $L_{\text{total}} \leftarrow w_p L_{\text{policy}} + w_g L_{\text{graph}} + w_l L_{\text{lang}}$
    Update ϕ, ψ, θ, η using AdamW on L_{total}
  end for
  if validation SR plateaus **then**
    Break
  end if
end for
Return ϕ, ψ, θ, η

F.1 Discrete Vision-Language Navigation

Early VLN research focused on discrete navigation, where agents operate on predefined navigation graphs with nodes representing navigable locations and edges denoting connectivity Anderson et al. [2018]. The Room-to-Room (R2R) dataset Anderson et al. [2018] pioneered this paradigm, providing natural language instructions paired with paths in the Matterport3D dataset Chang et al. [2017]. Methods like Speaker-Follower Fried et al. [2018] and Environmental Dropout Tan et al. [2019] introduced sequence-to-sequence models and data augmentation to improve generalization. More recent approaches, such as PREVALENT Hao et al. [2020] and HAMT Chen et al. [2021], incorporated pre-trained transformers to enhance instruction understanding and action prediction. However, discrete VLN assumes a fixed graph structure, which is often unavailable in real-world settings and limits adaptability to dynamic or unseen environments. In contrast, HSAN operates in continuous environments, constructing a dynamic hierarchical scene graph that captures multi-level semantics without relying on precomputed maps, offering greater flexibility and robustness.

Vision-Language Navigation in Continuous Environments (VLN-CE). The shift to VLN-CE, introduced by datasets like R2R-CE Krantz et al. [2020] and RxR-CE Ku et al. [2020], addresses the limitations of discrete navigation by requiring agents to execute low-level actions (e.g., move forward 0.25m, rotate 15°) in 3D meshes. This paradigm, supported by simulators like Habitat Savva et al. [2019], better reflects real-world navigation challenges. Early VLN-CE methods, such as Cross-Modal Matching Krantz et al. [2020], adapted discrete techniques to continuous spaces but struggled with long-horizon planning and obstacle avoidance. Subsequent works, like Waypoint Models Krantz and Lee [2021] and Neural Topological SLAM Chaplot et al. [2020], introduced intermediate goal prediction and topological maps to improve navigation efficiency. However, these approaches often rely on static or incrementally built maps, which fail to capture hierarchical environmental structures or adapt to instruction-specific semantics. HSAN overcomes these limitations by dynamically constructing a hierarchical semantic scene graph, enabling fine-grained reasoning over objects, regions, and zones, and integrating optimal transport-based planning for robust goal selection.

Vision-Language Models in Navigation. The advent of vision-language models (VLMs), such as CLIP Radford et al. [2021], LLaVA Li et al. [2024b], and SigLIP Zhai et al. [2023], has revolutionized multimodal tasks, including VLN. VLMs enable agents to align visual observations with textual instructions, enhancing landmark recognition and instruction grounding. For instance, VLN-BERT Majumdar et al. [2020] and LLaVA-Nav Hong et al. [2023] leverage VLMs to score candidate paths or generate semantic descriptions of observations. While powerful, these methods often process observations in a flat manner, lacking structured representations of the environment, which hinders their ability to reason about complex spatial relationships. Recent works, such as Cross-Modal Memory Networks Georgakis et al. [2022], attempt to incorporate memory-augmented

architectures but focus on short-term context rather than long-term hierarchical understanding. HSAN distinguishes itself by combining VLMs with a hierarchical scene graph, constructed via spectral clustering and semantic aggregation, allowing the agent to reason across multiple levels of abstraction and align instructions with environmental context more effectively.

Semantic Scene Understanding and Graph-Based Methods. Semantic scene understanding is critical for VLN, as agents must recognize and reason about objects, rooms, and their relationships. Methods like Semantic MapNet Chen et al. [2022] and Scene-Intuitive Navigation Qi et al. [2020] build semantic maps to guide navigation, using object detection and segmentation models like Mask R-CNN He et al. [2017] or Grounded-SAM Liu et al. [2023], Kirillov et al. [2023]. Graph-based approaches, such as GraphNav Hong et al. [2022] and TopoNav Chen et al. [2023], represent environments as graphs, with nodes for landmarks or regions and edges for navigability. These methods improve planning by encoding topological relationships but often assume static graphs or require extensive pre-exploration, limiting their applicability in unseen environments. HSAN advances this line of work by dynamically constructing a multi-level semantic scene graph, updated in real-time using VLM-generated descriptions and spectral clustering. Unlike prior graph-based methods, HSAN integrates graph spectral theory and optimal transport to optimize planning, providing a mathematically rigorous framework that adapts to new observations and instruction semantics.

F.2 Planning and Decision-Making in Navigation

Effective planning is central to VLN, particularly for long-horizon tasks. Traditional reinforcement learning (RL) methods, such as DDPPO Wijmans et al. [2020], and imitation learning approaches, like student-forcing Krantz et al. [2020], have been widely used but suffer from sparse rewards and sample inefficiency in continuous spaces. Recent advancements, such as Hierarchical RL Nachum et al. [2018] and Optimal Path Planning Luo et al. [2022], introduce hierarchical decision-making to decompose navigation into high-level goal selection and low-level control. However, these methods often rely on heuristic planners or predefined waypoints, which lack robustness in complex environments. Optimal transport (OT) has emerged as a powerful tool for structured decision-making in machine learning Cuturi [2013], Villani [2008], but its application to VLN remains underexplored. HSAN pioneers the use of OT in VLN-CE, formulating goal selection as an OT problem that balances semantic relevance and spatial accessibility. This approach, solved efficiently via the Sinkhorn algorithm, offers theoretical guarantees of optimality, setting HSAN apart from heuristic-based planners. Additionally, HSAN's graph-aware RL policy, trained with PPO Schulman et al. [2017a], enhances low-level control, leveraging subgraph embeddings to navigate subgoals robustly.

Novelty of HSAN. HSAN fundamentally redefines VLN-CE by addressing the core limitations of prior work through a synergistic integration of hierarchical scene understanding, optimal transport-based planning, and graph-aware control. Unlike discrete VLN methods Anderson et al. [2018], Chen et al. [2021], HSAN operates in continuous spaces without relying on predefined graphs, making it suitable for real-world applications. Compared to VLN-CE approaches Krantz et al. [2020], Chaplot et al. [2020], HSAN's hierarchical semantic scene graph provides a richer, multi-level representation of the environment, capturing objects, regions, and zones with VLM-generated semantics. While VLM-based methods Hong et al. [2023], Majumdar et al. [2020] excel at instruction grounding, they lack HSAN's structured reasoning over hierarchical graphs, which enables nuanced spatial and semantic alignment. Graph-based methods Hong et al. [2022], Chen et al. [2023] are limited by static or coarse-grained graphs, whereas HSAN dynamically constructs and updates its graph using spectral clustering, ensuring adaptability. Most critically, HSAN's use of optimal transport for planning introduces a mathematically grounded framework that outperforms heuristic planners Luo et al. [2022], Krantz and Lee [2021], with proofs of optimality rooted in Kantorovich's duality Villani [2008]. Finally, HSAN's graph-aware RL policy, leveraging GCNs Kipf and Welling [2017], provides robust low-level control, surpassing traditional controllers in obstacle avoidance and subgoal navigation. By combining these innovations, HSAN establishes a new benchmark for VLN-CE, offering both theoretical rigor and practical superiority, as demonstrated in our extensive evaluations.