

# Open Scene Graphs for Open-World Object-Goal Navigation

The International Journal of Robotics Research  
yy(zz):1–31  
©The Author(s) 2024  
Reprints and permission:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/ToBeAssigned  
www.sagepub.com/

SAGE

Joel Loo<sup>\*1</sup>, Zhanxin Wu<sup>\*2</sup> and David Hsu<sup>1</sup>

## Abstract

How can we build general-purpose robot systems for open-world semantic navigation, *e.g.*, searching a novel environment for a target object specified in natural language? To tackle this challenge, we introduce *OSG Navigator*, a modular system composed of foundation models, for open-world Object-Goal Navigation (ObjectNav). Foundation models provide enormous semantic knowledge about the world, but struggle to organise and maintain spatial information effectively at scale. Key to OSG Navigator is the *Open Scene Graph* representation, which acts as spatial memory for OSG Navigator. It organises spatial information hierarchically using *OSG schemas*, which are templates, each describing the common structure of a class of environments. OSG schemas can be automatically generated from simple semantic labels of a given environment, *e.g.*, “home” or “supermarket”. They enable OSG Navigator to adapt zero-shot to new environment types. We conducted experiments using both Fetch and Spot robots in simulation and in the real world, showing that OSG Navigator achieves state-of-the-art performance on ObjectNav benchmarks and generalises zero-shot over diverse goals, environments, and robot embodiments.

## Keywords

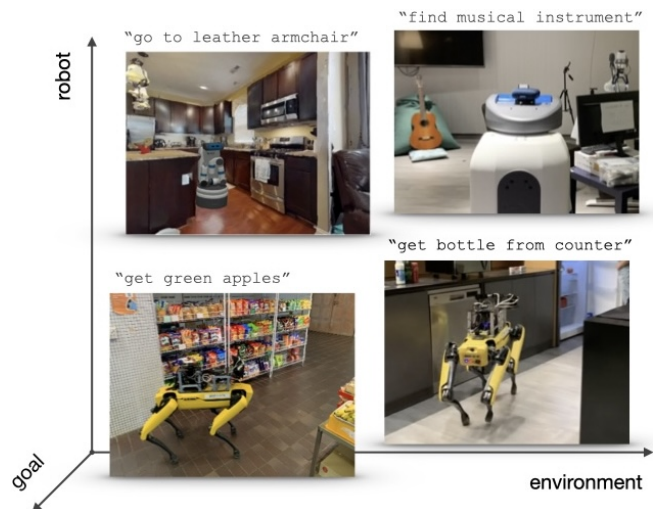
Object-goal navigation, scene graphs, robot foundation models

## 1 Introduction

Imagine a robot capable of taking verbal instructions from humans and quickly finding fever medicine in an unfamiliar pharmacy, finding a pack of cornflour in a new supermarket, or finding a bottle of wine in a friend’s home—all without a prior map of the environment. To do this, the robot must understand the rich semantics of the world, in addition to its geometry. This work focuses on *object-goal navigation* (ObjectNav), a semantic navigation task, in the *open world*. In ObjectNav, a robot navigates a *novel* environment to find a target object specified in natural language. We extend the standard ObjectNav task (Yadav et al. 2022a; Anderson et al. 2018) to the open-world setting with new challenges: the robot navigation system now needs to generalise over diverse goals, environments, and even robot embodiments (Figure 1):

- *Goals*. The robot navigates to potentially any object described in natural language, including novel ones not known *a priori*.
- *Environments*. The robot operates effectively in structurally different indoor environments—homes, offices, supermarkets, *etc.*—without environment-specific manual engineering.
- *Robots*. The same navigation system works on different robot embodiments, wheeled or legged.

To this end, we introduce *OSG Navigator*, a modular navigation system composed of foundation models for robot perception, state estimation, planning, and local navigation control (Figure 2). Foundation models offer a promising approach: *large language models* (LLMs) capture enormous



**Figure 1. ObjectNav in the open world.** Our experiments cover different combinations of goals, environments, and robots across both real world and simulation. We conducted experiments using both the Fetch and Spot robots in environments ranging from simulated homes (top-left) to real-world lab environments mimicking homes (top- and bottom-right) and stores (bottom-left).

<sup>1</sup> Smart Systems Institute, National University of Singapore

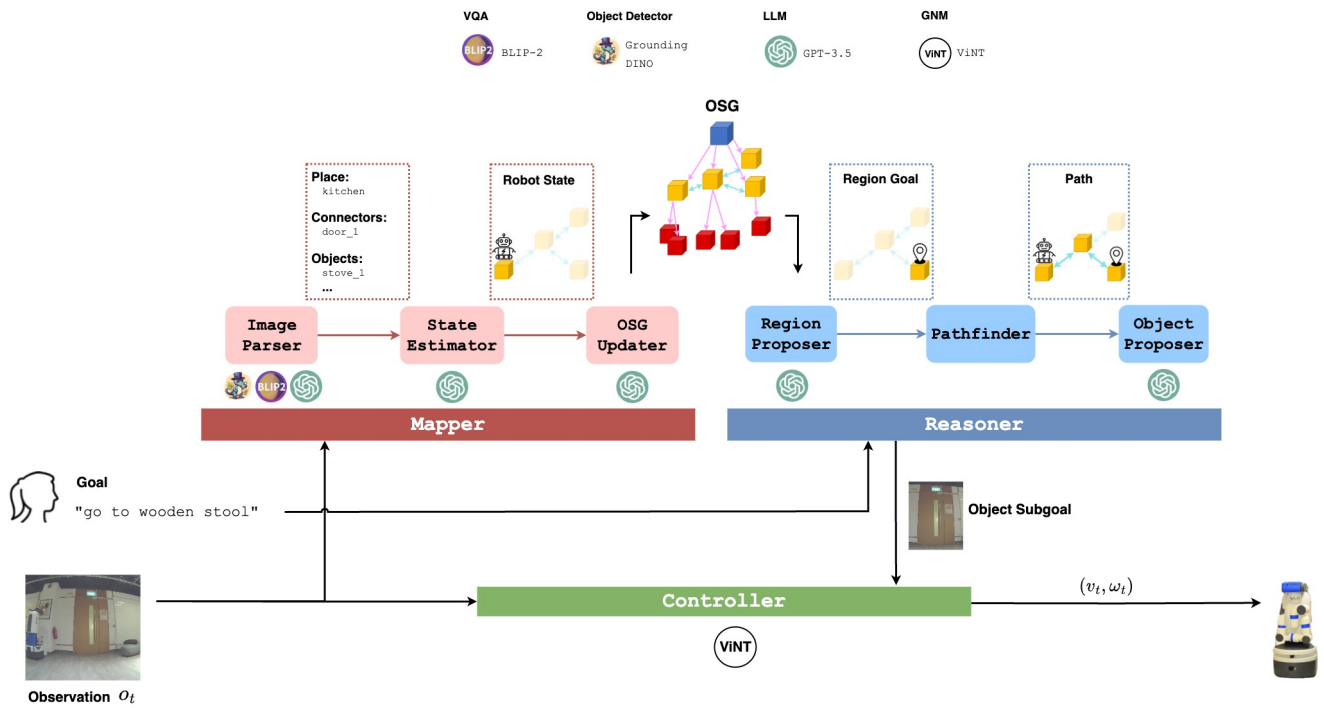
<sup>2</sup> College of Computing and Information Science, Cornell University

## Corresponding author:

David Hsu, Smart Systems Institute, National University of Singapore.

Email: dyhsu@comp.nus.edu.sg

\*Equal contribution.



**Figure 2. OSG Navigator.** OSG Navigator adopts a modular architecture, consisting of modules built from foundation models: VFMs, LLMs, and GNMs.

semantic knowledge about the world (Zhou et al. 2023; Chen et al. 2024; Yu et al. 2023; Shah et al. 2023a), while *vision foundation models* (VFMs) (Kirillov et al. 2023; Liu et al. 2023b; Li et al. 2023; Minderer et al. 2022) and *general navigation models* (GNMs) (Shah et al. 2023b; Sridhar et al. 2023) show strong performance in robot perception and action across diverse environments and embodiments, respectively. Specifically, OSG Navigator uses two VFMs, GroundingDINO (Liu et al. 2023b) and BLIP-2 (Li et al. 2023), to process perceptual information; it uses an LLM, GPT-3.5 (OpenAI et al. 2024), to organise and reason about spatial information; it uses a GNM, ViNT (Shah et al. 2023b), for low-level navigation control of robots. Finally, it composes these models in a neuro-symbolic architecture for open-world ObjectNav.

Foundation models are, however, insufficient by themselves, as they struggle to organise and maintain spatial information effectively at scale. A key element of OSG Navigator is the *Open Scene Graph* (OSG) representation (Figure 3). A scene graph represents an environment as a hierarchically organised graph, in which nodes represent semantic entities such as objects, places, regions, *etc.*, and edges represent relations among them. Scene graphs serve as the unified memory coordinating the various foundation models in OSG Navigator, by maintaining a coherent spatial understanding of the environment.

Traditional scene graphs are designed for specific environment types (Armeni et al. 2019; Rosinol et al. 2021; Werby et al. 2024). Each type has its own built-in semantic concepts: homes contain rooms and corridors, while supermarkets contain sections and aisles. This environment-specific design is impractical in the open world, as robots may encounter new environment types unknown in advance. To tackle this challenge, we introduce *OSG schemas*,

which are templates capturing the common structure among environments within a class (Figure 3). While schemas can be defined manually using domain-specific knowledge, we can again leverage foundation models, specifically, an LLM to generate schemas automatically, given simple semantic labels of indoor scenes, *e.g.*, “home” or “supermarket”. This enables OSG Navigator to adapt zero-shot to new types of environments.

Our key contribution is OSG Navigator, a neuro-symbolic system that combines foundation models and the OSG representation for open-world ObjectNav. OSG Navigator adopts a modular architecture, consisting entirely of foundation models as components, for semantic navigation in novel environments without prior maps. It leverages OSG and the associated schema for flexible, structured environment representation. We evaluated OSG Navigator in simulation and on two real robots, Fetch and Boston Dynamics Spot. Experimental results show that it achieves state-of-the-art performance on ObjectNav benchmarks and generalises zero-shot over diverse goals, environments, and robot embodiments. Further analyses point to the OSG representation as a key enabler of OSG Navigator’s strong performance. The experiments suggest the following main lessons:

- Integrating foundation models into the existing modular robot system architecture enables robot navigation that generalises zero-shot across diverse environments (Section 7.3), embodiments (Section 7.5), and open-vocabulary goals (Section 7.4).
- Foundation models, in particular, LLMs, benefit greatly from structured representations, such as OSG, in spatial planning (Section 7.3).

- Semantics offers a promising approach to generalisable robot navigation, independent of metric geometry (Sections 7.3–7.5, 7.6, and 7.10).

Several limitations currently remain in OSG Navigator. First, LLMs have made significant concurrent progress, and our current approach has yet to integrate the state of the art in multimodal models, multilingual models, or small, local models. Second, our approach has only been evaluated on indoor navigation tasks. Extending OSG Navigator to handle outdoor environments requires further investigation. Third, OSG Navigator adopts a topo-semantic approach and does not capture metric information and geometry explicitly. Metric geometry is complementary and will enrich the OSG representation and further improve navigation performance. In our view, this last one offers the most exciting research opportunity in the immediate future. See Section 8 for further discussion.

In the following, Section 2 reviews related work. Section 3 provides an overview of OSG Navigator. Section 4 describes the OSG representation in details. Sections 5 and 6 present the main modules of OSG Navigator. Section 7 presents the experimental results on open-world ObjectNav and evaluates the effectiveness of the OSG representation. Section 8 summarizes the main findings and points out directions for future research.

## 2 Related Work

### 2.1 Semantic Navigation

Robot navigation has expanded beyond tasks requiring geometric scene understanding and motion planning (Hutchinson et al. 1996) to encompass richer, more complex *semantic navigation* tasks (Yokoyama et al. 2024). These often specify abstract semantic goals, and require semantic reasoning and common-sense priors about environments to complete. Seminal examples include vision-and-language navigation (Gu et al. 2022), ImageNav (Krantz et al. 2022), and ObjectNav (Yadav et al. 2022a) which respectively require robots to follow natural language instructions, reach a goal specified by an image, and find instances of a given object category. Later works build on these, increasing their complexity, and need for semantic information and reasoning by extending them to zero-shot (Majumdar et al. 2022), open-vocabulary (Gadre et al. 2023), multi-goal (Wani et al. 2020), and interactive (Majumdar et al. 2023) settings. We show that a robot system composed from foundation models and Open Scene Graphs enables *semantic navigation* in *open world settings*, and demonstrate this on the task of open world ObjectNav (Figure 1).

### 2.2 Object-Goal Navigation

ObjectNav (Yadav et al. 2022a) is the task of searching for an instance of a specified object category in a novel unmapped indoor environment. The skill of moving to a specified object is a useful and general primitive relevant to many downstream tasks in navigation and mobile manipulation (Chang et al. 2023). As such, ObjectNav has drawn an extensive body of work (Wahid et al. 2021; Mousavian et al. 2019; Maksymets et al. 2021; Ye et al. 2021; Ramakrishnan et al. 2022), with notable progress made on

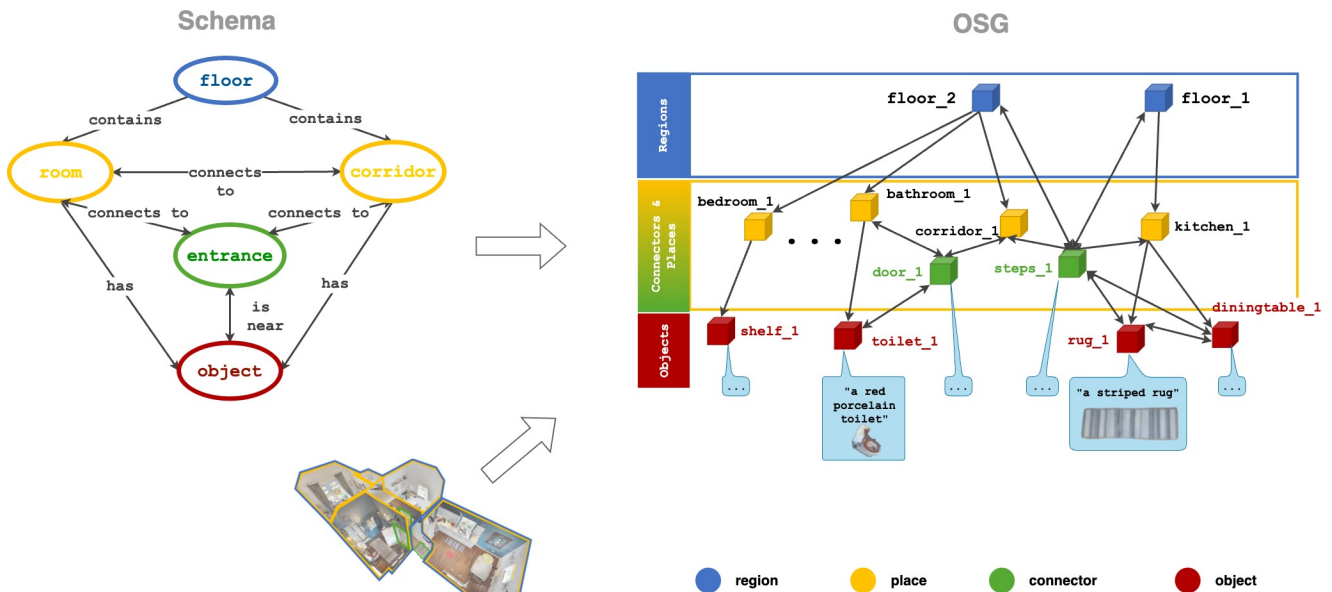
benchmarks over the years. However, as Majumdar et al. (2022) notes, these approaches typically require specific training on curated datasets for ObjectNav, limiting their ability to generalise to new scenes or goals. More broadly, the ObjectNav task is itself not fully *open-world*, as it predefines a small set of object categories as goals, and does not consider generalisation across embodiments.

Foundation models’ impressive zero-shot generalisation and open-world semantic knowledge (Hu et al. 2023) are driving extensions of the ObjectNav task to open-world scenarios, and enabling the design of generalisable ObjectNav agents. Majumdar et al. (2022); Al-Halah et al. (2022) propose zero-shot task variants like Zero-shot ObjectNav (ZSON), which expands the set of goals to open-vocabulary object categories and requires that agents not rely on ObjectNav-specific demonstrations or rewards. Gadre et al. (2023) extend this to more complex natural language queries in Language-driven ZSON (L-ZSON). Building on these, recent works explore designing agents to handle open-vocabulary object queries using LLMs (Shah et al. 2023a; Dorbala et al. 2024; Yu et al. 2023) or VLMs (Majumdar et al. 2022; Khandelwal et al. 2022; Gadre et al. 2023; Yokoyama et al. 2024) like CLIP (Radford et al. 2021). Our work goes further to address all three key dimensions of generalisation in the open world: diverse goals, environments and embodiments.

### 2.3 LLMs in Navigation

LLMs’ open-world semantic knowledge and reasoning capabilities are valuable for semantic navigation (Hu et al. 2023; Huang and Chang 2022). They are a queryable source of semantic priors, for instance, offering priors on object co-occurrence in different environments (Chen et al. 2023; Strader et al. 2024; Dorbala et al. 2024; Shah et al. 2023a). They are also increasingly used to handle abstract, semantic concepts and plan with them. For instance, LLMs can be queried to generate plans satisfying abstract language instructions (Ichter et al. 2023; Huang et al. 2022; Singh et al. 2023), or to reason semantically about scene structure in search or exploration (Zhou et al. 2023; Chen et al. 2024). Recent works show that LLMs can be integrated with formal methods for planning (Zhao et al. 2023; Huang and Zhang 2025), and for uncertainty handling (Wang et al. 2025) for more robust, structured reasoning. These capabilities make LLMs indispensable for semantic navigation tasks like ObjectNav, which require both strong semantic priors and planning.

How scene information is presented is a critical design choice in eliciting reasoning from LLMs (Liu et al. 2022; Strader et al. 2024). To overcome LLMs’ limited context lengths, an external memory is needed to temporally accumulate scene information. While some approaches prompt LLMs with unordered lists of objects (Cai et al. 2023; Yu et al. 2023; Shah et al. 2023a; Dorbala et al. 2024), incorporating spatial and semantic structure into the representation further enhances reasoning and reduces LLMs’ hallucinations (Rana et al. 2023; Honerkamp et al. 2024). Thus, a structured open-set scene representation is needed for semantic navigation with LLMs. Recent works apply LLMs to pre-built scene graphs for efficient semantic search (Rana et al. 2023), or to translate ambiguous semantic



**Figure 3. An example OSG and its schema.** Given a schema, OSG Navigator constructs an OSG incrementally from visual observations of the environment. An OSG schema captures the common structure among environments within a class, e.g., home.

tasks into formal grounded task specifications (Dai et al. 2024). Ravichandran et al. (2025); Honerkamp et al. (2024); Rajvanshi et al. (2023) apply LLMs to reason over graphical representations that are incrementally constructed, for under-specified, semantic navigation tasks.

OSG Navigator achieves open-world ObjectNav by using LLMs to both construct generalisable, structured OSG representations, and to reason with OSGs. This work focuses on designing the representation and associated mapping algorithm. We leave improvements in reasoning to future work, and note that ongoing advances in integrating LLMs with planning and formal techniques can complement our work.

## 2.4 Open-Set Scene Representations

Open-set scene representations are commonly geometric maps with open-set semantic features fused into them (Mascaro and Chli 2024). These include task-specific 2D costmaps (Yokoyama et al. 2024; Long et al. 2025; Ren et al. 2024), or 3D representations integrated with semantic embeddings—like point clouds (Jatavallabhula et al. 2023; Peng et al. 2023), NeRFs (Shen et al. 2023), and Gaussian splats (Wu et al. 2021; Qin et al. 2024). However, LLMs struggle to parse geometric maps and are better suited at understanding semantics. Retrieval-augmented generation (Lewis et al. 2020) offers vector databases as an alternative LLM-friendly representation (Anwar et al. 2024). Embodied-RAG (Xie et al. 2024) builds on this with a hierarchy of LLM-generated summaries for efficient indexing, though the lack of a schema to structure the hierarchy can lead to inconsistent abstractions.

Scene graphs are topological representations organising scenes in a hierarchy of spatial abstractions. As such they are suited for hierarchical reasoning for planning (Mascaro and Chli 2024; Amiri et al. 2022; Ravichandran et al. 2022; Honerkamp et al. 2024; Ray et al. 2024). In particular,

Rana et al. (2023); Rajvanshi et al. (2023) demonstrate that scene graphs’ symbolic and semantic nature facilitate task planning with LLMs. Recent progress has led to scene graphs that can be built incrementally (Wu et al. 2021; Hughes et al. 2022), and that capture open-set objects (Gu et al. 2023; Werby et al. 2024). However, a key weakness of most approaches, including Kimera (Rosinol et al. 2021) and HOVSG (Werby et al. 2024), is that they embed fixed schemas that only describe a specific environment class—often home/office environments comprised of rooms and floors. Recent works take steps toward enabling flexible schemas for mapping. Strader et al. (2024) proposes a learnable method to infer scene graph node labels that can be trained offline using only LLM guidance. Other approaches cluster geometric primitives based on a human-provided schema specifying open-vocabulary cluster labels—Lexis (Kassab et al. 2024) clusters rooms given a list of room labels, while Clio (Maggio et al. 2024) clusters objects or location nodes based on a list of tasks.

*Open Scene Graphs* offer an open-world, structured representation in terms of consistent spatial abstractions. This is enabled by OSG schemas which, unlike Lexis and Clio, support complex schemas capturing *hierarchies* of spatial abstractions.

## 3 System Overview

ObjectNav (Yadav et al. 2022b) is the task of searching for a specified object category in novel unmapped indoor scenes. The inputs to an ObjectNav agent are RGB, depth, and noiseless positioning, while the outputs are velocity commands. We extend this to the open-world setting, requiring the agent to perform ObjectNav zero-shot across diverse environments, embodiments and goals. Our more challenging task setting allows only for RGB input, and linear/angular velocity command outputs, which are common denominators across a diverse range of robots. Our

task setting is more realistic, as we do not assume access to privileged information like ground-truth localisation. The agent should also accept open-set object queries in natural language.

We formalise the task as an object-and-region-centric POMDP  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \Omega, O \rangle$ . This extends object-oriented POMDPs (OO-POMDPs) (Wandzel et al. 2019; Zheng et al. 2020, 2023) to incorporate regions and relations which provide context about scene structure valuable for object search (Grinvald et al. 2019; Chen et al. 2023). The state comprises agent state ( $s_a$ ), spatial regions ( $s_r$ ), objects ( $s_o$ ), and inter-object/region relations ( $s_e$ ). Observations are RGB images, and actions include `MOVETOOBJECT( $o$ )` and `MOVETOREGION( $r$ )`. Transitions,  $\mathcal{T}$ , are stochastic, and the observation model,  $O$ , reflects uncertainty about the target object’s location in novel, partially observable scenes. The goal,  $\mathcal{G}$ , is an open-vocabulary object description.

Solving this POMDP by maintaining a belief over arbitrary scene structures and open-vocabulary goals is intractable. Instead, we forgo explicit belief tracking and planning, and leverage foundation models to update a scene memory that serves as a maximum likelihood estimate of the state,  $\mathcal{S}$ . An LLM-based policy selects actions based on this memory.

*OSG Navigator* instantiates this approach (Figure 2) as a high-level mapping and reasoning layer coupled with a low-level navigation layer that implements the action primitives. To achieve the required generalisation, *OSG Navigator* is modularly composed from foundation models, specifically four types of models: (i) LLMs, (ii) GNMs, (iii) VFMs for Visual Question Answering (VQA), and (iv) VFMs for Open-set Object Detection.

The high-level layer builds and reasons over OSGs, which serve as a generalisable, structured scene memory, and generates image subgoals which guide the low-level navigation layer. It does so with the *Mapper* and *Reasoner* modules, which are templated algorithms that use OSG schemas to interface with the OSG. The *Mapper* extracts open-vocabulary text describing salient scene elements from images, then uses LLMs to update the OSG. The *Reasoner* uses LLMs to identify promising subgoals in the OSG, and plans a path over the OSG to reach it. The path is a sequence of image goals, with each goal an image crop of an object from an intermediate region along the path. The low-level navigation layer uses image goal-conditioned visuomotor controllers to drive the robot toward the subgoals. Specifically, it uses GNMs that execute at 10Hz. VLM perception in the higher layer operates at 2-2.5Hz, while LLM-based reasoning takes several seconds to update the scene graph and plan after reaching each subgoal, enabling near real-time object search.

## 4 Open Scene Graphs

An *Open Scene Graph* (OSG) is a structured representation of open-set scene objects and spatial regions across multiple levels of abstraction. We define an ontology that abstractly describes the generic structure of indoor environments in terms of broad concept classes (e.g. Objects, Places, Connectors, Region Abstractions) and relationship types (e.g. proximity, connectivity, enclosure and hierarchy). This

ontology provides the organisation principles for an OSG, and provides a framework in terms of which we can design and express a mapping algorithm for constructing OSGs.

For effective semantic reasoning, the OSG needs to describe scenes using specific semantic concepts. For instance, specific and meaningful concepts in terms of which to describe a home are rooms, corridors and floors (Figure 3). The abstract concepts in the ontology are *specialised* for a particular environment type, yielding *OSG schemas* that specify the spatial concepts and relationships characteristic to that environment. **OSG Schema 1** provides an example for general household environments. OSG schemas can be manually defined by the user, enabling them to incorporate domain knowledge about the particular class of environments the robot will encounter. However, we also propose an automated pipeline that leverages LLMs to generate OSG schemas automatically given simple semantic labels of indoor scenes (Section 5.1).

We first introduce the OSG framework in this section. OSGs can be constructed incrementally with an executable mapping routine described in the next section (Section 5). This is instantiated with an OSG schema, from a templated mapping algorithm specified in terms of the ontology. Formally, the resulting OSG instance  $\mathcal{G}$ , is a heterogeneous, simple, directed graph organized into  $N$  layers, capturing a single scene from an environment type  $E$  (e.g. homes, supermarkets, hospitals).

### OSG Schema 1: Example schema for home environments

```

Floor:
  layer_type: Region
  layer_id: 3
  contains: Room, Corridor
  connects_to: Stairs

Room:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Entrance, Room, Stairs

Corridor:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Entrance, Room, Stairs

Stairs:
  layer_type: Place
  layer_id: 2
  is_near: Object
  connects_to: Floor, Room

Entrance:
  layer_type: Connector
  layer_id: 2
  is_near: Object
  connects_to: Room, Corridor

Object:
  layer_id: 1

```

#### 4.1 Abstract Node and Edge Types

OSGs represent information about *objects* and *locations*, where locations refer to sets of points that are fixed in space.

We define four abstract node types in an OSG, each of which encapsulates different types and combinations of object and location information.

- **Objects:** These are semantically meaningful scene elements occupying spatially localised regions. [Figure 3](#) highlights a bed, sink and dining table as examples of Objects. To a robot, Objects provide semantic cues for reasoning, and can serve as navigation goals or localisation landmarks. In particular, the collection of neighbouring Objects can serve as a distinctive feature for localisation. This leads us to use *object features*, *i.e.* aggregated lists of nearby Objects, for state estimation and data association.
- **Places:** These represent locations, specifically the most granular semantically meaningful spatial regions of a scene. Thus each scene can be spatially partitioned into a network of Places. Concretely, Places in homes can be *rooms* or *corridors* ([OSG Schema 1](#)) which do not consistently have semantically significant subregions. In supermarkets, Places may instead be *aisles*.
- **Connectors:** These are localised structural elements that connect adjacent spatial regions and are therefore fixed on their boundaries, giving them properties of both objects and locations. Examples include *entrances*, *doors* and *gates*. Connectors can serve as specific waypoints in space, and allow paths between regions to be finely described. Also, Connectors are also semantically meaningful and significant frontiers useful for exploration.
- **Region Abstractions:** These represent coarse locations, and are higher-level spatial abstractions which contain multiple smaller spatial regions. For instance, a *floor* may contain many *rooms*, while a supermarket *section* comprises a cluster of *aisles*. Hierarchical abstractions like these can aid in reasoning, especially with LLMs ([Zheng et al. 2024](#)).

We define four abstract edge types to capture various relations among objects and locations in a scene, which are encoded as directed edges in OSGs.

- **Proximity:** “is near” edges are *object-object* relations that approximately represent spatial closeness between objects. Concretely, the endpoints of such edges may be either Object or Connector nodes. This relation is used to select nearby nodes to use in *object feature* landmarks, enabling us to distinguish objects and locations by their context.
- **Connectivity:** “connects to” edges are *location-location* relations that represent reachability between locations. This is a fundamental relation to enable navigation, also captured in representations ranging from occupancy grids to topological graphs. The endpoints of such edges may be any node representing locations, *i.e.* Places, Connectors or Region Abstractions.

- **Inclusion:** “has” edges connect *locations to objects*, in particular identifying that an Object node (destination) is spatially contained within a Place node (source).
- **Hierarchy:** “contains” edges are *location-location* relations representing spatial containment of a region within another region. For example, it might connect a *floor* Region Abstraction node to a *room* Place node. These edges serve to define a tree-like hierarchy of spatial abstractions on the OSG, where each higher level captures a coarser abstraction. Source nodes must be Region Abstraction nodes, and destination nodes must be the location nodes at the next lower level in the tree, which may be either a Region Abstraction or Place node.

## 4.2 Layers and Structure of an OSG

We describe the structural constraints and hierarchy of OSGs, and detail the attributes maintained in them. This ensures that OSGs contain the information necessary to support robot functions.

**Layer 1: Objects.** This layer captures the distinct objects in the scene, along with proximity relations among them. Object nodes are leaf nodes of the OSG, and each maintain four attributes: **(i)** an open-vocabulary text label for the object,  $v_{label}^o$  (*e.g.* *coffeetable*, *armchair*); **(ii)** an open-vocabulary text description of the object’s appearance,  $v_{desc}^o$  (*e.g.* *white wooden* for a *coffeetable* node); **(iii)** a unique node ID,  $v_{id}^o$ ; **(iv)** an image crop of the object,  $v_{img}^o$ .

We define an Object node’s object features as the aggregated text labels and appearance descriptions of neighbouring nodes connected via “is near” edges. *I.e.*,  $[(v_{label}^o, v_{desc}^o)_0, \dots, (v_{label}^o, v_{desc}^o)_{K-1}]$  for a node with  $K$  neighbours. These features are distinctive landmarks to distinguish Object nodes.

**Layer 2: Places and Connectors.** This layer captures fine-grained spatial location information, and describes the connectivity among locations in a scene. Both Place and Connector nodes may be found in this layer. During navigation, the robot’s state is specified as a Place node from this layer.

Each Place node  $v^p$  contains the attributes: **(i)** its class as a text string drawn from the OSG schema,  $v_{cls}^p$  (*e.g.* *rooms*, *aisles*); **(ii)** an open-vocabulary text label describing the Place  $v_{label}^p$ , (*e.g.* *living room* for rooms, *dairy aisle* for aisles); **(iii)** a unique node ID,  $v_{id}^p$ . Outgoing “connects to” edges to Place/Connector nodes define the scene connectivity and topology. Outgoing “contains” edges identify the Objects that lie within the Place node. Formally a Place node’s *object features* are the aggregated text labels and appearance descriptions of Object nodes contained within it, *i.e.* nodes connected by “contains” edges.

As Connectors share object properties, they inherit the attributes and relations of Object nodes. They are also considered leaf nodes, which do not contain Objects. However as locations, they may also be linked to other locations with “connects to” edges. We note that not all environments have Connector nodes (*e.g.* open-plan environments).

**Layers 3- $N$  (Optional): Region Abstractions.** Each layer defines a different spatial abstraction, with abstractions in higher layers being more general. Each layer partitions the scene in terms of its associated abstraction. Specifically, the abstractions in layer  $(i + 1)$  should be a meaningful partition over nodes in layer  $i$ : e.g. *floors* forming a partition over the *rooms* in a multi-storey house, and *sections* serving as a partition over *aisles* in a supermarket context.

Thus nodes in the  $(i + 1)$ th layer can only have “contains” edges to one or more nodes in the  $i$ th layer. Each node may be joined to Connector nodes via “connects to” edges to capture spatial connectivity—e.g. from Figure 3, *floor* abstraction nodes are linked via *steps* (a Connector node). Each node contains at least the attributes: (i) an open-vocabulary text label describing the region  $v_{label}^i$ ; (ii) a unique node ID,  $v_{id}^i$ .

## 5 Topo-Semantic Mapping and Localisation

### Algorithm 1 Mapper

**Input:** Image  $I_t$ , OSG schema  $\mathcal{S}$ , previous OSG  $\mathcal{G}$ , previous robot state (Place)  $s_{t-1}^a$

**Output:** Updated OSG  $\mathcal{G}'$ , updated current state  $s_t^a$

// Identify Objects, Connectors and Places

1:  $\mathcal{O}_{parsed} \leftarrow \text{IMAGEPARSER}(\mathcal{S}, I_t)$

// Estimate state (i.e. current occupied Place node)

2:  $\hat{s}_t^a \leftarrow \text{STATEESTIMATOR}(\mathcal{S}, \mathcal{G}, s_{t-1}^a, \mathcal{O}_{parsed})$

// Integrate observations into the OSG

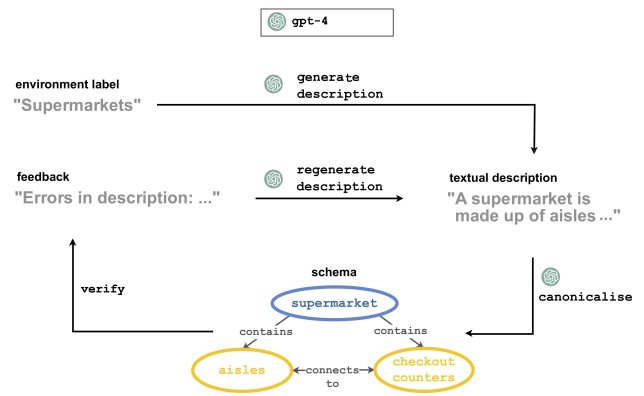
3:  $\mathcal{G}', s_t^a \leftarrow \text{OSGUPDATER}(\mathcal{S}, \mathcal{G}, \hat{s}_t^a, s_{t-1}^a, \mathcal{O}_{parsed})$

4: **return**  $\mathcal{G}', s_t^a$

OSG Navigator performs topo-semantic localisation and mapping, constructing and updating an OSG state representation online during navigation using the *Mapper*. The Mapper (Algorithm 1) routine given an OSG schema. The schema defines environment-specific spatial abstractions, filling in the template’s parameters with concrete spatial concepts and their relational structure. Since these concepts are open-vocabulary, the Mapper leverages LLMs to interpret and reason about them.

Section 5.1 describes an offline OSG schema generation pipeline, that draws on LLMs’ open-world knowledge of indoor environments to construct schemas given a text label of the environment type. This enables the Mapper, and by extension the OSG Navigator, to generalise zero-shot to new environments *without human guidance*.

The Mapper constructs OSG online, and comprises three components (Figure 2): (1) an *image parser* to extract semantic information from RGB observations (Section 5.2); (2) a *state estimator* (Section 5.3); (3) an *OSG updater* to integrate parsed information into the OSG (Section 5.4). These components rely on foundation models—LLMs, VQA models and open-set object detection (dependencies highlighted in brown in the algorithms below). While the Mapper is deterministic, we propose an approach in Section 5.5 to extend it to incorporate probabilities and maintain multiple hypotheses about the state and OSG.



**Figure 4. Automatic OSG schema generation.** OSG schemas are generated from environment labels, e.g. “supermarkets”, via an iterative process.

### 5.1 Automatic OSG Schema Generation

The Mapper’s ability to build a coherent and descriptive scene graph relies on having a sound and sufficiently representative OSG schema for that environment. Our OSG schema generator can automate the design process for such schemas: given only a simple text label describing the environment type, e.g. “homes” or “supermarkets”, the generator produces an OSG schema for that environment. Specifically, the generator produces a JSON-structured OSG schema that is *correct*, i.e. satisfies OSG’s structural requirements, while providing *coverage*, i.e. uses suitable spatial abstractions and provides sufficient detail to describe the typical structure of this environment type. Intuitively, we draw on LLMs’ prior knowledge of environments and ability to organise open-vocabulary information for this task. Our approach (Figure 4) decomposes the generator into a three-stage pipeline, where each stage focuses on optimising either correctness or coverage. The pipeline is run iteratively to refine the generated schema until it complies with the OSG’s requirements. While the current pipeline requires an input label of the environment type, we note that this can be further automated by using VLMs to classify the environment based on sensor input.

**Stage 1: Description Generation.** We prompt the LLM to generate an unstructured, long-form natural language description of the given environment type (LLM Prompt 14). It is instructed to provide details on the typical spatial abstractions used to describe such environments, and possible relations between these abstractions. We hypothesise that this text generation task is more indistribution for LLMs (Brown et al. 2020) and an effective way to query large amounts of common-sense knowledge.

**Stage 2: Canonicalise.** We guide the LLM to convert the unstructured description into an OSG schema in the form of a canonical graph representation that captures identified spatial abstractions as nodes and relations among them as edges. We take a similar approach to Zhang and Soh (2024): the LLM first extracts triplets [*Abstraction1*, *Relation*, *Abstraction2*] from the text (LLM Prompt 16), then canonicalises the abstractions and relations (LLM Prompt 18). Specifically, the LLM maps spatial abstractions to specified node types (*Objects*, *Connectors*, *Places*, *Region Abstractions*), and

selects the subset of relations corresponding to allowed OSG edge types (e.g. `connects to`, `contains`). We assemble the canonical graph from the canonicalised triplets.

**Stage 3: Verify.** The canonical graph is checked against the OSG’s structural requirements. The OSG implements a verifier class to parse OSG schemas and provide natural language feedback for detected violations of an OSG’s requirements. If errors are detected, the feedback is appended to the textual description from Stage 1, and the LLM is prompted to re-generate the description to correct the errors. Otherwise, the OSG schema is returned in a JSON format.

## 5.2 Image Parser

This extracts image and open-vocabulary textual information about Places, Objects and Connectors from RGB images. It relies on VFMs to parse visual inputs: an *open-set object detector* — GroundingDINO (Oquab et al. 2023) — localises and labels scene elements, while a *VQA model* — BLIP2 (Li et al. 2023) — translates image observations into compact language descriptions. We provide details on incorporating extracted images and textual information into OSG in Section 5.3 and Section 5.4.

**Parsing Place.** We obtain a semantically meaningful text label for the current Place node by querying a VQA model with RGB images of the robot’s surroundings. As the OSG schema may contain multiple types of Places, e.g. *rooms*, *corridors etc.*, we first ask the VQA model to select the Place type that best describes the observations (VQA Prompt 1). Given this inferred type, the VQA model is then queried for a more specific label of the currently occupied Place that reflects its function or semantic meaning (VQA Prompt 2). E.g., given that the current Place is a *room* instance, the VQA is required to label it specifically, as a *living-room* or *kitchen* for example.

**Parsing Objects/Connectors.** An open-set object detector identifies and labels Objects and Connectors in the scene. As the detector does not distinguish between Objects and Connectors, we prompt an LLM to sort the detections into the two types of leaf nodes (LLM Prompt 1). The VQA model is queried to describe image crops of each Object/Connector, providing the textual attributes for each node (VQA Prompt 3). We use a heuristic that connects leaf nodes with “is near” edges if pixel distances between their corresponding bounding boxes fall below a threshold.

**Goal Detection.** The image parser prompts an LLM with the detected objects and their descriptions, and asks it to identify whether any object in the set matches the described goal well. The agent will navigate directly to any detected goal and terminate the episode. We find empirically that goals specifying commonplace object categories can robustly be detected using string matching on the object labels output by the BLIP-2 VQA model, to a similar success rate as LLMs.

## 5.3 State Estimator

The state estimator compares textual observations extracted by the Image Parser from visual input, with the OSG to determine the robot’s state  $s_t^a$ , which is the currently occupied Place node. Conceptually, the robot’s state is identified through place recognition, based on matching

---

### Algorithm 2 State Estimator

---

**Input:** OSG schema  $\mathcal{S}$ , OSG  $\mathcal{G}$ , previous robot state  $s_{t-1}^a$ , parsed observations  $(P, O, C)$   
**Output:** Estimated robot state  $\hat{s}_t^a$

- 1:  $\mathcal{V}_P \leftarrow \text{GETPLACESLAYER}(\mathcal{G})$   
*// Object features from observations*
- 2:  $F_{obs} \leftarrow O \cup C$   
*// Get semantically similar nodes (LLM Prompt 3)*
- 3:  $\mathcal{V}_{sim} \leftarrow \text{GETSIMILARPLACES\_LLM}(\mathcal{S}, P, \mathcal{V}_P)$   
*// Sort nodes by increasing distance from  $s_{t-1}^a$*
- 4:  $\mathcal{V}_{sorted} \leftarrow \text{SORTBYDISTANCE}(\mathcal{V}_{sim}, s_{t-1}^a)$
- 5: **for**  $v$  **in**  $\mathcal{V}_{sorted}$ :
- 6:    $F_{obj} \leftarrow \text{GETOBJECTFEATURES}(\mathcal{G}, v)$   
*// Object feature matching (LLM Prompt 5)*
- 7:   **if**  $\text{PAIRWISEPLACEMATCH\_LLM}(\mathcal{S}, F_{obj}, F_{obs})$ :
- 8:     **return**  $v$
- 9: **return** None

---

textual observations of the surroundings with the *object features* of possible state nodes in the OSG.

In practice, the state estimator only attempts to match with semantically similar Place nodes in the OSG, greedily prioritising those likely to be nearby (Algorithm 2). LLMs play two key roles in state estimation, being employed in both cases as *similarity metrics* for semantic concepts expressed in ambiguous natural language. Firstly, LLMs help narrow down the set of Place nodes to those with semantically similar labels to the Place label inferred from observations. For instance, if the robot observes that it is in a *bedroom*, the LLM should select nodes of similar function or meaning like *bedrooms* and *guestrooms* as plausible robot states. Secondly, LLMs match Place nodes with observations in a pairwise fashion. They are guided to make a binary decision on the similarity between observed objects and each Place node’s object features, with emphasis on larger objects.

## 5.4 OSG Updater

The OSG updater incrementally integrates observations from the Image Parser into the OSG. Conceptually, it first updates the OSG with directly observable scene elements—i.e. Places, Objects, Connectors, and the edges between them—then abstracts them using the semantic spatial abstractions from the OSG schema (Algorithm 3). Objects/Connectors and their spatial proximity relations, as well as Places, are directly updated from the parsed observations. Spatial connectivity between Places and Connectors is determined from the parsed observations, estimated state and history of previous robot states, following the valid connectivity rules from the OSG schema.

Two challenges underlie this approach: (i) data association, to identify if an observed scene element is seen and thus determine whether to add a new node or update an existing one; (ii) reasoning to describe and abstract away spatial regions using semantic abstractions. As both involve dealing with semantic information and concepts expressed in natural language, we address them with LLMs.

**Data Association.** We hypothesise that a scene element’s *object features*, i.e. a list of nearby Objects/Connectors and their descriptions, serves as a distinctive signature

**Algorithm 3** OSG Updater

---

**Input:** OSG schema  $\mathcal{S}$ , OSG  $\mathcal{G}$ , estimated state  $\hat{s}_t^a$ , previous state  $s_{t-1}^a$ , parsed observations  $(P, O, C)$

**Output:** Updated current state  $s_t^a$ , updated OSG  $\mathcal{G}'$

```

1: if  $\hat{s}_t^a$  is None:
2:   // Unvisited area. Add new Place and leaf (Object and Connector) nodes.
3:    $s_t^a \leftarrow \text{ADDPLACE}\text{NODE}(\mathcal{G}, P)$ 
4:    $\text{ADDLEAF}\text{NODES}\text{AND}\text{EDGES}(\mathcal{S}, \mathcal{G}, O \cup C)$ 
5:   // Update successively higher region layers, using LLMs to form abstractions (LLM Prompt 8)
6:    $v_t, v_{t-1} \leftarrow s_t^a, s_{t-1}^a$ 
7:   for  $i$  in  $\text{range}(4, \text{GETNUM}\text{LAYERS}(\mathcal{S}) + 1)$ :
8:      $\mathcal{V}_i \leftarrow \text{GETLAYER}(\mathcal{G}, i)$ 
9:      $v_{abs} \leftarrow \text{INFERREGION}\text{LLM}(\mathcal{S}, \mathcal{V}_i, v_t, v_{t-1})$ 
10:    if  $v_{abs}$  in  $\mathcal{V}_i$ :
11:      //  $s_t^a$  in existing region abstraction node
12:       $\text{ADDEGE}(\mathcal{S}, \mathcal{G}, v_{abs}, v_t)$ 
13:      break
14:    else
15:      // Add new region abstraction node
16:       $\text{ADD}\text{NODE}(\mathcal{V}_i, v_{abs})$ 
17:       $\text{ADDEGE}(\mathcal{S}, \mathcal{G}, v_{abs}, v_t)$ 
18:       $\text{ADDEGE}(\mathcal{S}, \mathcal{G}, v_{abs}, v_{t-1})$ 
19:       $v_t, v_{t-1} \leftarrow v_{abs}, \text{GETPARENT}(\mathcal{G}, v_{t-1})$ 
20:    else
21:      // Previously visited Place. Associate observations with leaf nodes to update current Place's contents
22:       $s_t^a \leftarrow \hat{s}_t^a$ 
23:       $\mathcal{V}_{\text{leaf}}, \mathcal{F}_{\text{leaf}} \leftarrow \text{GETLEAF}\text{NODES}\text{AND}\text{FEATS}(\mathcal{G}, s_t^a)$ 
24:      for  $o$  in  $O \cup C$ :
25:         $f_o \leftarrow \text{GETOBJECT}\text{FEATURES}(o, O \cup C)$ 
26:        // Associate  $o$  with leaf nodes (LLM Prompt 7)
27:         $o' \leftarrow \text{MATCHOBJECTS}\text{LLM}(o, f_o, \mathcal{V}_{\text{leaf}}, \mathcal{F}_{\text{leaf}})$ 
28:        if  $o'$  is None
29:           $\text{ADDLEAF}\text{NODES}\text{AND}\text{EDGES}(\mathcal{S}, \mathcal{G}, \{o\})$ 
30:        else
31:           $\mathcal{E} \leftarrow \text{GETOBSERVED}\text{EDGES}(\mathcal{G}, o, O \cup C)$ 
32:           $\text{UPDATELEAF}\text{NODES}\text{AND}\text{EDGES}(\mathcal{G}, o', o, \mathcal{E})$ 
33:      return  $\mathcal{G}, s_t^a$ 

```

---

by which they may be recognised. We use the LLM to gauge *semantic similarity* between the object features of an observed Object/Connector and those of the leaf nodes in the robot's vicinity (Algorithm 3, Line 27). The LLM is prompted to return the associated leaf node  $v \in \mathcal{V}_{\text{leaf}} \cup \{\emptyset\}$ , where  $\mathcal{V}_{\text{leaf}}$  is the set of all leaf nodes.

**Abstracting Spatial Regions.** To construct the OSG's hierarchy of spatial abstractions, we partition layer  $i$ 's nodes based on the abstraction specified for layer  $(i + 1)$  in the OSG schema. Spatial abstractions can sometimes be inferred from spatial connectivity and topology, based on the structure given in the OSG schema. For instance in Figure 3, the *stairs* Connector serves as the marker to segment the environment into different *floor* abstraction nodes. In other cases however, semantic reasoning is needed. For instance, the different *sections* in a supermarket (e.g. deli, household items, produce sections) may not have clear structures like Connectors to delimit their boundaries. Thus semantic

reasoning may be required to cluster the individual *aisles* into such semantically meaningful *sections*. We prompt LLMs to perform this function in a greedy manner (Algorithm 3, Line 9): for each new state the robot enters, the LLM decides if it belongs to an existing abstraction node, or if the current state should belong to a new abstraction node.

### 5.5 Probabilistic OSG Mapping

Since our focus is on incorporating flexible semantics and structure into scene graphs, we design the Mapper as a deterministic algorithm for simplicity. Practically, it is capable of building accurate scene graphs over diverse environments, partly due to LLMs' strong ability to handle ambiguity and observational uncertainty (Section 7.7). However, we also propose a means to incorporate uncertainty handling into the Mapper for improved robustness, by extending it to consider perceptual errors and uncertainty. Specifically, we extend OSGs to maintain multiple hypotheses about the *topology* of the scene.

We build on *Online Probabilistic Topological Mapping* (OPTM) (Ranganathan and Dellaert 2011), a framework which maintains multiple hypotheses about possible map topologies and computes a posterior over them, using a particle filter where each particle represents a specific map topology. Concretely, a map topology  $\mathcal{T}$  is uniquely defined by a partition of the robot's observations accumulated over time, i.e.  $O = \{o_1, o_2, \dots, o_T\}$ . E.g.,

$$\mathcal{T}_T = \{v_1, \dots, v_\xi\} = \{\{o_1, o_2, o_T\}, \{o_3, o_5\}, \dots\} \quad (1)$$

Each  $v_i$  represents a node in  $\mathcal{T}_T$  and edges between nodes are implicitly represented through consecutive observations  $o_{i-1}, o_i$  that belong to separate nodes.

Like standard particle filters, OPTM first performs a predict step, which predicts likely new map topologies  $\mathcal{T}_t$  for each particle, based on their current topology  $\mathcal{T}_{t-1}$ . Next, it updates each particle's weight based on how well the observations describe the topology it represents, then resamples particles to focus on more probable topologies. Specifically in a probabilistic OSG, each topology comprising nodes from the Places and Connectors layer. Thus each particle represents a hypothesis on the fine-grained spatial connectivity of the scene. Observations  $o_t$  correspond to the parsed output from the Image Parser.

**Prediction.** For each particle the filter predicts how its topology is likely to be updated at the next timestep  $t$ , by sampling from the proposal distribution  $p(\mathcal{T}_t | \mathcal{T}_{t-1})$ . This amounts to selecting the  $v \in \mathcal{T}_{t-1} \cup \{\emptyset\} = \{v_1, \dots, v_\xi, \emptyset\}$  to which observation  $o_t$  belongs, where  $v_i$  is an existing node and the empty set is a new node. The proposal distribution is designed to favour topologies where  $o_t$  is added to frequently visited nodes (i.e. the partitions  $v_i$  with high  $|v_i|$ ), and nodes in close proximity to the robot's previous state. The former captures the assumption that the robot visits fewer new areas over time and that highly visited nodes are central areas more likely to be revisited (e.g. a corridor or lobby area), while the latter is an informal prior on the robot's possible range of motion over a short duration. Like OPTM, we model the former with a Dirichlet process, and model the latter with a uniform distribution over nearby nodes in the graph. This heuristic proposal can potentially be extended to include

observations for targeted sampling, *e.g.* increasing weight on nodes with the same Place label as the current observation's.

**Weight Update and Resampling.** Similar to OPTM, importance weights are updated as,

$$w_t^{(i)} \propto p(o_t | \mathcal{T}_t^{(i)}) w_{t-1}^{(i)}, \text{ for the } i\text{th particle} \quad (2)$$

Let node  $v' \in \mathcal{T}_t$  be the current node to which  $o_t$  belongs, and let  $\mathcal{V} \subseteq \mathcal{T}_t$  be nodes with semantically similar labels to  $o_t$ 's Place label. We define the observation model as,

$$p(o_t | \mathcal{T}_t^{(i)}) = p(o_t | v') \prod_{v_i \in \mathcal{V} \setminus \{v'\}} (1 - p(o_t | v_i)) \quad (3)$$

This captures the notion that  $o_t$  is most likely to have been generated from node  $v'$ , *i.e.* it matches best the features observed at  $v'$  while matching more poorly with other nodes. This can be viewed as a probabilistic extension to the greedy Place estimation algorithm in Section 5.3. Similar to it, we query LLMs to match observations to Place object features with Yes/No questions and obtain  $p(o_t | v)$  as the normalised logits to its response. However, this extension differs in that it fully considers all nodes in  $\mathcal{V}$  and accounts for their probabilities. The updated importance weights are then used to resample particles to focus on probable topologies.

## 6 Reasoning and Control

### 6.1 Reasoner

---

#### Algorithm 4 Reasoner

---

**Input:** OSG schema  $\mathcal{S}$ , OSG  $\mathcal{G}$ , current state  $s_t^a$ , target region  $r_{\text{subgoal}}$ , object goal  $g$

**Output:** Object goal image  $I_o$ , target region to explore  $r_{\text{subgoal}}$

```

1: if  $r_{\text{subgoal}}$  is None:
2:   // Identify promising region in  $\mathcal{G}$  to search for goal  $g$ 
3:    $r_{\text{subgoal}} \leftarrow \text{REGIONPROPOSER\_LLM}(\mathcal{S}, \mathcal{G}, g)$  (Algorithm 5)
4: if REACHEDTARGETREGION( $r_{\text{subgoal}}, s_t^a$ ):
5:   // Query LLM for an object in  $\mathcal{G}$  around which to search for  $g$  (LLM Prompt 12)
6:    $I_o \leftarrow \text{OBJECTPROPOSER\_LLM}(\mathcal{S}, \mathcal{G}, r_{\text{subgoal}})$ 
7:    $r_{\text{subgoal}} \leftarrow \text{None}$ 
8: else
9:   // Plan path to  $r_{\text{subgoal}}$  and return next waypoint
10:   $r_{\text{wp}} \leftarrow \text{PATHFINDER}(\mathcal{G}, r_{\text{subgoal}}, s_t^a)$ 
11:  // Query LLM for an Object/Connector that takes the robot toward  $r_{\text{subgoal}}$  (LLM Prompt 12)
12:   $I_o \leftarrow \text{OBJECTPROPOSER\_LLM}(\mathcal{S}, \mathcal{G}, r_{\text{wp}})$ 
13: return  $I_o, r_{\text{subgoal}}$ 

```

---

The *Reasoner* proposes promising subgoals to search for the target object based on the OSG state representation. At a high level, it semantically reasons over the OSG  $\mathcal{G}$  with LLMs to identify a subgoal, then finds a feasible path to the subgoal with graph search over  $\mathcal{G}$ . As with the Mapper, it is a templated algorithm that is grounded into an executable routine with an OSG schema, which is described in Algorithm 4. It comprises 3 components (Figure 2):

**Region Proposer.** This is a coarse-to-fine, greedy search through the layers of the OSG to propose a Place node

---

#### Algorithm 5 REGIONPROPOSER\_LLM

---

**Input:** OSG schema  $\mathcal{S}$ , OSG  $\mathcal{G}$ , object goal  $g$

**Output:** Target region to explore  $r_{\text{subgoal}}$

```

1:  $N \leftarrow \text{GETNUMLAYERS}(\mathcal{S})$ 
2: // Initialise subgraph for search
3:  $\mathcal{G}_{\text{search}} \leftarrow \text{GETLAYER}(\mathcal{G}, N)$ 
4: // Iterate over region layers, from highest to lowest
5: for  $i$  in range( $N, 1$ ):
6:   // Query LLM for most promising region to explore at current layer (LLM Prompt 10)
7:    $r_i \leftarrow \text{SELECTREGION\_LLM}(\mathcal{G}_{\text{search}}, g)$ 
8:    $\mathcal{G}_{\text{search}} \leftarrow \text{GETCHILDNODES}(\mathcal{G}, r_i)$ 
9: return  $r_i$ 

```

---

subgoal to search. At each layer, an LLM is queried to identify the node most promising to search. This ends after iterating through all layers, with a Place node identified as subgoal,  $r_{\text{subgoal}}$ .

**Pathfinder.** This module extracts the connectivity subgraph of Places and Connectors from the OSG, and plans a path with Dijkstra from the robot's current location to the subgoal node returned by the Region Proposer,  $r_{\text{subgoal}}$ .

**Object Proposer.** This identifies objects as fine-grained subgoals for navigation. When the frontier region  $r_{\text{subgoal}}$  has been reached, the Object Proposer queries LLMs to select objects in it that are likely to be close to the target. Separately, when the robot is navigating along a path to the frontier region  $r_{\text{subgoal}}$ , the Object Proposer is used to sample object subgoals from each region along the path. This is done because most current navigation models take in goals that are spatially localised, like objects. Thus, moving to a region is implemented as a sequence of MOVE TO OBJECT( $o$ ) actions, where  $o$  is an Object or Connector. We follow Shah et al. (2023a) in prompting the LLM to provide its reasoning, so as to encourage responses more relevant to the task.

### 6.2 Local Navigation Controller

The Controller module is a low-level sensorimotor loop implementing the MOVE TO OBJECT primitive. It maps RGB inputs to velocity commands, conditioned on a crop of a goal object. The MOVE TO OBJECT primitive is only expected to work locally, for goal object instances that are nearby—*i.e.* within field-of-view, or slightly occluded but still in the same area as the robot. Image goals are issued by the Reasoner, which grounds high-level plans to a sequence of local MOVE TO OBJECT actions.

Recent work on image-goal visuomotor policies have shown that they form a good substrate for generalisable visual navigation across embodiments and environments (Shah et al. 2023b; Sridhar et al. 2023). We build on these to implement MOVE TO OBJECT. Concretely, we use the ViNT GNM (Shah et al. 2023b) zero-shot for visual navigation toward target objects. We approximate an object instance goal by conditioning the GNM on an image crop of the target object—taken direct from object detector, or saved in the OSG. To decide when the goal is reached, we apply a heuristic threshold based on the temporal distance metric predicted by ViNT.

## 7 Experiments

We evaluate OSG Navigator in simulation on various ObjectNav benchmarks and also on two real robots, Fetch and Boston Dynamics Spot. We summarise our experimental findings below:

- F1** *OSG Navigator generalises zero-shot across diverse environments, embodiments and goals in the open world.* Experiments show that foundation models enable OSG Navigator to generalise across diverse environments in simulation benchmarks such as HM3D (Ramakrishnan et al. 2021) and Gibson (Xia et al. 2018) (Section 7.3) and also on real robots (Section 7.10). OSG Navigator also generalises over robot embodiments, both wheeled and legged robots, with different camera models and mountings (Section 7.5 and Section 7.10). Finally, OSG Navigator successfully handles open-vocabulary goals (Section 7.4).
- F2** *To achieve effective, generalisable object-goal navigation, OSG Navigator combines foundation models, particularly, LLMs, with OSGs.* OSGs provide a structured representation that helps LLMs to organise their rich semantic knowledge about environments for effective search (Section 7.3). In particular, OSG schemas support OSG Navigator’s ability to generalise over different environment types. We evaluated this hypothesis on multiple LLMs, including small local models (Section 7.9).
- F3** *Semantics, independent of metric geometry, provides a rich source of information for robot navigation.* OSG Navigator achieves robust performance using only semantics, without explicit metric geometry (Section 7.3). It uses semantic *object features* for localisation, and uses foundation models for spatial reasoning and image-goal navigation. We find that object features enable surprisingly strong data association for localisation using LLMs (Section 7.6).
- F4** *OSGs can be constructed for diverse environments in the open world.* We show that OSGs can be constructed zero-shot over diverse environments (Section 7.7). We further show that OSG schemas can be automatically generated for different environment types (Section 7.8).

### 7.1 Simulation Experiments Setup

**7.1.1 Datasets.** We use the Habitat simulator (Szot et al. 2021) with the Gibson (Xia et al. 2018) and HM3D-Semantics v0.1 (Ramakrishnan et al. 2021) datasets, both of which contain photorealistic 3D reconstructions of indoor environments. For Gibson, we follow Ramakrishnan et al. (2022); Chen et al. (2023) to use the tiny split’s validation set, comprising 1000 episodes evenly split over five household scenes (each with 12-28 rooms across 3 floors). For HM3D, we extend the setup of Shah et al. (2023a), doubling the number of evaluation episodes sampled from the validation set to 400. The episodes are sampled evenly across the validation set’s 20 scenes (each containing 4-19 rooms across 1-3 floors).

**7.1.2 Metrics.** We evaluate with 3 standard metrics used in the Habitat Challenge for ObjectNav (Yadav et al. 2022a):

- *Success rate (SR):* The ratio of episodes where the agent succeeds, defined by reaching a position within 1.0m from the goal instance, from which the goal can be directly observed.
- *Success-weighted path length (SPL).* Ratio of agent’s traversed distance to the ground-truth shortest path distance between the start and goal, for successful episodes. Intuitively measures efficiency of the agent’s path compared to the optimal path. Given by,

$$\frac{1}{N} \sum_{i=1}^N S_i \frac{l_i}{\max(p_i, l_i)} \quad (4)$$

$l_i, p_i$  are the optimal path length and agent’s traversed distance.  $S_i$  is a success indicator variable.  $N$  is the total number of episodes.

- *Distance-to-goal (DTG).* Average distance of the agent from the goal instance at the end of each episode.

**7.1.3 Baselines.** We compare OSG Navigator to state-of-the-art ObjectNav works, and in particular analyse its benefits over recent LLM-based approaches. Baselines which do not incorporate perception and instead use ground-truth semantic annotations are marked with *GT*.

We select state-of-the-art approaches which both (i) require ObjectNav-specific training, and (ii) which are training-free. For (i), we evaluate with SemExp (Chaplot et al. 2020) and PONI (Ramakrishnan et al. 2022). For (ii), we evaluate with purely geometric frontier-based exploration (FBE) (Yamauchi 1997), SemUtil (Chen et al. 2023) which combines FBE with semantic costs, and approaches which use LLMs to suggest or score exploration goals in the form of LGX (Dorbala et al. 2024) and LFG (Shah et al. 2023a).

LGX and LFG are ablations of OSG Navigator which use less rich/structured representations compared to OSGs. In particular, we highlight the following differences among the LLM ObjectNav baselines:

- *LGX-GT* (Dorbala et al. 2024). Prompts LLM to decide a direction to search towards, based on objects in the immediate field-of-view. *I.e.* a greedy approach which does not maintain any persistent scene representation.
- *LFG-GT* (Shah et al. 2023a). Prompts LLM with list of objects around the frontiers extracted from a metric map, to select promising frontiers to search. *I.e.* an approach which maintains a scene representation containing information on *objects* in the scene.
- *OSG-Nav-GT.* Our OSG Navigator system which prompts LLMs with OSGs to decide promising places to search. OSGs are richer representations that capture information on spatial regions and scene topology, in addition to objects.
- *OSG-Nav.* Full OSG Navigator system which does not rely on ground-truth annotations, but uses VFMs for perception.

**Table 1. ObjectNav performance of LLM ObjectNav baselines in HM3D.** We highlight the **best** and **second-best** scores.

Method	SR (↑)	SPL (↑)	DTG(↓)
FBE-GT	0.626	0.310	2.677
LGX-GT	0.275	0.080	5.078
LFG-GT	0.675	<b>0.389</b>	2.411
OSG-Nav-GT	<b>0.775</b>	<b>0.380</b>	<b>1.702</b>
OSG-Nav	<b>0.693</b>	0.283	<b>2.338</b>

We implement the LLM baselines by adapting their publicly available source code for the ObjectNav setup. We implement a Fast Marching Method (FMM) (Sethian 1996) controller based on Chaplot et al. (2020), and standardise the baselines (LGX-GT, LFG-GT) and variants of our approach (OSG-Nav-GT, OSG-Nav) to use this as a local planner in simulation, to compare their performance on the basis of differences in scene representation and reasoning.

Apart from our experiments with automatically generated OSG schemas in Section 7.8, all other simulation experiments use the hand-designed schema for household environments in OSG Schema 1.

## 7.2 Real-Robot Experiments Setup

### OSG Schema 2: Open-plan studio apartment

```
Room:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Entrance, Room
Entrance:
  layer_type: Connector
  layer_id: 2
  is near: Object
  connects_to: Room
Object:
  layer_id: 1
```

We evaluate zero-shot generalisation to different robots and diverse object queries in the real world. We test in an open-plan studio apartment comprising a living room, dining room, kitchen and office, using a manually designed OSG schema described in OSG Schema 2. We deploy the full OSG Navigator (with Mapper, Reasoner, and GNM-based Controller) on two robot systems, running all modules locally on a Jetson AGX Orin, aside from queries to OpenAI’s LLMs which is done over a Wi-Fi connection.

- *Spot*: A quadrupedal robot, with a 170° FoV fisheye RGB camera mounted  $\sim 0.8\text{m}$  above ground.
- *Fetch*: A differential-drive mobile manipulator, with a 87° FoV Realsense RGB camera mounted  $\sim 1.4\text{m}$  above ground.

## 7.3 ObjectNav in Diverse Environments

Simulation experiments in both HM3D (Table 1) and Gibson (Table 2) support our findings that OSG Navigator enables

**Table 2. Comparison with ObjectNav baselines in Gibson.** TF / NM denote training-free/non-metric approaches.

Method	SR (↑)	SPL (↑)	DTG (↓)	TF	NM
SemExp	0.657	0.339	1.474	✗	✗
PONI	<b>0.736</b>	<b>0.410</b>	<b>1.250</b>	✗	✗
LGX	0.310	0.052	4.775	✓	✗
LFG	0.645	<b>0.406</b>	1.812	✓	✗
FBE	0.641	0.283	1.780	✓	✗
SemUtil	0.693	0.405	<b>1.488</b>	✓	✗
OSG-Nav	<b>0.734</b>	0.386	1.722	✓	✓

effective ObjectNav (F2), while generalising across diverse scenes (F1). Since OSG Navigator internally reasons with and uses purely semantic cues, its strong performance on a navigation task supports F3.

Overall we find that OSG Navigator has strong zero-shot performance on ObjectNav across diverse indoor environments. The OSG-Nav variant achieves a high success rate competitive with the strongest learned baselines like PONI (Table 2) despite not having ObjectNav-specific training. Notably, we also show stronger performance than other training-free approaches, outperforming by a wide margin SemUtil (Chen et al. 2023), which combines classical planning with closed-set semantics, and other recent methods which use LLMs. The high success rates of OSG-Nav and OSG-Nav-GT on HM3D point towards strong cross-environment generalisation, given the diverse collection of household scenes found in HM3D. While our topo-semantic approach does not let us optimise for shortest geometric paths, the OSG Navigator variants still remain competitive with state-of-the-art baselines in SPL and DTG. In particular, we note that we also outperform by a large margin frontier-based exploration (FBE) (Yamauchi 1997), a purely metric exploration approach. This highlights the value of semantic priors and reasoning for efficient search across diverse environments.

We analyse the factors contributing to OSG Navigator’s success, and find that structured representations capturing both *objects* and *spatial regions* enhance ObjectNav performance. From Table 1, LFG-GT improves significantly over LGX-GT, highlighting the need to maintain a global scene memory for planning in ObjectNav, as opposed to greedily making choices based on current observations. OSG-Nav-GT outperforms LFG-GT by additionally retaining information on regions and scene topology in the OSG, suggesting the value of information on a scene’s *spatial structure* to reasoning for ObjectNav.

Retaining scene information on spatial structure can enable efficient search, especially when object information is lacking. We find that (i) region or place semantics (*e.g.* the room type) provides strong cues for reasoning, while (ii) *Connectors* like doors and entrances are often good frontiers for exploration. In particular, an OSG schema includes *Connectors* as first-class citizens to emphasise their value in reasoning for ObjectNav. Figure 5 highlights examples that demonstrate these benefits. In (a), both LFG-GT and OSG-Nav-GT face a scene with multiple rooms branching off a main corridor, and with few objects in sight. The emphasis on Connectors focuses OSG-Nav-GT on searching through new doors in the scene, while its ability to explicitly



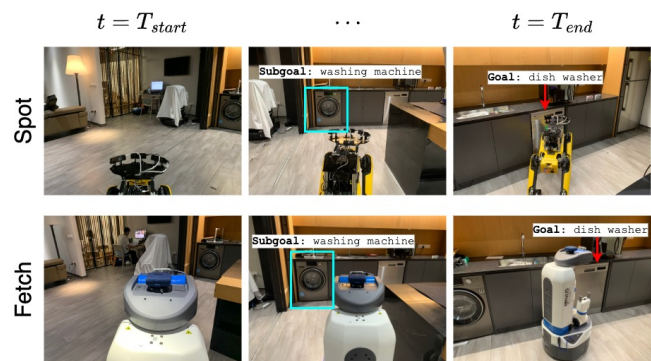
**Figure 5. Compare OSG-Nav-GT and LFG-GT in search efficiency.** OSG-Nav-GT uses region-level semantics from OSGs to search more efficiently, compared to LFG-GT which lacks a topo-semantic memory. In both scenes, the robot sees few objects at the start. **(a)** LFG-GT defaults to an exhaustive search, while OSG-Nav-GT quickly heads to a living room to find a chair. **(b)** OSG-Nav-GT sees that it starts in a bedroom and searches nearby entrances for a possible bathroom, while LFG-GT starts an exhaustive search that incidentally leads it away from the nearest toilet.

identify rooms and understand their semantic utility enables it to quickly eliminate new regions that are unsuitable for exploration. In contrast, LFG-GT defaults to an inefficient exhaustive search over the scene. Similarly in **(b)**, both LFG-GT and OSG-Nav-GT begin in a sparsely furnished bedroom. OSG-Nav-GT is able to identify its surroundings and reasons that toilets are commonly found in the vicinity of bedrooms, leading it to focus its search on nearby doors to efficiently locate the toilet. In contrast, LFG-GT defaults to frontier exploration on a metric map, selecting suboptimal object frontiers that lead it away from the nearest toilet.

#### 7.4 Generalisation over Open-Vocabulary Goals

We find that OSG Navigator has a strong ability to handle diverse open-vocabulary goals (**F1**). Our real-world experiments (**Table 3**) show that OSG Navigator is capable of locating open-set goals like “guitar” and “dish washer”, which are not specified goal classes in ObjectNav (Yadav et al. 2022a). We provide a more detailed analysis of OSG Navigator’s open-vocabulary abilities by evaluating on *rare queries* and *compositional queries* in simulation. The former are uncommon, long-tailed object concepts, which we select based on the LVIS dataset (Gupta et al. 2019). The latter are complex language descriptions of an object at different levels of granularity, which compose an object noun (e.g. “chair”) with descriptors (e.g. “leopard-print”, “fabric” etc.). While the individual nouns and adjectives may be common, the resulting queries are overall uncommon and unlikely to be present in the models’ training data. We design new navigation episodes in HM3D for these queries.

We find that OSG Navigator generalises well over *rare object queries*. To evaluate these we select two uncommon nouns based on LVIS: *settee*, an uncommon synonym for the commonplace couch; *nightstand*, an object uncommon enough that it is excluded from LVIS.



**Figure 6. Cross-embodiment generalisation in the real world.** OSG Navigator is deployed zero-shot across two different robots, Spot and Fetch, without any parameter tuning. It perceives, reasons and acts similarly despite the change in embodiment and sensors, successfully reaching the goal dish washer in both cases.

**Table 4** shows strong performance when queried with these rare nouns, suggesting strong generalisation to rare object queries. To evaluate *compositional object queries*, we compose selected object nouns with one or more phrases to describe object properties like colour, material or pattern. The navigation episodes designed for these require OSG Navigator to disambiguate amongst different object instances based on the provided descriptors (e.g. differentiating a “leopard-print” chair from a “floral-print” chair). From **Table 5**, we find that OSG Navigator performs well in general on compositional queries. We note slight drops in success rate as descriptors are added and the requirements on object appearance become more complex. We attribute this to perception errors which misclassify the object properties and cause OSG Navigator to fail to recognise the goal.

**Table 3. Real-world, open-world ObjectNav.**

Object goal	Spot	Fetch
guitar	4/5	4/5
dish washer	5/5	4/5

**Table 4. Performance of OSG-Nav and LGX on rare object queries.**

Method	Settee			Nightstand		
	SR (↑)	SPL (↑)	DTG (↓)	SR (↑)	SPL (↑)	DTG (↓)
LGX (Dorbala et al. 2024)	0.1	0.023	6.186	0.4	0.081	2.873
OSG-Nav	0.8	0.285	0.894	0.7	0.287	1.699

**Table 5. Success rate of OSG-Nav in handling open-vocabulary compositional queries.** Queries are object goals composed with one or more phrases describing colour (C), material (M) or pattern (P). *E.g.*, a “red, floral bed” (C+P with goal “bed”).

Object goal	Colour (C)	Material (M)	Pattern (P)	Success rate (over 10 trials each)						
				C	M	P	C+M	C+P	M+P	C+M+P
chair	<i>black and white</i>	<i>fabric</i>	<i>leopard-print</i>	0.8	0.6	0.8	0.6	0.6	0.6	0.6
bed	<i>red</i>	<i>cotton</i>	<i>floral</i>	0.8	0.8	0.8	0.6	0.6	0.6	0.4

## 7.5 Generalisation over Robot Embodiments

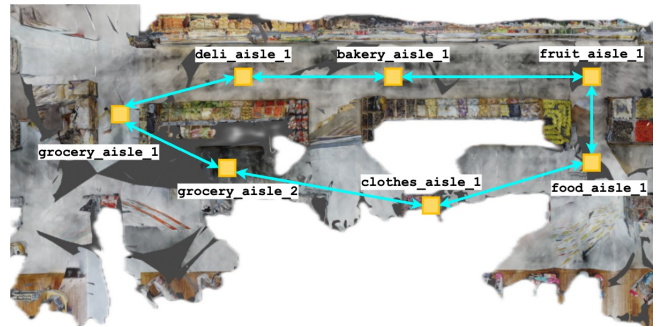
We evaluate cross-embodiment generalisation through real world experiments on different robots (Figure 6), supporting our findings in F1. These robots feature different dynamics (quadrupedal vs. differential drive) and different sensor inputs (different mounting heights, fisheye vs pinhole cameras). Table 3 shows that OSG Navigator succeeds often on various ObjectNav tasks across both robots in spite of the widely varying inputs and dynamics. This is enabled by composing OSG Navigator fully from foundation models which have shown broad generalisation across perceptual inputs (VFMs) and robot dynamics (GNMs). We note that the GNM tends to perform slightly worse at goal-reaching when narrow field-of-view cameras like the Realsense are used, and that its performance on goal-directed navigation in cluttered areas is limited: these issues induced the observed failures in the real world experiments.

## 7.6 Object Features for Data Association

As highlighted in F3, semantics alone are a rich source of information, enabling key functions in navigation that traditionally rely on geometric information. Specifically, we find that purely semantic *object features* are surprisingly effective for associating regions and objects. Such data association is essential for place recognition and object re-identification, and usually requires dense visual descriptors or geometric features.

This section evaluates the accuracy of object features in associating Objects, Connectors and Places. To do so, we use a dataset of views sampled from HM3D that is curated to examine how well object features enable LLMs to *recognise* the same Object/Connector/Place from different views and to *distinguish* among different Objects/Connectors/Places. The dataset is constructed by sampling rooms across 10 randomly selected HM3D scenes, then sampling views within selected rooms. We ensure that there are multiple views of each labelled object, and also ensure that multiple rooms of the same function and type (*e.g.* multiple bedrooms) are sampled, to make the task realistic and challenging.

From Table 6, we find that object features reliably enable LLMs to recognise and distinguish Objects, Connectors and Places in the environment. We note that the tests to distinguish Objects and Connectors involve two distinct Object/Connector instances with the same type or semantic

**Figure 7. Generated OSG for supermarket.** The Mapper constructs an OSG of the Gratz supermarket scene from the Gibson dataset using OSG Schema 6 (in Appendix E.1), which uses “aisles” as Place nodes.

label (*e.g.* two instances of “chairs”). We find that LLMs using object features can struggle to correctly distinguish distinct object instances of the same type, as human environments often have similar objects clustered closely (*e.g.* multiple chair instances grouped closely around a table). In these cases, each object’s neighbours (and hence features) are too similar, causing aliasing. Practically, such incorrect associations do not significantly affect ObjectNav, since erroneously identified objects are close to the correct object, and the system will navigate to the vicinity of the correct object. Additionally, Table 6 also shows that LLMs can strongly distinguish Places of different types (*e.g.* bedroom and kitchen) purely using the objects contained within them. This suggests that object features provide a robust source of localisation information to augment place classification-based localisation.

## 7.7 Online OSG Construction

We show that the Mapper can accurately map and capture the spatial structure of scenes across varied environment types, supporting F4. First, we evaluate the accuracy of OSG construction over five different homes in HM3D by comparing the generated OSGs to human-annotated scene graphs. Second, we provide qualitative examples of OSGs built across different environment types in both simulation and the real world.

Table 7 shows OSG quality in terms of the precision and recall of Mapper in predicting OSG nodes and their outgoing

**Table 6. Accuracy of data association with object features.**

We test LLMs’ ability to use object features to *recognise* the same scene element from different views, or *distinguish* different elements (e.g. two different Objects, or two different Places).

Node	Recognise	Distinguish	
		Same type	Diff. type
Objects	0.85	0.58	-
Connectors	0.75	0.83	-
Places	0.88	0.80	0.94

edges. We use human-annotated graphs as ground truth, and consider Connector (door), Place (room) and Region Abstraction (floor) nodes, along with the edges connecting them, in our evaluation. We observe strong performance in general across all tests and metrics, indicating that most OSGs generated by the Mapper accurately capture the scene’s spatial structure and topology. We show examples of OSGs generated from these experiments in Figure 8. We note that the OSG successfully identifies and labels most of the spatial regions (i.e. *rooms*, *hallways*, *floors*) in the scene, and also accurately captures the connectivity and hierarchy of these regions. The mapper can sometimes oversegment long hallways into several smaller hallway regions, which does not significantly affect the topology of the OSG. Another key failure mode comes from occasional mislabelling of the Place nodes by the BLIP2 VQA model, an issue that can likely be addressed through the use of more powerful VLMs, e.g. LLaVA (Liu et al. 2023a).

We demonstrate the zero-shot applicability of the OSG framework to novel environment types, like a supermarket from the Gibson dataset (Figure 7). We make no modifications to the Mapper, only providing an OSG schema for supermarkets that specifies “aisles” as a suitable Place abstraction. The resulting OSG is able to decompose the scene into aisles, and accurately identifies the *type* or semantic utility of each aisle (e.g. marking an aisle for deli items, or for clothing).

We extend our tests to the real world, building OSGs for novel environments like an *office* or *mall*. This requires the Mapper to handle diverse spatial abstractions, ranging from offices/hallways in office environments to stores/walkways in malls. We find that it is largely able to handle these abstractions well and generate OSGs which accurately represent these scenes. For instance, Figure 9 shows an OSG of a real-world office scene, which largely captures the connectivity of the environment based on sensor data from the traversed trajectory. In particular, Mapper succeeds at a performing an object feature-based loop closure at `hallway_2`. Appendix E.1 provides the schema used for this scene, along with further examples.

## 7.8 OSG Schema Generation

We demonstrate our pipeline’s ability to generate meaningful OSG schemas, that are also useful for ObjectNav (F4). We do so by qualitatively assessing schemas generated across various human environments of differing complexity—homes, hospitals and airports— and validating the performance of generated home schemas on ObjectNav in selected HM3D scenes.

**Table 7. Evaluation of scene graph quality.** We measure Precision and Recall in the estimation of nodes and outgoing edges for *floors* (Region Abstractions), *rooms* (Places), *doors* (Connectors)

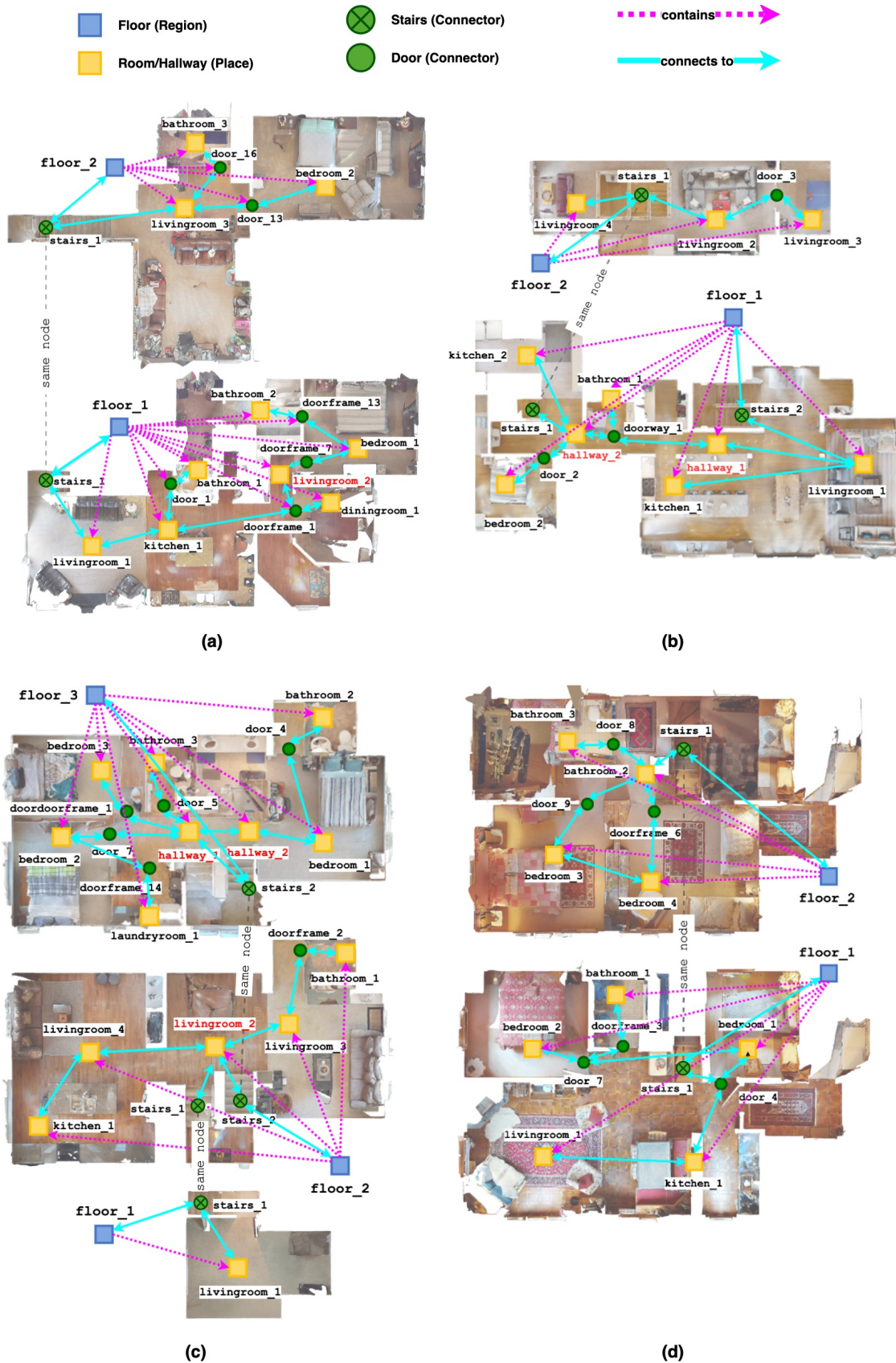
	Nodes		“contains”		“connects to”	
	Pr	Re	Pr	Re	Pr	Re
Floors	1.000	1.000	0.889	0.914	1.000	1.000
Rooms	0.846	0.880	-	-	0.771	0.831
Doors	0.845	0.800	-	-	0.783	0.857

**Table 8. Performance of OSG-Nav with automatically generated OSG schemas for homes.** Results on 10 HM3D ObjectNav episodes with OSG Navigator, using schemas generated with different methods: (i) manually specified by user, (ii) automatically generated by directly prompting LLM, (iii) automatically generated with our pipeline (Section 5.1).

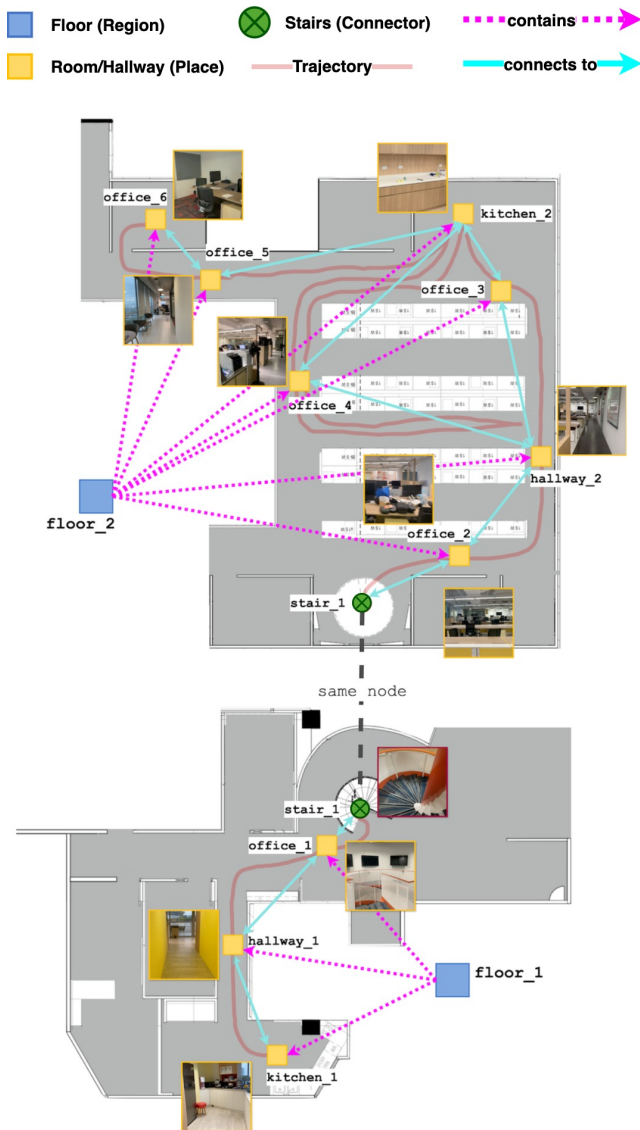
OSG spec	Valid schema SR (↑)	SR (↑)	SPL (↑)	DTG (↓)
Manual	-	0.7	0.241	1.110
Auto: LLM	0.0	0.0	-	-
Auto: Ours	1.0	0.6	0.220	2.759

Our system produces schemas that are *correct*, but which tend to have limited *coverage* in complicated environments. The schemas are correct in that the generated spatial concepts are relevant to the prompted environment (e.g. *rooms* in homes, *wards* in hospitals, *gates/terminals* in airports), and these concepts are correctly organised hierarchically (e.g. *gates* contained within *terminals*). Our system also ensures good coverage in simple environments like apartment homes, i.e. the schemas are sufficiently broad and descriptive enough to apply to such environments. We attribute this to LLMs’ strong ability to capture mainstream spatial concepts from Internet-scale text. However, we find that our system may omit more specialised region concepts—e.g. including *wards* but leaving out *operating theatres* in hospitals. It also tends to omit regions not directly related to the environment’s function—e.g. leaving out *shops* from airports. We find that these issues can be addressed by adding more user-provided guidance: e.g. extending the prompt from “airport” to “airport containing shopping areas”. Future work could also explore the design of an interactive language-based interface by which humans can incorporate richer priors and guidance into the schema generator.

We validate that the generated schemas enable effective OSG construction and ObjectNav on selected HM3D episodes. Specifically, we automatically generate schemas for apartment homes at the start of each ObjectNav episode, using our Generate-Canonicalise-Verify pipeline. Table 8 shows that our generated schemas enables performance comparable to manually designed ones. The main observed failure mode was that the generated schema sometimes left out optional Connector (*door*) nodes. While suitable for open-plan apartments, this prevented it from accurately capturing spatial connectivity in household scenes with multiple doors. We also find our iterative approach to be essential for generating valid schemas. In contrast, directly prompting LLMs to do so yielded malformed JSON schemas that often did not satisfy OSG structural requirements.



**Figure 8. Generated OSGs for multi-storey homes.** We use Mapper to construct OSGs using sensor data from teleoperated trajectories from HM3D scenes. The OSGs are built with **OSG Schema 1**. The resulting OSGs accurately capture Places (rooms/hallways), Connectors (doors/stairs), and Regions (floors), along with their spatial connectivity and hierarchy. The main failure modes observed are oversegmenting of hallways in (b), (c) and misclassifications by the BLIP2 VQA model—identifying a hallway as living room in (a), and a dining room as a living room in (c). The errors are highlighted in red.



**Figure 9. Generated OSG for a real-world office scene.** The OSG is constructed using data from the red trajectory, and includes the Place concepts of *office*, *room*, *hallway*, and the Connector *stairs* (OSG Schema 7, in Appendix E.1).

#### OSG Schema 3: Apartment home (Auto-generated)

```

Home:
  layer_type: Region
  layer_id: 3
  contains: Room
Entrance:
  layer_type: Connector
  layer_id: 2
  is near: Object
  connects_to: Room
Room:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Entrance
Object:
  layer_id: 1

```

#### OSG Schema 4: Hospital (Auto-generated)

```

Hospital:
  layer_type: Region
  layer_id: 3
  contains: Ward
Corridor:
  layer_type: Connector
  layer_id: 2
  is_near: Object
  connects_to: Ward
Ward:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Corridor
Object:
  layer_id: 1

```

#### OSG Schema 5: Airport (Auto-generated)

```

Terminal:
  layer_type: Region
  layer_id: 3
  contains: Gate
  connects_to: Terminal
Gate:
  layer_type: Place
  layer_id: 2
  has: Object
  connects_to: Gate
Object:
  layer_id: 1

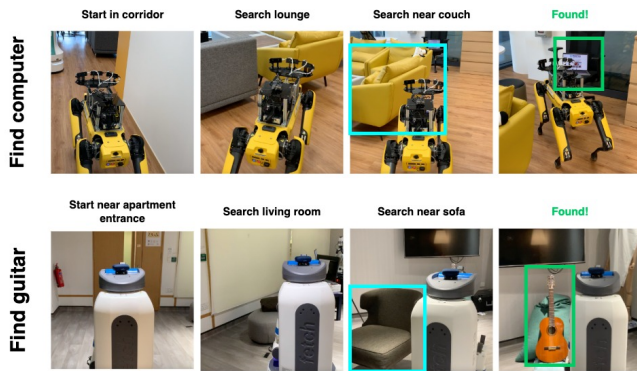
```

### 7.9 Choice of Foundation Models

We analyse the role of LLM choice on the effectiveness of ObjectNav in Table 9, to provide deeper understanding for F2. LLMs are essential to OSG Navigator for semantic reasoning and belief updating, but can be costly to run. We therefore explore the performance of OSG Navigator on smaller models that enable efficient local inference, *i.e.* the LLaMA 2 (Touvron et al. 2023) and 3 (Dubey et al. 2024) family of models. For all experiments with local models, we run the LLMs on a single Nvidia A5000 GPU. We find that performance is halved when using the 7B LLaMA 2 variant, owing to (1) its poor performance in predicting stop tokens, causing infinite recursion; (2) its tendency to reply to the prompts used in our state estimator with “cannot decide”; (3) mistakes in classifying detections into Objects, Connectors and Places when updating the OSG. These challenges are mitigated with the 13B variant. While performance remains markedly poorer than GPT-3.5, we believe these smaller models offer a reasonable tradeoff, potentially enabling OSG Navigator to run fully locally and improve efficiency.

### 7.10 Real-Robot Experiments

We support F1 with additional real-world experiments in an open-plan studio and nursing home mock-up (Figure 10). As with our previously described real-world tests, the



**Figure 10. Real-world ObjectNav.** The Spot robot searches common areas in a nursing home to find a computer (upper row), while the Fetch robot searches an apartment to find a guitar (lower row).

**Table 9. Experiments on OSG-Nav’s ObjectNav performance using different LLMs.** Results are based on 100 episodes, with 5 episodes sampled from each scene in the HM3D validation set.

LLM	SR ( $\uparrow$ )	SPL ( $\uparrow$ )	DTG ( $\downarrow$ )
Meta LLaMA 2 (7B) (Touvron et al. 2023)	0.349	0.095	4.190
Meta LLaMA 3.1 (8B) (Dubey et al. 2024)	0.492	0.145	3.241
Meta LLaMA 2 (13B) (Touvron et al. 2023)	0.538	0.124	2.867
OpenAI GPT-3.5	<b>0.693</b>	<b>0.282</b>	<b>2.385</b>

OSG Navigator is deployed zero-shot on different robots (Section 7.2) without any change in model weights or parameters from the simulation tests. We find that it is generally robust to shifts in environment and visual appearance, and demonstrates the ability to perform ObjectNav and locate open-vocabulary objects outside the usual ObjectNav categories, like a guitar and a computer. The supplementary video contains these and other tests, along with a demonstration of topo-semantic SLAM using OSGs in the real world.

VFM’s strong generalisation over environments and across camera models and mountings are a key enabler of this robust transfer. While VFMs overall perform well in zero-shot settings, we find that they still remain sensitive to specific environmental conditions (*e.g.* strong illumination changes). We also note that open-set object detectors like GroundingDINO appear to trade off some detection accuracy compared to specialised detectors trained on closed-set object detection with a much more limited vocabulary, a phenomenon also noted by Chang et al. (2023).

Generally, we find that GNMs are the main factor limiting the OSG Navigator’s ObjectNav performance in the real world. Firstly, while GNMs are capable of reactive obstacle avoidance, their capacity for goal-directed visual navigation is limited. We attribute this to limited local planning ability, causing GNMs to get stuck in local minima in slightly more cluttered scenes. Our experiments take place in more open, cleaner environments, reducing the occurrence of such issues. We view this as an important issue for practical implementation, which can be addressed through adding heuristic recovery behaviours, sampling exploration trajectories from a GNM to escape local minima as done in Sridhar et al. (2023), and through future work on GNMs. Secondly, GNMs ability to handle different sensing

modalities is notably more limited than those of VFMs. We find empirically that GNMs have a clear bias toward wider field-of-view cameras, making more dynamic movements and taking tighter turns with them. The authors attribute this to the composition of the dataset for training GNMs.

## 8 Conclusion and Future Work

This paper presents OSG Navigator, a neuro-symbolic system that combines foundation models and the OSG representation in a modular architecture for open-world ObjectNav. OSG is a novel scene graph representation that leverages *schemas* to enable effective generalisation over diverse environments. Experiments demonstrate that OSG Navigator achieves state-of-the-art performance on ObjectNav benchmarks, while generalising across environments, robots and open-vocabulary queries. They further indicate that OSGs provide the rich, structured representation of the scene necessary to enable such performance.

Our approach currently relies only on topo-semantic information. However, incorporating metric information—*e.g.* positioning information like wheel odometry, or geometric information like depth from foundation models (Yang et al. 2024)—can enhance understanding of rich, 3D scenes. This extends to providing finer-grained geometric understanding needed for higher precision tasks like manipulation.

While our work leverages OSG schemas for cross-environment generalisation, it does not yet fully exploit their flexibility. These schemas can be adapted online to better suit the environments being navigated, improving performance across episodes in complex or heterogeneous spaces (*e.g.*, a transport hub linked to a shopping centre). This is reminiscent of “iterative” vision-and-language navigation (Krantz et al. 2023) which emphasises lifelong learning, and developing a similar benchmark for our setting is a promising direction. Although we focus on indoor spaces, the generic ontology can describe outdoor, human-built environments as well. However handling large-scale, open-plan areas may require advances in our mapping pipeline. Outdoor spaces are often more heterogeneous than indoor spaces, potentially requiring more complex schemas to describe. Lastly, our current ontology targets spatial regions for navigation but could be extended to support finer-grained regions relevant to manipulation tasks.

The OSG Navigator currently uses LLMs in a limited capacity. It relies on very large, cloud-based LLMs like GPT-3.5, and does not fully exploit their multilingual and multimodal capabilities. We highlight that ongoing research into building compact, efficient LLMs (Deitke et al. 2024), along with promising preliminary results with smaller LLaMA models (Section 7.9) offer positive indications that OSG Navigator can be optimised for more constrained, fully onboard compute. We also leave to future work the harnessing of LLMs’ multilingual and multimodal capabilities, which can enable richer interaction with users.

## Acknowledgements

This research is supported by the DSO National Laboratories’ Graduate Fellowship.

## References

- Al-Halah Z, Ramakrishnan SK and Grauman K (2022) Zero experience required: Plug & play modular transfer learning for semantic visual navigation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 17031–17041.
- Amiri S, Chandan K and Zhang S (2022) Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters* .
- Anderson P, Chang A, Chaplot DS, Dosovitskiy A, Gupta S et al. (2018) On evaluation of embodied navigation agents.
- Anwar A, Welsh J, Biswas J, Pouya S and Chang Y (2024) Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation.
- Armeni I, He ZY, Gwak J, Zamir A, Fischer M et al. (2019) 3d scene graph: A structure for unified semantics, 3d space, and camera. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* : 5663–5672.
- Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD et al. (2020) Language models are few-shot learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M and Lin H (eds.) *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., pp. 1877–1901.
- Cai W, Huang S, Cheng G, Long Y, Gao P et al. (2023) Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill.
- Chang M, Gervet T, Khanna M, Yenamandra S, Shah D et al. (2023) Goat: Go to any thing.
- Chaplot DS, Gandhi DP, Gupta A and Salakhutdinov RR (2020) Object goal navigation using goal-oriented semantic exploration. In: Larochelle H, Ranzato M, Hadsell R, Balcan M and Lin H (eds.) *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., pp. 4247–4258.
- Chen J, Li G, Kumar S, Ghanem B and Yu F (2023) How To Not Train Your Dragon: Training-free Embodied Object Goal Navigation with Semantic Frontiers. In: *Proceedings of Robotics: Science and Systems*. Daegu, Republic of Korea.
- Chen J, Lin B, Xu R, Chai Z, Liang X and Wong KYK (2024) Mapgpt: Map-guided prompting for unified vision-and-language navigation.
- Dai Z, Asgharivaskasi A, Duong T, Lin S, Tzes ME et al. (2024) Optimal scene graph planning with large language model guidance. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 14062–14069. DOI:10.1109/ICRA57147.2024.10610599.
- Deitke M, Clark C, Lee S, Tripathi R, Yang Y et al. (2024) Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models.
- Dorbala VS, Mullen JF and Manocha D (2024) Can an embodied agent find your “cat-shaped mug”? IIm-based zero-shot object navigation. *IEEE Robotics and Automation Letters* 9(5): 4083–4090.
- Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A et al. (2024) The llama 3 herd of models.
- Gadre SY, Wortsman M, Ilharco G, Schmidt L and Song S (2023) Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. *CVPR* .
- Grinvald M, Furrer F, Novkovic T, Chung JJ, Cadena C et al. (2019) Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters* .
- Gu J, Stefani E, Wu Q, Thomason J and Wang X (2022) Vision-and-language navigation: A survey of tasks, methods, and future directions. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 7606–7623.
- Gu Q, Kuwajerwala A, Morin S, Jatavallabhula K, Sen B et al. (2023) Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *arXiv* .
- Gupta A, Dollar P and Girshick R (2019) LVIS: A dataset for large vocabulary instance segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 5351–5359.
- Honerkamp D, Büchner M, Despinoy F, Welschhold T and Valada A (2024) Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *arXiv preprint arXiv:2403.08605* .
- Hu Y, Xie Q, Jain V, Francis J, Patrikar J et al. (2023) Toward general-purpose robots via foundation models: A survey and meta-analysis.
- Huang C and Zhang L (2025) On the limit of language models as planning formalizers.
- Huang J and Chang KCC (2022) Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403* .
- Huang W, Abbeel P, Pathak D and Mordatch I (2022) Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In: Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G and Sabato S (eds.) *Proceedings of the 39th International Conference on Machine Learning, Proceedings of Machine Learning Research*, volume 162. PMLR, pp. 9118–9147.
- Hughes N, Chang Y and Carlone L (2022) Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization. In: *Proceedings of Robotics: Science and Systems*. New York City, NY, USA.
- Hutchinson S, Hager G and Corke P (1996) A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on* 12: 651 – 670.
- Ichter B, Brohan A, Chebotar Y, Finn C, Hausman K et al. (2023) Do as i can, not as i say: Grounding language in robotic affordances. In: Liu K, Kulic D and Ichnowski J (eds.) *Proceedings of The 6th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 205. PMLR, pp. 287–318.
- Jatavallabhula K, Kuwajerwala A, Gu Q, Omama M, Chen T et al. (2023) Conceptfusion: Open-set multimodal 3d mapping. *Robotics: Science and Systems (RSS)* .
- Kassab C, Mattamala M, Zhang L and Fallon M (2024) Language-extended indoor slam (lexis): A versatile system for real-time visual scene understanding. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 15988–15994.
- Khandelwal A, Weihs L, Mottaghi R and Kembhavi A (2022) Simple but effective: Clip embeddings for embodied ai. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 14829–14838.
- Kirillov A, Mintun E, Ravi N, Mao H, Rolland C et al. (2023) Segment anything. *arXiv:2304.02643* .
- Krantz J, Banerjee S, Zhu W, Corso J, Anderson P, Lee S and Thomason J (2023) Iterative vision-and-language navigation.

- In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 14921–14930.
- Krantz J, Lee S, Malik J, Batra D and Chaplot DS (2022) Instance-specific image goal navigation: Training embodied agents to find object instances.
- Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih Wt, Rocktäschel T, Riedel S and Kiela D (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Larochelle H, Ranzato M, Hadsell R, Balcan M and Lin H (eds.) *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., pp. 9459–9474.
- Li J, Li D, Savarese S and Hoi S (2023) BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S and Scarlett J (eds.) *Proceedings of the 40th International Conference on Machine Learning, Proceedings of Machine Learning Research*, volume 202. PMLR, pp. 19730–19742.
- Liu H, Li C, Wu Q and Lee YJ (2023a) Visual instruction tuning.
- Liu J, Shen D, Zhang Y, Dolan B, Carin L and Chen W (2022) What makes good in-context examples for GPT-3? In: Agirre E, Apidianaki M and Vulić I (eds.) *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Dublin, Ireland and Online: Association for Computational Linguistics, pp. 100–114.
- Liu S, Zeng Z, Ren T, Li F, Zhang H et al. (2023b) Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*.
- Long Y, Cai W, Wang H, Zhan G and Dong H (2025) Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. In: Agrawal P, Kroemer O and Burgard W (eds.) *Proceedings of The 8th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 270. PMLR, pp. 2049–2060.
- Maggio D, Chang Y, Hughes N, Trang M, Griffith D et al. (2024) Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters* 9(10): 8921–8928. DOI: 10.1109/LRA.2024.3451395.
- Majumdar A, Aggarwal G, Devnani B, Hoffman J and Batra D (2022) Zson: Zero-shot object-goal navigation using multimodal goal embeddings. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K and Oh A (eds.) *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., pp. 32340–32352.
- Majumdar A, Xia F, Ichtter B, Batra D and Guibas L (2023) Findthis: Language-driven object disambiguation in indoor environments. In: Tan J, Toussaint M and Darvish K (eds.) *Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 229. PMLR, pp. 1335–1347.
- Maksymets O, Cartillier V, Gokaslan A, Wijmans E, Galuba W et al. (2021) Thda: Treasure hunt data augmentation for semantic navigation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 15374–15383.
- Mascaro R and Chli M (2024) Scene representations for robotic spatial perception. *Annual Review of Control, Robotics, and Autonomous Systems*.
- Minderer M, Gritsenko A, Stone A, Neumann M, Weissenborn D et al. (2022) Simple open-vocabulary object detection. In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*. Berlin, Heidelberg: Springer-Verlag, p. 728–755.
- Mousavian A, Toshev A, Fišer M, Koščeká J, Wahid A and Davidson J (2019) Visual representations for semantic target driven navigation. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE Press, p. 8846–8852.
- OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L et al. (2024) Gpt-4 technical report.
- Oquab M, Darcet T, Moutakanni T, Vo HV, Szafraniec M et al. (2023) Dinov2: Learning robust visual features without supervision.
- Peng S, Genova K, Jiang C, Tagliasacchi A, Pollefeys M and Funkhouser T (2023) Openscene: 3d scene understanding with open vocabularies. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 815–824.
- Qin M, Li W, Zhou J, Wang H and Pfister H (2024) LangSplat: 3D Language Gaussian Splatting. In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 20051–20060.
- Radford A, Kim JW, Hallacy C, Ramesh A, Goh G et al. (2021) Learning transferable visual models from natural language supervision. In: Meila M and Zhang T (eds.) *Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research*, volume 139. PMLR, pp. 8748–8763.
- Rajvanshi A, Sikka K, Lin X, Lee B, Chiu HP and Velasquez A (2023) Saynav: Grounding large language models for dynamic planning to navigation in new environments.
- Ramakrishnan SK, Chaplot DS, Al-Halah Z, Malik J and Grauman K (2022) Poni: Potential functions for objectgoal navigation with interaction-free learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 18890–18900.
- Ramakrishnan SK, Gokaslan A, Wijmans E, Maksymets O, Clegg A et al. (2021) Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Rana K, Haviland J, Garg S, Abou-Chakra J, Reid I and Suenderhauf N (2023) Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. In: Tan J, Toussaint M and Darvish K (eds.) *Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 229. PMLR, pp. 23–72.
- Ranganathan A and Dellaert F (2011) Online probabilistic topological mapping. *The International Journal of Robotics Research* 30(6): 755–771.
- Ravichandran Z, Murali V, Tzes M, Pappas GJ and Kumar V (2025) Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments. *International Conference on Robotics and Automation (ICRA)*.
- Ravichandran Z, Peng L, Hughes N, Griffith JD and Carlone L (2022) Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using

- graph neural networks. In: *2022 International Conference on Robotics and Automation (ICRA)*. pp. 9272–9279.
- Ray A, Bradley C, Carlone L and Roy N (2024) Task and motion planning in hierarchical 3d scene graphs.
- Ren AZ, Clark J, Dixit A, Itkina M, Majumdar A and Sadigh D (2024) Explore until confident: Efficient exploration for embodied question answering.
- Rosinol A, Violette A, Abate M, Hughes N, Chang Y et al. (2021) Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research* 40(12-14): 1510–1546.
- Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences* 93(4): 1591–1595.
- Shah D, Equi MR, Osiński B, Xia F, Ichter B and Levine S (2023a) Navigation with large language models: Semantic guesswork as a heuristic for planning. In: Tan J, Toussaint M and Darvish K (eds.) *Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 229. PMLR, pp. 2683–2699.
- Shah D, Sridhar A, Dashora N, Stachowicz K, Black K et al. (2023b) Vint: A foundation model for visual navigation. In: Tan J, Toussaint M and Darvish K (eds.) *Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 229. PMLR, pp. 711–733.
- Shen W, Yang G, Yu A, Wong J, Kaelbling LP and Isola P (2023) Distilled feature fields enable few-shot language-guided manipulation. In: Tan J, Toussaint M and Darvish K (eds.) *Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 229. PMLR, pp. 405–424.
- Singh I, Blukis V, Mousavian A, Goyal A, Xu D et al. (2023) Progprompt: Generating situated robot task plans using large language models. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 11523–11530.
- Sridhar A, Shah D, Glossop C and Levine S (2023) NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. *arXiv pre-print* .
- Strader J, Hughes N, Chen W, Speranzon A and Carlone L (2024) Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies. *IEEE Robotics and Automation Letters* .
- Szot A, Clegg A, Undersander E, Wijmans E, Zhao Y et al. (2021) Habitat 2.0: Training home assistants to rearrange their habitat. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang P and Vaughan JW (eds.) *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., pp. 251–266.
- Touvron H, Martin L, Stone K, Albert P, Almahairi A et al. (2023) Llama 2: Open foundation and fine-tuned chat models.
- Wahid A, Stone A, Chen K, Ichter B and Toshev A (2021) Learning object-conditioned exploration using distributed soft actor critic. In: Kober J, Ramos F and Tomlin C (eds.) *Proceedings of the 2020 Conference on Robot Learning, Proceedings of Machine Learning Research*, volume 155. PMLR, pp. 1684–1695.
- Wandzel A, Oh Y, Fishman M, Kumar N, Wong LL and Tellex S (2019) Multi-object search using object-oriented pomdps. In: *2019 International Conference on Robotics and Automation (ICRA)*. pp. 7194–7200.
- Wang J, He G and Kantaros Y (2025) Probabilistically correct language-based multi-robot planning using conformal prediction. *IEEE Robotics and Automation Letters* 10(1): 160–167.
- Wani S, Patel S, Jain U, Chang A and Savva M (2020) Multion: Benchmarking semantic map memory using multi-object navigation. In: Larochelle H, Ranzato M, Hadsell R, Balcan M and Lin H (eds.) *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., pp. 9700–9712.
- Werby A, Huang C, Büchner M, Valada A and Burgard W (2024) Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. *Robotics: Science and Systems* .
- Wu SC, Wald J, Tateno K, Navab N and Tombari F (2021) Scenegrphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 7515–7525.
- Xia F, R Zamir A, He ZY, Sax A, Malik J and Savarese S (2018) Gibson env: real-world perception for embodied agents. In: *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, pp. 9068–9079.
- Xie Q, Min SY, Ji P, Yang Y, Zhang T, Bajaj A, Salakhutdinov R, Johnson-Roberson M and Bisk Y (2024) Embodied-rag: General non-parametric embodied memory for retrieval and generation.
- Yadav K, Ramakrishnan SK, Turner J, Gokaslan A, Maksymets O et al. (2022a) Habitat challenge 2022. <https://aihabitat.org/challenge/2022/>.
- Yadav K, Ramrakhya R, Ramakrishnan SK, Gervet T, Turner J, Gokaslan A, Maestre N, Chang AX, Batra D, Savva M et al. (2022b) Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633* .
- Yamauchi B (1997) A frontier-based approach for autonomous exploration. In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. pp. 146–151.
- Yang L, Kang B, Huang Z, Xu X, Feng J and Zhao H (2024) Depth anything: Unleashing the power of large-scale unlabeled data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 10371–10381.
- Ye J, Batra D, Das A and Wijmans E (2021) Auxiliary tasks and exploration enable objectgoal navigation. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 16097–16106.
- Yokoyama N, Ha S, Batra D, Wang J and Bucher B (2024) Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 42–48.
- Yu B, Kasaei H and Cao M (2023) L3mvm: Leveraging large language models for visual target navigation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 3554–3560.
- Zhang B and Soh H (2024) Extract, define, canonicalize: An llm-based framework for knowledge graph construction. *arXiv preprint arXiv:2404.03868* .
- Zhao Z, Lee WS and Hsu D (2023) Large language models as commonsense knowledge for large-scale task planning. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M and Levine S (eds.) *Advances in Neural Information Processing Systems*,

volume 36. Curran Associates, Inc., pp. 31967–31987.

Zheng HS, Mishra S, Chen X, Cheng HT, Chi EH, Le QV and Zhou D (2024) Take a step back: Evoking reasoning via abstraction in large language models. In: Kim B, Yue Y, Chaudhuri S, Fragkiadaki K, Khan M and Sun Y (eds.) *International Conference on Representation Learning*, volume 2024. pp. 20279–20316.

Zheng K, Paul A and Tellex S (2023) A system for generalized 3d multi-object search. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1638–1644.

Zheng K, Sung Y, Konidaris GD and Tellex S (2020) Multi-resolution pomdp planning for multi-object search in 3d. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Zhou G, Hong Y and Wu Q (2023) Navgpt: Explicit reasoning in vision-and-language navigation with large language models.

## A Mapper details

### A.1 Image Parser prompts

#### VQA Prompt 1: Prompt for Identifying Place Type

Which of the place types [`<PlaceClass1>`, `<PlaceClass2>`, ... ] best describes the scene?

#### VQA Prompt 2: Prompt for Place Labelling

What `<PlaceClass>` are we in?

#### VQA Prompt 3: Prompt for Object/Connector Appearance Description

What color is the `<object>/<connector>` in the image?

What material is the `<object>/<connector>` made of?

#### LLM Prompt 1: Templated prompt for Scene Element Classification

##### Few-shot prompts

`<Few-shot prompts follow format of query shown below. Examples of few-shot prompts given in LLM Prompt 1>`

##### Query

We observe the following: ["livingroom\_0", "window\_13", ...]. Please eliminate redundant strings in the element from the list and classify them into `<PlaceClass>`, `<ConnectorClass>`, Object classes.

##### Answer:

`<PlaceClass>`: ...  
`<ConnectorClass>`: ...  
 object: ...

#### LLM Prompt 2: Example interaction with LLM for Scene Element Classification

##### Few-shot prompts

There is a list: ["bathroom\_0", "bathroom mirror\_1", "bathroom sink\_2", "toilet\_3", "bathroom bathtub\_4", "lamp\_5", "ceiling\_10"]. Please eliminate redundant strings in the element from the list and classify them into "room," "entrance," and "object" classes. Ignore floor, ceiling and wall .

##### Answer:

room: bathroom\_0  
 entrance: none  
 object: mirror\_1, sink\_2, toilet\_3, bathtub\_4, lamp\_5

...

##### Example query

We observe the following: ["livingroom\_0", "window\_13", "door\_2", "doorway\_3", "table\_4", "chair\_5", "livingroom sofa\_6", "floor\_7", "wall\_8", "doorway\_9", "stairs\_10", "tv\_16", "stool\_17", "couch\_18", "remote\_19"]. Please eliminate redundant strings in the element from the list and classify them into room, doorway, object classes .

##### Example LLM reply

##### Answer:

room: livingroom\_0  
 entrance: door\_2, doorway\_3, doorway\_9  
 stair: stairs\_10  
 object: table\_4, chair\_5, sofa\_6, window\_13, tv\_16, stool\_17, couch\_18, remote\_19

### A.2 State Estimator prompts

#### LLM Prompt 3: Templated prompt for Place Label Similarity

**Question:** In the scene, there are ["kitchen\_1", "livingroom\_1", "livingroom\_2", "bedroom\_1", "familyroom\_2"]. Now, I need to find a "livingroom". Please give me all the `<PlaceClass>` with similar semantic meaning from the list. Please directly return the answer in a Python list. Follow the format: Answer: `<your answer>`."

#### LLM Prompt 4: Example interaction with LLM for Place Label Similarity

##### Example query

**Question:** In the scene, there are ["kitchen\_1", "livingroom\_1", "livingroom\_2", "bedroom\_1", "familyroom\_2"]. Now, I need to find a "livingroom". Please give me all the rooms with similar semantic meaning from the list. Please directly return the answer in a Python list. Follow the format: Answer: <your answer>."

**Example LLM reply**

Answer: livingroom\_1, livingroom\_2, familyroom\_2.

**Question:** These are depictions of what I observe from two different vantage points. Please assess the shared objects and spatial relationship in the descriptions to determine whether these two positions are indeed in the same place. Provide a response of True or False, along with supporting reasons. In each direction, focus on only two to three large objects for reasoning.

**Answer:** \_

**Reasoning:** \_

### LLM Prompt 5: Templated prompt for Pairwise Place Matching

**Context**

You are a robot exploring an environment for the first time. You will be given an object to look for and should provide guidance on where to explore based on a series of observations. Observations will be given as descriptions of objects seen from four cameras in four directions. Your job is to estimate the robot's state. You will be given two descriptions, and you need to decide whether these two descriptions describe the same *<PlaceClass>*. For example, if we have visited the *<PlaceClass>* before and got one description, when we visit a similar *<PlaceClass>* and get another description, it is your job to determine whether the two descriptions represent the same *<PlaceClass>*. You should understand that descriptions may contain errors and noise due to sensor noise and partial observability. Always provide reasoning along with a deterministic answer. If there are no suitable answers, leave the space after 'Answer: None.' Always include: Reasoning: <your reasoning> Answer: <your answer>.

**Few-shot prompts**

<Few-shot prompts follow format of query shown below. Examples of few-shot prompts given in LLM Prompt 6>

**Query**

**Description1:** I can see a brown wood headboard, ...

**Description2:** I can see a white wooden door, ...

### LLM Prompt 6: Example interaction with LLM for Pairwise Place Matching

**Context**

You are a robot exploring an environment for the first time...

**Few-shot prompts**

**Description1:** On the left, I can see a brown wood headboard, white paper pillow. On the right, I can see a black metal television, gray plastic laundry basket, white wood closet dresser, brown wood drawer. In front of me, I can see a white wood bed, white wood window, brown metal lamp, brown wood dresser, brown wood dresser nightstand, black silk curtain, white plastic curtain, white metal wall lamp, brown wood drawer. Behind me, I can see a brown wood cabinet.

**Description2:** On the left, I can see a white wood door. On the right, I can see a white wood bed, white glass lamp, white glass window, white plastic curtain, brown wood dresser nightstand, white glass window, white wood nightstand, blue fabric curtain, white cotton pillow, white metal ceiling fan, silver metal wall lamp. In front of me, I can see a white glass lamp, brown wood headboard, white cotton pillow, brown wood dresser, white wood bed, white cotton pillow, red metal wall lamp door, brown wood drawer. Behind me, I can see a brown wood bureau, black glass television, brown wood stool, brown wood drawer, brown wood drawer, brown wood drawer.

**Question:** These are depictions of what I observe from two different vantage points. Please assess the shared objects and spatial relationship in the descriptions to determine whether these two positions are indeed in the same place. Provide a response of True or False, along with supporting reasons. In each direction, focus on only two to three large objects for reasoning.

**Answer:** True

**Reasoning:** To simplify the description, given the abundance of objects, we initially focus on common perceptions of object sizes. Our attention is directed toward larger objects, as these are less prone to detection errors.

Description 1: On the left, there is brown wood headboard, white paper pillow. On the right, black metal television, brown wood dresser, and white wood closet dresser are of relatively large size. In front of me, there is a white wood bed and a brown wood dresser. Behind me, I can see a brown wood cabinet and black silk curtain.

Description 2: On the left, there is white wood door. On the right, the white wood bed, brown wood dresser, nightstand, and ceiling fan are of relatively large size. In front of me, there is a white wood bed, black glass television a brown wood dresser. Behind me, I can see a brown wood cabinet.

Shared Large Objects: the two descriptions exhibit significant commonalities, prominently featuring large and easily observable items such as a brown wood dresser, a brown wood drawer, a white wood bed, a metal wall lamp, a television, and a brown wood headboard. Spatial Relationship: the spatial relationships within both descriptions remain consistent, with the dresser and wall lamp positioned near the bed in each scenario. Despite minor variations in the color or material of smaller objects like stools or curtains, these discrepancies appear more likely to stem from observational nuances rather than indicating distinct rooms.

...

#### Example query

**Description1:** On the left, there is a silver metal faucet, white glass mirror, black metal wall lamp, white stainless steel sink, white formica countertop, blue plastic accessory, white wood white door, silver metal faucet, white metal wall lamp, brown wood cabinet, black wood white door doorway, brown wood cabinet, white wood mirror wall. On the right, there is a brown tile floor, white drywall ceiling. In front of me, there is a silver metal faucet, silver glass mirror, white soap soap, white porcelain sink, brown wood cabinet, white porcelain sink, silver metal wall lamp, silver metal wall lamp. At the rear, there is an orange glass lamp, brown wood floor, white porcelain tub, white cotton bed.

**Description2:** On the left, there is a white glass mirror, silver metal faucet, white porcelain sink, white wood bathroom sink, white wood bed, brown metal wall lamp. On the right, there are no specified items. In front of me, there is a silver metal faucet, white porcelain sink, white plastic toiletry, silver glass mirror, blue plastic soap toiletry, white white bathroom sink countertop, blue metal wall lamp, black wood bathroom cabinet, black metal wall lamp. Behind me, there is a white porcelain tub, black cloth curtain, white cotton bed, brown wood bed, black cloth curtain, white porcelain bath.

**Question:** These are depictions of what I observe from two different vantage points. Please assess the shared objects and spatial relationship in the descriptions to determine whether these two positions are indeed in the same place. Provide a response of True or False, along with supporting reasons. In each direction, focus on only two to three large objects for reasoning.

#### Example LLM reply

**Answer:** True

**Reasoning:** To simplify the description, given the abundance of objects, we focus initially on common perceptions of larger objects, as they are less prone to detection errors.

Description 1: On the left, there is brown wood cabinet, silver metal faucet, white wood bathroom sink, and white wood bed. On the right, there is nothing. In front of me, there is a silver metal faucet, silver glass mirror, white porcelain sink, brown wood cabinet, black metal wall lamp. Behind me, there is an orange glass lamp, white porcelain tub, white cotton bed.

Description 2: On the left, there is white porcelain sink and white wood bed. On the right, there are nothing. In front of me, there is a silver metal faucet and white plastic toiletry. Behind me, I can see white porcelain tub, black cloth curtain, white cotton bed.

Shared Large Objects: Both descriptions mention silver metal faucets, white beds, and sink, contributing to a consistent thematic presence. Spatial Relationship: The placement of key items like silver metal faucets and brown wood cabinets described in similar configurations create a sense of coherence. Potential for Small Errors: Description 2 lacks specified items, such as accessory, door and countertop. Besides, there are small errors in the material of bed. Errors in material or color, as well as the potential omission of small-size objects, can indeed be attributed to variations in observation. Despite this, the two descriptions are depicting the same room, with the potential for small errors accounting for minor variations.

### A.3 OSG Updater prompts

#### LLM Prompt 7: Example interaction with LLM for Object Data Association

##### Few-shot prompts

**Object to be matched:** We want to find a doorframe that is near a tv, a chair and a stool.

**List of observed objects:** Now we have seen the following object: doorframe\_2 that is near chair and sofa. doorframe\_3 that is near a tv and a chair. wooden door\_2 that is near table, sink and lamp.

**Question:** Please select one object that is most likely to be the object I want to find. Please only select one object in the list and use this element name in answer. Use the exact name in the given sentences. Always follow the format: Answer: <your answer>."

**Answer:** doorframe\_3

**Reasoning:** Among the given objects, "doorframe\_3" is mentioned to be near a TV and a chair, most likely meeting the specified criteria of being near a TV, chair, and stool.

...

##### Example query

**Object to be matched:** We want to find a door that is near a dining table, window glass and a table cloth.

**List of observed objects:** Now we have seen the following object: door\_2 that is near table and window glass. doorframe\_4 that is near a table cloth and a glass. door\_4 that is near photo, sofa and windows.

**Question:** Please select one object that is most likely to be the object I want to find. Please only select one object in the list and use this element name in answer. Use the exact name in the given sentences. Always follow the format: Answer: <your answer>."

##### Example LLM reply

**Answer:** door\_2

**Reasoning:** Among the given objects, 'door\_2' is mentioned to be near a table and window glass. Although 'doorframe\_4' is also near a tablecloth and a glass, it seems to meet the criteria. However, since the target goal is a door and not a doorframe, 'door\_2' is more suitable.

#### LLM Prompt 8: Templated prompt for Inferring Region Abstractions

Previously, we were in a <AbstractionClass> containing the <PlaceClass>, and we move towards <Subgoal>. Now we arrive in a <PlaceClass>. Do you think the current state <PlaceClass> belongs to [list of existing <AbstractionClass>s]?

If it belongs to any other existing region abstraction, return the region abstraction name; otherwise propose the name for this new <AbstractionClass> formatted as "<name> <AbstractionClass> (New)". Ensure that your response follows the format:  
Reasoning: <your reasoning>.  
Answer: <your answer>

#### LLM Prompt 9: Example interaction with LLM for Inferring Region Abstractions

##### Example query

Previously, we were in the Cooked Food Section containing the Deli Aisle, and we move towards Freezer. Now we arrive in a Dairy Aisle. Do you think the current state Dairy Aisle belongs to any of the existing region abstractions: ["Cooked Food Section", "Clothing Section", "Dry Goods Section"]?

If it belongs to any other existing region abstraction, return the region abstraction name; otherwise propose the name for this new section formatted as "<name> Section (New)". Ensure that your response follows the format:  
Reasoning: <your reasoning>.  
Answer: <your answer>

**Example LLM reply**

Reasoning: The Dairy Aisle typically contains products such as milk, cheese, yogurt, and butter, which are perishable food items. Given its nature as a section primarily housing food products, it doesn't fit within the existing region abstractions provided ("Cooked Food Section", "Clothing Section", "Dry Goods Section").

Answer: Perishable Food Section (New)

**B Reasoner details****B.1 Region Proposer prompts****LLM Prompt 10: Templated prompt for Region Proposal****Few-shot prompts**

<Few-shot prompts follow format of query shown below. Examples of few-shot prompts given in **LLM Prompt 11**>

**Query**

**Open Scene Graph:** You see the partial layout of the environment: {"room": {"livingroom\_1": ...

**Goal:** Your goal is to find a <Goal> object.

**Question:** If any of the <LayerType> in the layout are likely to contain the target object, specify the most probable <LayerType> name. If all the <LayerType> are not likely to contain the target object, provide the <ConnectorClass> (that could connect <LayerType>) you would select for exploring a new <LayerType> where the target object might be found.

**Answer:** \_

**Reasoning:** \_

**LLM Prompt 11: Example interaction with LLM for Region Proposal****Few-shot prompts****Open Scene Graph:**

You see the partial layout of the apartment: {"room": {"kitchen\_1": {"connects to": ["stair\_1"]}, "livingroom": {"connects to": ["stair\_1"]}, "entrance": {"stair\_1": {"is near": []}}}}

**Goal:** Your goal is to find a sink.

**Question:** If any of the rooms in the layout are likely to contain the target object, specify the most probable room name. If all the room are not likely contain the target object, provide the door you would select for exploring a new room where the target object might be found.

**Answer:** kitchen\_1

**Reasoning:** There are kitchen and livingroom in the layout. Among all the rooms, kitchen is usually likely to contain sink. Since we haven't explored the kitchen yet, it is possible that the sink is in the kitchen. Therefore, I will explore kitchen.

**Open Scene Graph:**

You see the partial layout of the apartment: {"room": {"livingroom\_1", "connects to": ["door\_1", "door\_2"]}, "diningroom\_1": {"connects to": ["door\_1"]}, "entrance": {"door\_1": {"is near": ["towel\_1"], "connects to": ["livingroom\_1", "diningroom\_1"]}, "door\_2": {"is near": [], "connects to": ["livingroom\_1"]}}}

**Goal:** Your goal is to find a sink.

**Question:** If any of the rooms in the layout are likely to contain the target object, specify the most probable room name. If all the room are not likely contain the target object, provide the door you would select for exploring a new room where the target object might be found.

**Answer:** door\_1

**Reasoning:** There is only livingroom in the layout. livingroom is not likely to contain sink, so I will not explore the current room. Among all the doors, door1 is near to towel. A towel is usually more likely to near the bathroom or kitchen, so it is likely that if you explore door1 you will find a bathroom or kitchen and thus find a sink.

...

**Example query****Open Scene Graph:**

You see the partial layout of the environment: {"room": {"livingroom\_1": {"connects to": ["doorway\_1", "door\_2"]}, "entrance": {"doorway\_1": {"is near": ["table\_1"]}, "door\_2": {"is near": ["clock\_1"], "connects to": ["livingroom\_1"]}}}}

**Goal:** Your goal is to find a oven.

**Question:** If any of the rooms in the layout are likely to contain the target object, specify the most probable room name. If all the room are not likely contain the target object, provide the door you would select for exploring a new room where the target object might be found.

**Example LLM reply****Answer:** doorway\_1

**Reasoning:** There are only livingroom in the layout. Among all the rooms, livingroom is usually unlikely to contain an oven, making it less likely for me to find an oven in the current room. Instead, I plan to explore other rooms connected to the current living room via entrances. Evaluating the entrances, doorway1 stands out as it is close to a table. Tables are commonly found in kitchens, which often contain ovens. Therefore, I have decided to explore through doorway\_1.

There is a list: ["mirror\_2", "lamp\_1", "picture\_7", "tool\_6", "toilet\_8", "sofa\_11", "floor\_12", "wall\_13"].

**Question:** Please select one object that is most likely located near a sink. Always follow the format: Answer: <your answer>. Reasoning: <your Reasoning>.

**Example LLM reply****Answer:** mirror\_2

**Reasoning:** Among the given options, the object most likely located near a sink is a "mirror." Mirrors are commonly found near sinks in bathrooms for personal grooming and hygiene activities.

## B.2 Object Proposer prompts

**LLM Prompt 12: Templated prompt for Object Proposal****Few-shot prompts**

<Few-shot prompts follow format of query shown below. Examples of few-shot prompts given in **LLM Prompt 13**>

**Query**

**Objects in current Place node:** There is a list: ["mirror\_2", "lamp\_1", ...].

**Question:** Please select one object that is most likely located near a <Goal>. Always follow the format: Answer: <your answer>.

**Answer:** \_**Reasoning:** \_**LLM Prompt 13: Example interaction with LLM for Object Proposal****Few-shot prompts****Objects in current Place node:**

There is a list: ["chair\_4", "sofa\_2", "bed\_9", "dresser\_1", "ceiling\_6", "closet\_5", "window\_7", "wall\_10"].

**Question:** Please select one object that is most likely located near a table. Always follow the format: Answer: <your answer>. Reasoning: <your Reasoning>.

**Answer:** chair\_4

**Reasoning:** Among the given options, the object most likely located near a table is a "chair." Chairs are commonly placed around tables for seating during various activities such as dining, working, or socializing. ...

**Query****Objects in current Place node:**

## C Automatic OSG generation details

### C.1 Description generation prompts

**LLM Prompt 14: Templated prompt for Environment Description Generation****Context**

You are a helpful system. Given an environment type, you are required to describe the environment layout based on the meta structure.

In meta structure, we define 4 abstract types of classes to describe the environment.

1. Objects. Static scene elements occupying spatially localised regions: e.g. tables, chairs etc. They provide semantic cues for reasoning, and act as navigation goals and localisation landmarks - specifically, we distinguish Objects/Places with object features, aggregated lists of nearby objects.
2. Places. The smallest semantically meaningful spatial regions in a scene, e.g. rooms in a home or aisles in a supermarket, which typically cannot be subdivided in a consistent, meaningful way.
3. Region Abstractions. Spatial abstractions encompassing multiple smaller spatial regions: e.g. a floor containing multiple rooms, or a supermarket section that is a cluster of aisles.
4. Connectors. These are specific types of Objects that are also structural scene elements connecting spatial regions: e.g. doors connecting rooms, steps connecting floors.

**Query**

We are using text to describe the environment layout. Please follow the meta-structure to describe the environment.

Based on the example, please return the text to describe `<EnvironmentLabel>`'s layout. Object is always required. Follow the response format: Text: `<your text>`.

### LLM Prompt 15: Example interaction with LLM for Environment Description Generation

#### Context

You are a helpful system. Given an environment type...

#### Query

We are using text to describe the environment layout. Please follow the meta-structure to describe the environment.

Example:

Environment Type: household

Text: In the household environments, there are often three layers including room, entrance and object. Home contains various rooms. The room contains spatial regions and is Places in meta structure. Rooms can be connected to rooms or entrance. Entrances are Connectors in meta structure. Rooms contain objects. Entrance and Objects is near objects.

Based on the example, please return the text to describe `<EnvironmentLabel>`'s layout. Object is always required. Follow the response format: Text: `<your text>`.

#### Feedback

The text you generate contains the error: `<MetastructureErrorMessage>`. Please fix this issue and re-generate the text to describe `<EnvironmentLabel>` layout. Object is always required. Follow the response format: Text: `<your text>`.

We use triplets to structure the text. Triplets follow the template: [Subject, Relation, Object].

Example:

Text: In the household environments, there are often three layers including room, entrance and object. Home contains various rooms. The room contains spatial regions and is Places in meta structure. Rooms can be connected to rooms or entrance. Entrances are Connectors in meta structure. Rooms contain objects. Entrance and Objects is near objects.

Triplets:

```
[room, contains, object]
[room, connects, entrance]
[room, connects, room]
[entrance, is near, object]
```

Based on the example, please return the triplets of `<EnvironmentLabel>` environment. Only allow `<MetastructureRelationTypes>` these relations. Follow the format: Triplets: `<your triplets>`. Text: `<text_description>`.

### LLM Prompt 17: Example interaction with LLM for Triplet Extraction

#### Few-shot prompts

**Question:** We use triplets to structure the text. Triplets follow the template: [Subject, Relation, Object].

Given the description of household environment: In the household environments, there are often three layers including room, entrance and object. Home contains various rooms. The room contains spatial regions and is Places in meta structure. Rooms can be connected to rooms or entrance. Entrances are Connectors in meta structure. Rooms contain objects. Entrance and Objects is near objects. Please return the triplets of the household environment. Follow the format: Triplets: `<your triplets>`.

Triplets:

```
[room, contains, object]
[room, connects, entrance]
[room, connects, room]
[entrance, is near, object]
```

#### Query

We use triplets to structure the text. Triplets follow the template: [Subject, Relation, Object].

Example:

## C.2 Canonicalisation prompts

### LLM Prompt 16: Templated prompt for Triplet Extraction

#### Few-shot prompts

`<Few-shot prompts follow format of query shown below. Examples of few-shot prompts given in LLM Prompt 17>`

#### Query

Text: In the household environments, there are often three layers including room, entrance and object. Home contains various rooms. The room contains spatial region and is Places in meta structure. Rooms can be connected to rooms or entrance. Entrances are Connectors in meta structure. Rooms contains objects. Entrance and Objects is near objects. Triplets:  
 [room, contains, object]  
 [room, connects, entrance]  
 [room, connects, room]  
 [entrance, is near, object]

Based on the example, please return the triplets of `<EnvironmentLabel>` environment. Only allow `<MetastructureRelationTypes>` these relations. Follow the format: Triplets: `<your triplets>`.

### LLM Prompt 18: Templated prompt for Triplet Canonicalisation

#### Query

Given a piece of text and a relational triplet extracted from it, and the definition of the relation in it, choose the most appropriate relation to replace it in this context if there is any and modify the triplet. If the entity is too specific, use the abstract word to replace it. In the `<EnvironmentLabel>`, `<Abstraction1>` `<Relation>` `<Abstraction2>`. Triplet: `(<Abstraction1>`, `<Relation>`, `<Abstraction2>`)

We define 4 abstract types of classes to describe the environment.

1. Objects. Static scene elements occupying spatially localised regions: e.g. tables, chairs etc. They provide semantic cues for reasoning, and act as navigation goals and localisation landmarks - specifically, we distinguish Objects/Places with object features, aggregated lists of nearby objects.
2. Places. The smallest semantically meaningful spatial regions in a scene, e.g. rooms in a home or aisles in a supermarket, which typically cannot be subdivided in a consistent, meaningful way.
3. Region Abstractions. Spatial abstractions encompassing multiple smaller spatial regions: e.g. a floor containing multiple rooms, or a supermarket section that is a cluster of aisles.

4. Connectors. These are specific types of Objects that are also structural scene elements connecting spatial regions: e.g. doors connecting rooms, steps connecting floors.

For subject entity or object entity, you should first decide the abstract class that the entity belongs to and make sure the entity name is at the right abstraction level for `<EnvironmentLabel>`. If the subject entity or object entity is too specific, use an abstract word to replace it. For example, if products are a specific kind of object and are not Connectors, we should replace them with "objects". If the object is a Connector that connects regions, such as entrances, we should not replace it with "objects". Do not use any class type name, including "Places", "Region Abstractions", and "Connectors", as the entity name in triplets except for "objects".

We define 4 relations as follows.

1. "contains": Specifies spatial containment relations among regions/objects. The subject entity should contain the object entity. The subject entity is required to be a region or a place. Example: [floor, contains, room], ['section', 'contains', 'aisle']
2. "connects to": Specifies reachability between the subject entity and object entity, and can connect Place, Region Abstraction, and Connectors. Object cannot "connects to" any other items. Example: [room, connects, entrance], [room, connects to, room]
3. "is near": It approximately capture closeness between the subject entity and object entity. The subject entity must be Connector and the object entity must be objects. Example: [entrance/door, is near, object]
4. "invalid": If the relation is none of the above, it is invalid. For example, the object cannot be near a place or a region.

For example, in household environment, given [place, has, desk], we should modify it to [room, contains, object]. In household environment, given [connector, connects to, rooms], we should modify it to [entrance, connects to, room].

Please follow the format: Answer: `<modified triplets>`.

## D System details

### D.1 System hyperparameters

In Table 10, we list the various modules in the system, describe the hyperparameters used in each and provide the values used in OSG Navigator for our experiments. The Mapper’s and Reasoner’s hyperparameters are kept the same across both simulation and real-world tests, and across robot platforms in the real world. Some of the thresholds are sensor-dependent and can benefit from tuning for a specific sensor - *i.e.* centroid nearness and minimum object size thresholds that are expressed in pixel-based units, and hence depend on image resolution or field-of-view. In practice, our centroid nearness threshold is defined to be large enough that sufficient object features can be accumulated when using both RGB sensors in simulation and on our robots in the real world.

## E Additional experimental results

### E.1 Experimental results and details for OSG mapping

#### OSG Schema 6: Supermarket

```
Aisle:
  layer_type: Place
  layer_id: 2
  contains: Object
  connects_to: Aisle
Object:
  layer_id: 1
```

#### OSG Schema 7: Office

```
Floor:
  layer_type: Region
  layer_id: 3
  contains: Hallway, Office, Room
  connects_to: Stairs
Hallway:
  layer_type: Place
  layer_id: 2
  contains: Object
  connects_to: Hallway, Office, Room,
  Entrance
Office:
  layer_type: Place
  layer_id: 2
  is_near: Object
  connects_to: Hallway, Office, Room,
  Entrance
Room:
  layer_type: Place
  layer_id: 2
  is_near: Object
  connects_to: Hallway, Office, Room,
  Entrance
Stair:
  layer_type: Connector
  layer_id: 2
  is_near: Object
  connects_to: Floor
Entrance:
```

```
layer_type: Connector
layer_id: 2
is_near: Object
connects_to: Hallway, Office, Room
Object:
  layer_id: 1
```

#### OSG Schema 8: Mall

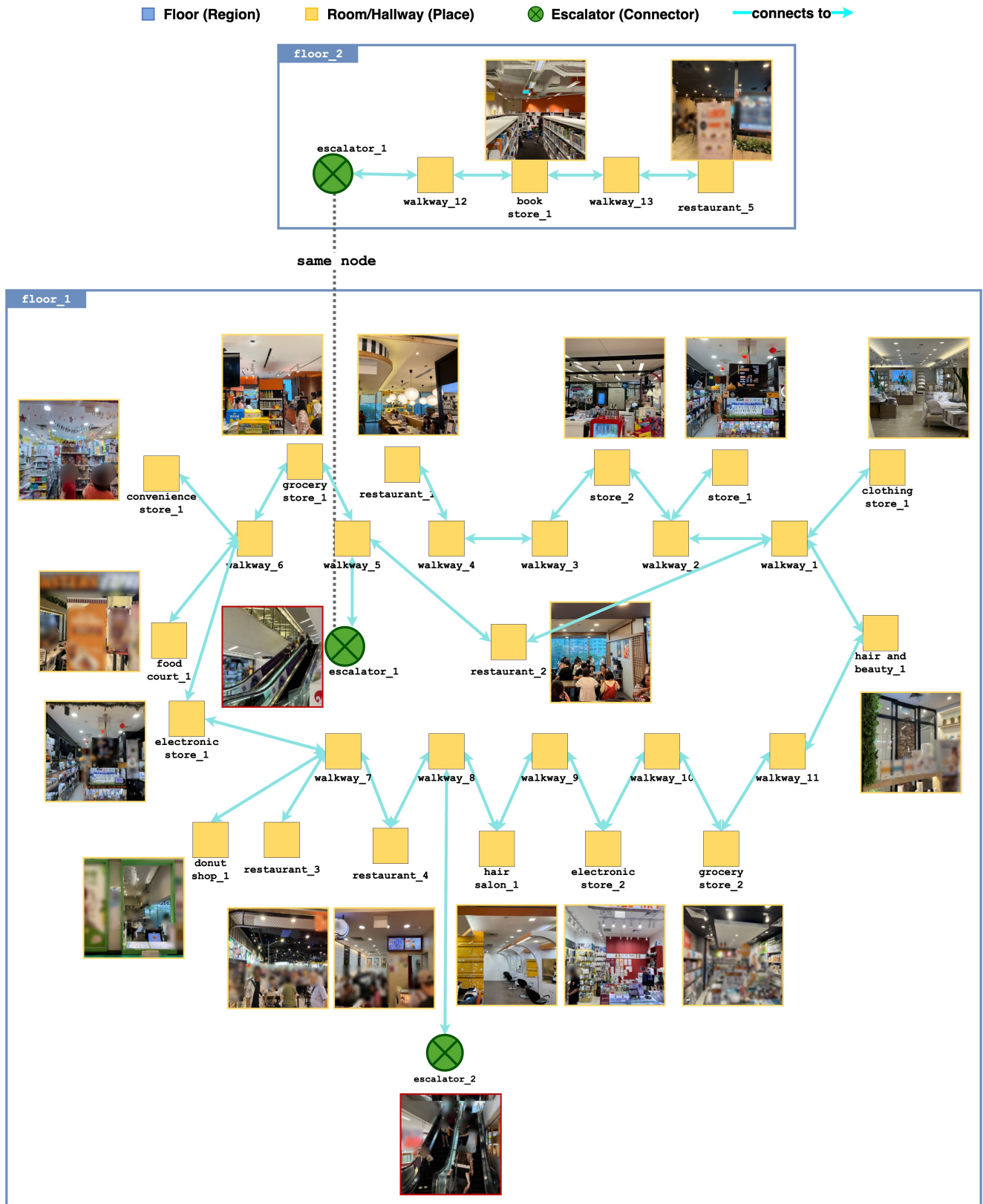
```
Floor:
  layer_type: Region
  layer_id: 3
  contains: Walkway, Store
  connects_to: Escalator
Walkway:
  layer_type: Place
  layer_id: 2
  contains: Object
  connects_to: Walkway, Store, Escalator
Store:
  layer_type: Place
  layer_id: 2
  is_near: Object
  connects_to: Store, Walkway
Escalator:
  layer_type: Connector
  layer_id: 2
  is_near: Object
  connects_to: Floor, Walkway
Object:
  layer_id: 1
```

Table 10. Hyperparameters used in the OSG Navigator system.

Subsystem	Hyperparameter	Value
Mapper	Centroid nearness $\beta_{pix}$ (object nearness heuristic)	100 (pixels)
	Bounding box overlap $\beta_{IOU}$ (object nearness heuristic)	0.1 (IoU)
	Min. object size threshold	200 (pixels <sup>2</sup> )
	LLM temperature	0.3
Reasoner	LLM temperature	0.3
Controller (Simulation - FMM)	Occupancy map resolution	0.05 (m)
	Occupancy map size	4.8 (m)
	Min. obstacle height	0.5 (m)
	Turning action angle	30 (deg)
	Forward action distance	0.25 (m)
Controller (Real-world - GNM)	Stopping threshold	2.25
	Max linear velocity	0.5 (m/s)
	Max angular velocity	0.3 (rad/s)



Figure 11. Spatial concepts from Mall/Office OSG specs. Full OSG specifications using these concepts are given in [OSG Schema 7](#) and [OSG Schema 8](#).



**Figure 12. OSG of real-world Mall environment.** The spatial concepts used to describe this *walkways* and *stores* for Places, and *escalators* for Connectors. We provide the full OSG specification in [OSG Schema 8](#).