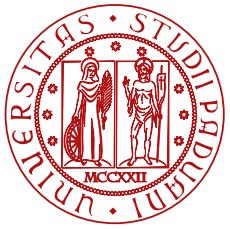

Università degli Studi di Padova

Dipartimento di Fisica e Astronomia Galileo Galilei

Physics of Data

Zero-Shot Object Goal Navigation using Online Image Retrieval

Master Candidate

Jelin Raphael Akkara

Student ID 2072064

Supervisor

Prof. Lamberto Ballan

University of Padova

Co-supervisor

Dr. Tommaso Campari

Fondazione Bruno Kessler

Co-supervisor

Filippo Ziliotto

University of Padova

Academic Year
2024/2025

To my parents
and friends

Abstract

Embodied AI has advanced significantly with the advent of queryable maps, enabling agents to perceive and interact with complex environments. However, the sheer number of diverse approaches make it challenging to cross-experiment and compare, demanding time and familiarity. To ensure easy access, we develop ModNav, a modular and holistic framework for Visual SLAM mapping techniques.

Conventional methods, including ones leveraging queryable maps, often struggle to recognize rare tail-end objects. We propose a novel approach that leverages online image retrieval to provide additional contextual information about the target object, and this allows for efficient spatial localization. To experiment with long-tail categories' distribution, we also release HSSD-rare, a dataset containing a collection of 1362 episodes spanning 17 scenes, each corresponding to a long-tail target goal. Our approach, especially on the HSSD-rare dataset, shows that augmenting text-based queries with online visual data can significantly enhance localization in complex, open-set environments.

Contents

List of Figures

List of Tables

1

Introduction

Recent progress in fields like Computer Vision, Natural Language Processing, and Reinforcement Learning have been steadily building towards the collective, long-term goal of Artifical General Intelligence (AGI). The past decade has witnessed major milestones pass by: Residual Networks that made deep architectures trainable [16], AlphaGo's [34] surprising successes which showed neural networks surpassing human expertise, the Transformer model [40] with its attention-based architecture reshaped sequence learning and paved the way for the GPT series [28], and AlphFold [19] which cracked a protein-folding problem that had stalled for generations. More recently, the explosive growth of Large Language and Generative models has revealed interesting scaling laws and emergent behaviours.

A surging unifying theme across recent breakthroughs is the use-case of multi-modality, where training on diverse data types to build rich, inter-modal representations leads to better real-world performance. Embodied AI encompasses this principle in a most direct way, placing agents in novel lifelike simulations where they must observe, navigate and manipulate their surroundings using vision, language, touch and learned policies. By tackling problems like sim-to-real transfer, social interaction, object-goal navigation, and object rearrangement, the field has witnessed exciting growth and continues to progress on in its efforts towards a general, adaptive intelligence.

In recent years, the developments in Embodied AI have enabled agents to interact intelligently with their environments, utilizing queryable maps to construct semantic representations of scenes. From 2D Semantic mapping [9] to

Vision Language Frontier Mapping (VLFM) [45], the field has been populated with complex and rewarding approaches, many often not so easily accessible to start working with, and cross-experimentation remains a challenge. We develop ModNav (Modular Navigation) to address this concern, where different mapping techniques are extracted and made compatible under an accessible framework. This provides a platform for quick and efficient adaptation of a relevant scene mapping approach, which can complement agent path-planning policies.

Apart from improving accessibility, the field is still bettering itself on various tasks and benchmarks, like Object Goal Navigation, Vision Language Navigation, and so on. Contemporary approaches such as VLFM [45] and VLMaps [17] have relied on aligning textual and visual embeddings, guiding the agent with queryable maps towards the target goal. However, these methods often struggle to recognize rare "long-tail" objects, since aligning very specific categories prove to be challenging and memory constrained. To address this, we turn towards online image search engines, which offer a rich resource for retrieving visual examples. We propose IMPRINT, an approach that integrates online-sourced images into queryable maps, which could possibly enhance the capabilities of embodied agents. Upon receiving a textual query, our algorithm retrieves N related images, extracts relevant features using a Vision-Language Model (VLM) and builds a memory of the scene in the form of an embedding map. This map is then used to navigate towards a target goal by the embodied agent. Storing features contextualized by both the text and relevant image queries would be an efficient way of localizing and navigating towards long-tailed object categories. This mapping method is a lightweight VLFM-based adaptation optimized for efficient storage, significantly reducing memory requirements by 99.75% compared to the standard VLFM implementation.

For evaluation on long-tail object navigation, we require a dataset made of such rare objects situated in simulated environments. However, contemporary datasets only cater to a general class of labels, with no dataset specifically made for long-tailed category navigation. The HSSD dataset [22] contains rare and long-tailed categories, with meta information like the object dimensions, which would be optimal for our use case. However, this does not provide a navigation dataset with these categories, that is, it lacks a set of episodes and evaluation viewpoints that are required to properly evaluate an agent's performance. For this reason, we are required to generate episodes and viewpoints around target

categories, effectively creating a navigation dataset. We create HSSD-rare, a dataset curated from the HSSD dataset, and comprising of more than 1300 objects spanning 17 scenes ideal for zero-shot evaluation of queryable maps.

1.1 Contribution

Our key contributions are as follows:

- We introduce a zero-shot approach that enhances queryable maps by augmenting text queries and relevant web-searched images, improving spatial localization.
- We present a hardware-constrained method to consistently store the embedding features while mapping the environment, requiring 400x less hardware space.
- We present HSSD-rare, an ObjectNav dataset specifically for long-tail categories extracted from high-quality HSSD scenes.

In the following sections, we introduce and dive into the details regarding the above tasks and implementations. Starting off, in Section 2, we introduce the field of Embodied AI, tracing its historical roots and grounding it to its contemporary relevance by discussing the major tasks in the field, along with the state-of-the-art techniques employed currently. In Section 3, we discuss the dissertation's problem statement, along with the framework (Habitat AI) that enables one to work on this task of Object Goal Navigation. In the following sections, we cover ground on the three major contributions, providing motivation for each, discussing the challenges faced and the solutions employed to surpass them.

In Section 4, we address the issue of accessibility among mapping techniques, to which ModNav is proposed as an accessible framework. We discuss the implemented techniques, along with the challenges faced along the way. In Section 5, we generate HSSD-rare, a task-specific dataset for long-tailed categories, and we walk through the two major processes of generating viewpoints and the whole dataset. In Section 6, we improve upon baselines in the task of Object Goal Navigation by developing IMPRINT. This enables zero-shot navigation using a queryable map constructed out of online-sourced images. The pipeline, both static and dynamic, is introduced along with the major components that make up the technique. Finally, in Section 7, we detail the experiments run for both the

static and dynamic pipelines, additionally covering ablations and relevant interpretations, and we establish that IMPRINT leads to a 4-8% increase in Success Rate, including baseline performances, while reducing memory consumption by $\sim$ 99%. In Section 8, we conclude by summarizing the contributions and their individual progress in three different axes of the field, along with possible interesting directions this work can be taken further in the future.

2

Embodied AI: A Review

2.1 Advent of Embodied AI

Evolution of the field of AI Over the last decades, the landscape of Artificial Intelligence (AI) has changed drastically, starting out as a purely computational discipline and morphing into a highly interdisciplinary one at recent times. Known by the name "Good Old-Fashioned AI" (GOFAI), the field launched (around the 1950s) on the premise that intelligence can be expressed through symbolic and rule-based systems, leading to works in knowledge representations and logic programming. However, due to its difficulty handling uncertainty, and especially the knowledge acquisition bottleneck, the field failed to deliver on its ambitious promise.

In the 1980s, revival came in the form of statistical machine learning, where systems were built to learn patterns from data rather than explicitly encoded knowledge. Major developments include backpropagation [32], support vector machines [11], and Bayesian neural networks [27]. With the advent of GPUs and more capable computational resources, the field found celebrated success in narrow domains like chess programs (Deep Blue [6]), recommendation systems (GroupLens [31]) and early web search algorithms. In the 2000s, progress came in the form of deep learning architectures, with key innovations like ReLu activation [26] and better regularization techniques like dropout [35]. The current era had progressed field like computer vision and natural language processing to impressive lengths, with most contemporary models built to be general-purpose models trained on massive datasets, and later adapting them to specific tasks.

Although the field of AI has progressed in diverse ways, with implications that are far-reaching in all sectors of academics and industry, it does come with its own costs. In the mid-1980s, classical approaches were only applicable to perfect environmental conditions, like in factory settings. Specifically, embedded systems with computer vision were highly successful when the lighting conditions, the geometry of the scene and the types of potential objects could be known, but failed in the real world due to the numerous variables that could not be accounted for. This was also the case in tasks like object manipulation, scene navigation, knowledge grounding, and speech systems [29]. Notably, tasks that are perceived as more natural to the everyday human have proved to be exceptionally hard to emulate, often known as Moravec's paradox [1].

This holds true to certain degree even in this day, where the most progress achieved has been in supervised models catered to specific tasks and sub-tasks. This focus on distinct sub-task prevent a true understanding of intelligence in the real world, which is highly multi-modal and unpredictable. Although there have been general-purpose models like Llama [39], and GPT-4 [28], which have learnt beyond their designated task of sequence prediction and claimed to have developed reasoning and multi-modal emergence capabilities, recent studies have challenged this claim stating such LLMs mimic the usage of a language, be it academic or creative, while failing to perceive the reasoning or meaning behind such constructions [38]. Nonetheless, such behavior arises out of a local minima that is the convergence of multiple modal behaviours, thereby enabling the LLM to act in ways that can seem to be reasonable and rational. This paradigm was realized by a couple researchers back in the mid-1980s, campaigning for more "high-bandwidth" interactions in the AI field, which led to the emergence of Embodied AI.

Evolution of Embodied AI Embodied AI was established under the belief that to truly form general-purpose intelligent agents, they had to be exposed to the real world from the start, where they could be allowed to learn and develop solutions by interacting with the environment. Such interactions were termed as high-bandwidth, where the agent is able to sense the environment through touch sensors, or visual input, or even auditory data, while also being exposed to an ever changing environment, enabling the agent to observe causal phenomena in real time. For example, the simple act of pouring water can teach the agent about the effect of gravity, water's viscosity, the confluence of such

factors, and the consequence of its action. This is opposed to the mainstream low-bandwidth paradigm, where models are confined to servers or desktops and trained in specific tasks like image classification, usually with stationary datasets that fail to capture the causal and sequential relationships between objects in frame. Thus, embodied AI generally violates the clean dynamics of structured environments, and require agents to cope with noisy sensors, effectors, dynamics, and other agents, which creates unpredictable outcomes [29]. The field argues that intelligence is intrinsically connected to embodiment in bodily experience, and that separating them has distorting effects on research [13].

The first generation of Embodied AI researchers focused on robotic embodiments, employing physical agents in the real world and forcing interactions with their noisy environments through a set of sensors and effectors [13]. However, recently, such endeavors have been supported by simulation frameworks derived from real building scans and ensuring real life physics and collision effects. This allows training the agent in close to real life scenarios without the physical and financial constraints of the former way. With tasks consisting of navigation, rearrangement and vision-and-language challenges, the field has seen considerable progress and continues to improve on these tasks and introduce new challenging tasks. In summary, this field is more than just the study of agents that are exposed and made active to interact with its environment, but it is also an exploration of the properties of intelligence that surfaces from the confluence of multiple modalities through vision, language and audio.

2.2 Embodied AI: Tasks and State-of-the-Art

There have been numerous tasks and methods that have formed from the above principle, the most influential of which will be discussed in this section. The challenges can be widely divided into three: navigation, rearrangement, and vision-and-language tasks. Within these tasks, we present the evolution of major techniques employed, leading up to the state of the art. All tasks are evaluated in the common Habitat simulation framework, containing the same observations and action spaces, and differing in metrics and datasets used.

2.2.1 Navigation Tasks

In a nutshell, this task consists of an agent tasked with navigating a given environment and finding a specific target. The challenges within this task differ on numerous fronts, such as how the target is expressed, whether in terms of a specific location (PointNav) or a target object (ObjectNav), or the type of environment considered like a dynamic or static one.

Challenges: In PointNav, an agent is initialized randomly in an unfamiliar scene and is tasked to navigate efficiently towards a designated target location. The agent is aided by RGB-D and egomotion sensors, and the episode is deemed successful only if the agent calls $stop$ within $0.2m$ of the target location within 500 maximum steps. The Success Rate (SR) and Success Path Lenght (SPL) are considered to measure the efficiency of the agent's trajectory. Likewise, in adjacent challenges like ObjectNav or Social PointNav, the formulation is the same, with slight modifications made to accomodate the updated task. For example, in ObjectNav, the agent is expected to navigate towards a target object, and the episode is a success if the agent calls $stop$ within $1m$ of the object, and the object is visible in the camera frame.

Prominent Techniques: Early solutions to the navigation task involved a combination of classical and learning-based methods, where the agent projects its field of vision to a create obstacle and explored maps, while the policy to reach the target is trained as a CNN visual encoder like Resnet18 [16, 8]. With the challenge redefined to account for more realistic noise and collission dynamics, new techniques emerged, especially focusing on large-scale training and modular architectures. These feature semantic mapping modules [45], some leveraging the recent success of Vision-Language Models [5], pre-training on large-scale simulations like that of 10K procedurally generated houses in ProcTHOR-10k, and fine-tuned imitation learning architectures. These have led to achieving an SPL of 0.74 and SR of 0.96 in PointNav, effectively considering the challenge to be solved [20]. ObjectNav has witnessed an SPL of 0.288 and SR of 65% [12], setting the stage for further improvements to be made.

2.2.2 Rearrangement Tasks

Rearrangement tasks require an agent not only to navigate a given scene, but also to interact with and manipulate objects to achieve a desired configuration. These tasks introduce additional challenges involving physical interaction, object recognition, and planning, going beyond pure navigation. The agent is tasked with moving or detecting the changes occurring in the states of the scene. Thus, if an object like a banana is moved, the agent is expected to detect the change and put it back into place [13].

Challenges: The major challenges can be neatly divided into detecting scene changes and performing interactive rearrangement. Common to both, in the first phase, the agent is allowed to survey the scene, while the scene is changed slightly in the second phase. The agent is expected to flexibly encode and track dynamic environment states while making long-term plans for efficient rearrangement. Regarding metrics, [15] introduced the OMQ measure and [41] provided the FIXEDSTRICT measure, both of which effectively evaluates the change in the final 3D object cuboid providing an object-level quality score ranging from zero to one [13].

Prominent Techniques: Although initially there had been limited engagement with these challenges, with techniques failing to surpass established baselines, in 2022, several promising approaches were introduced leading to an increase from 9% to 24% in the FIXEDSTRICT metric. These were made possible due to the use of CLIP pre-trained visual encoders [21] and large-scale pre-training using procedurally generated environments like ProcTHOR [12].

2.2.3 Vision-and-Language Tasks

The defining characteristic of this task is the use of natural language to convey the goal or instructions to the agent, with the motivation to create embodied agents that can assist humans in day to day tasks through simple conversations. For example, the agent can be directed to go to a specific room, followed by navigating to a certain object on the table present in the room. This requires the seamless integration of language comprehension, visual perception, and embodied action to complete the required set of complex instructions.

Challenges: This task presents unique challenges that stem from grounding unconstrained language instructions in embodied scenarios. Common challenges include the ones mentioned below [13]. Here, success is measured using metrics such as Success Rate (SR), Success weighted by Path Length (SPL), Navigation Error (NE) and specialized ones like Dynamic Time Warping (nDTW) [18] for path alignment.

- **Navigation Instruction Following**: Defined by the RxR Habitat Challenge, agents are expected to interpret language instructions like "Walk down the hallway, turn left at the sofa, and stop at the sink", and follow the instructions in a previously unseen environment. This emphasizes complex, long-horizon planning often testing with partial observability and ambiguous language.
- **Interactive Instruction Following:** Utilizing the ALFRED benchmark, this challenges the agent to follow instructions that require interaction with the environment, such as "Pick the mug, rinse it and place it on the table top". This requires the agent to comprehend and ground the directive, and execute sequences that can involve navigation and manipulation.
- **Dialog-based Instruction Following:** Using the TEACh dataset, which is made up of human-human dialogues and demonstrations of household tasks, the agent is expected to engage in natural language interaction with people, both following directives and using conversation as a means of surpassing ambiguous objectives. TEACh involves a Commander agent that acts as the human director, and a Follower agent that executes tasks and clarifies any directives. This challenge captures the complexity of human-robot interaction, paving the way towards conversational grounding in embodied contexts.

Prominent Techniques: Early approaches often relied on modular pipelines combining language parsing, symbolic reasoning, and classical planning [2]. Recent state-of-the-art systems employ end-to-end learning architectures that integrate vision and language using attention mechanisms and transformers, often coupled with spatial-semantic mapping to build persistent environment representations for grounding instructions [4]. Large-scale datasets such as RxR-Habitat, ALFRED, and TEACh have driven progress by providing rich multimodal benchmarks with natural language, human demonstrations, and interaction. Agents have benefited from pretrained language-vision models, hierarchical planners that decomposed complex instructions into waypoints, and modular policies that separate perception, language grounding, and control.

3

Problem and Platform: ObjectGoal Nav in Habitat AI

The challenges in embodied AI consist of incorporating traditional intelligence concepts from vision, language, and reasoning into an artificial agent trying to make sense of a novel virtual environment [14]. In this project, we are mainly concerned with a subset of the diverse tasks associated with Embodied AI, namely **Object Goal Navigation**. In this task, the agent is placed randomly within the scene and is tasked with navigating to a target goal object. Once it is sufficiently close to the object, it is expected to call stop (Refer to Figure 3.1). We contribute to the task in two different ways, one to improve accessibility to already existing approaches and the other to enhance navigability to rare, long-tail objects.

3.1 Problem Statement

Specifically, our **problem statement** consists of the following two objectives:

- **ModNav:** Create an easily accessible and standardized interface for prominent mapping techniques employed in Object Goal Navigation tasks. With a few consistent class methods, this would enable quick retrieval, efficient experimentation and lead to a holistic Embodied AI framework.
- **HSSD-rare, IMPRINT:** Enhancing an embodied agent's navigability and localization abilities for rare, long-tailed object categories. An object category is labeled long-tailed if it corresponds to a specific brand, shape, or

contains a unique characteristic that sets it apart from its generic high-level category. For example, "Acura Rugs Contempo Hand-Tufted White-Black Contemporary Rug" would be considered long-tail, instead of the generic label "carpet".

In the following chapters, we detail the methods, their implementation and eventually the experiments done to validate our approaches, supported by the results obtained from them. These objectives aim to complement and progress upon the field of Embodied AI, bettering the agent's capabilities to eventually converge onto human-like capabilities. However, before proceeding to such ambitions, we spent some time in the following section on the platform on which all such approaches are built and tested out.

3.2 Habitat AI

The Habitat Environment provides a framework that solidifies the philosophy of the field of Embodied AI; to train and test agents in life-like scenarios with the confluence of vision, language and interactability, creating a capable embodied agent. Since practical deployment of agents in the real world is time inefficient, economically costly and possibly dangerous, Habitat provides a 3D simulation platform where one can train configurable agents and build policies for diverse tasks like navigation, object interactions, and grounded question answering. It allows benchmarking approaches using predefined and standardized metrics like Success Rate (SR) and Success Path Length (SPL). The existence of such a flexible platform helps with deployment of carefully trained and tested agents onto the real world, accounting for the changes occuring during Sim2Real transfer.

The Habitat Environment is made of two interfaces; Habitat Sim, which is the core 3D simulator, and Habitat-Lab which provides an accessible interface to manipule the simulations and benchmark experiments. Both these modules will be discussed, explaining their base structure and their role in training an embodied agent.

3.2.1 Habitat-Sim

This is the core 3D simulator that handles low-level simulation tasks like constructing the scene's navigation mesh, enabling agent movement and sensor data generation, and maintaining proper physics and collision detection. Built in

C++, it is made to efficiently render a scene and the objects within it. For example, when rendering a scene form the MP3D dataset [7], Habitat-Sim achieves several thousand frames per second (FPS) running single-threaded, and reached over 10,000 FPS multi-process on a single GPU [33].

The simulation is configured in two parts:

- **Simulator Backend:** This contains configuration parameters that are required to initialize and get the simulator running. Here, we define the scene we are interested in, enabling physics and the GPU device to host the simulation.
- **Agent Config:** This defines the agent and sensor configurations, enabling fine-grained access to modify the way we navigate a space. Here, we can define the sensor parameters like dimensions, position, horizontal field of vision (HFOV), and even extending control to define the agent's action space.

Both these configurations are used to build a Simulator class, after which the agent is fully equipped to explore the chosen scene. The agent is kept confined within the scene boundaries with the help of Navigation Meshes (NavMesh), which is a collection of two-dimensional convex polygons that defines navigable and non-navigable regions. For navigation, the agent can use the "step" method of the Simulator parent class, and the episode ends when the agent calls "stop".

3.2.2 Habitat-Lab

Wrapped on top of Habitat Sim, Habitat Lab is a high-level and modular python framework that provides accessible abstractions like task definitions, easily configurable experiments and managing a wide range of simulation datasets. This helps simplify the development and benchmarking pipelines by utilizing reusable modules and standard metrics, removing the need to manage the low-level simulation details.

Managing Parameters and Tasks It uses Hydra for efficient configuration management with YAML config files, providing the option to override parameter values directly from the command line. Among these parameters, Habitat-Lab provides modular support for diverse embodied tasks like ObjectGoal Navigation. These modules can be extended to define custom tasks and modify the action spaces or reward functions.

Managing Datasets Habitat-Lab expects the dataset and episodes to be constructed in a specific but straightforward manner, ensuring efficient loading and parallelizing of episodes. An episode consists of the agent's start position and orientation, the target goal's position, the desired scene to load and related task specific metadata. The exact pipeline for creating a dataset and episodes is detailed in chapter 5. Currently, Habitat-Lab supports standardized datasets like Gibson [42], Matterport3D [7], HM3D [30], Replica [36] and HSSD [22].

Most importantly, Habitat-Lab includes utitilies and scripts to automate RL training loops, paired with the evaluation and logging of metrics like Success Rate (SR) and Success Path Length (SPL). It ensures episodes are run in a distributed fashion while training, leveraging the full throttle of available GPU units.

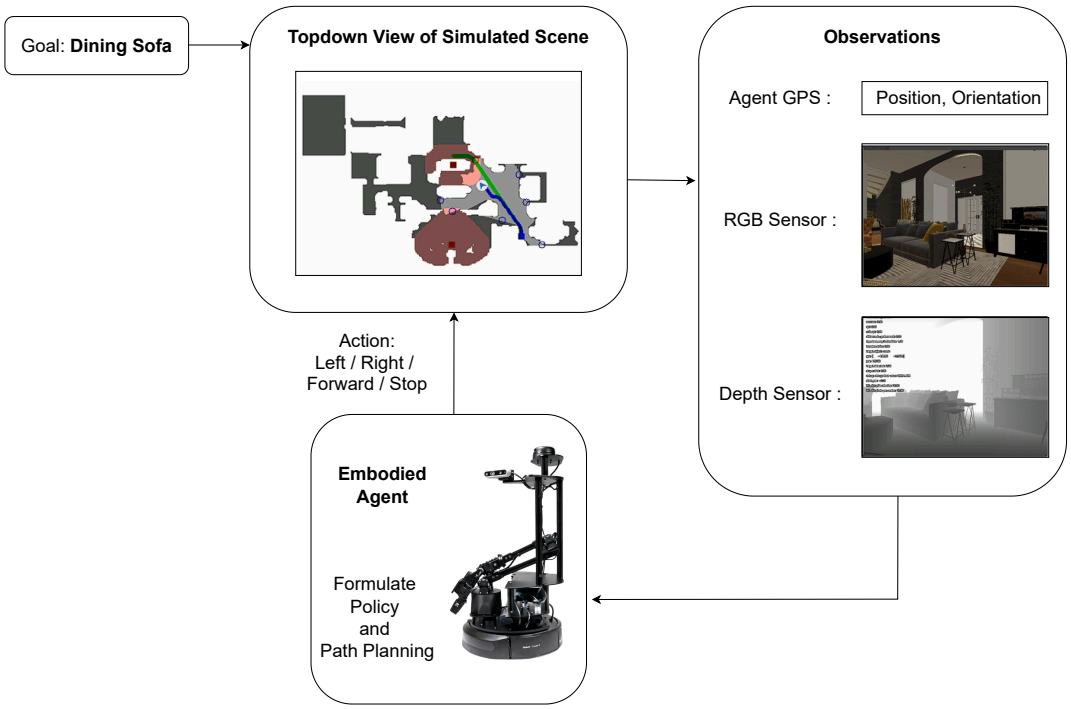

Figure 3.1: Pipeline for ObjectNav Task. The agent is tasked with navigating to a target goal object, relying on its RGB and Depth sensors, along with its GPS tracker. Its policy is supposed to take these inputs and predict the next local action, helping navigate and eventually reach the target goal (here, Dining Sofa)

4

ModNav: Modular Navigation

ModNav aims to unify different environmental mapping techniques into a single accessible and modular framework, spanning approaches such as 2D Semantic mapping [9] and Vision Language Frontier Mapping [45]. As each approach contains both the mapping and policy parts intertwined, we extract the mapping phase and wrap it with an accessible interface, which contains the core methods mentioned below. Apart from these core methods there can be additional available methods depending on the mapping technique. For example, in VLFM, the method $get\_best\_frontiers$ returns the best frontiers scored with respect to the value map.

- **__init__(self)**: Initializes a new map for the scene.
- **update_map(self, rgb, depth, pose)**: Updates the map with new observations.
- **get_map(self)**: Returns the current map.
- **reset(self)**: Resets the map to its initial state.
- **visualize_map(self)**: Visualizes the current map.

Thus, with a steady input of the current observations consisting of the RGB, Depth and Pose (position and quaternion) information, the map gets updated as the agent explores the scene. Such consistent modularity is of use especially when testing baselines and comparing approaches, paving the way to a holistic framework for Modular Embodied AI.

4.1 Approaches Implemented

4.1.1 2D Semantic Mapping

The main principle behind this approach [9] is to construct an obstacle map of the environment while semantically labeling the perceived objects. This was an early influential work that introduced a context-aware mapping method, and continues to be a good introduction to the contemporary techniques and concepts prevalent in Object Goal Navigation. Thus, assigning easy accessibility and modularity to this base approach was essential.

The pipeline consists of three main phases, starting after obtaining the RGB, depth and Pose observations as input.

- **Point Cloud:** Using the depth input, we project the pixels onto three-dimensional points, constructing a point cloud.
- **Semantic Predictions:** Passing the RGB input through a pretrained Mask-RCNN, we obtain the semantic predictions for possible objects within the frame. Each pixel belonging to the semantic object is assigned the semantic label.
- **Constructing Map:** Now that we have the point cloud, with each point assigned a semantic label (if any), we sum across the height axis producing a map containing obstacles, the explored area, and the semantic objects.

The final map is of shape (K, M, M), with M being the dimensions of the map, while K corresponds to the number of channels, consisting of obstacles, the explored area, and the semantic categories (Figure 4.1).

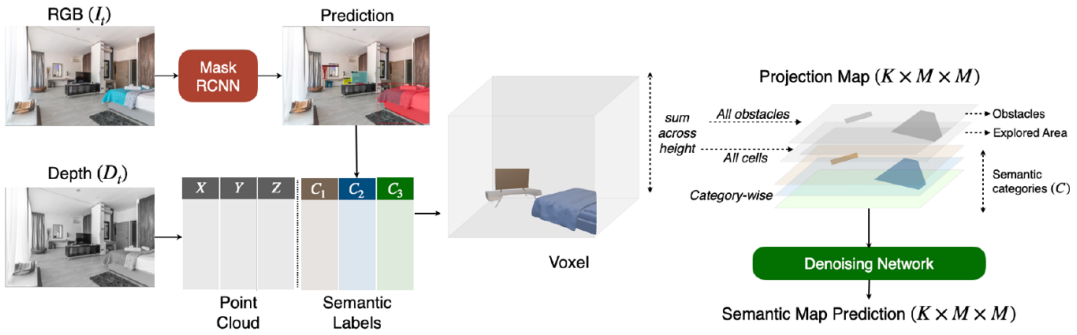

Figure 4.1: Pipeline for 2D Semantic Mapping. Figure taken from [9].

4.1.2 Vision-Language Frontier Mapping (VLFM)

The main principle behind this approach is to guide the agent using both an obstacle map and a value map, which is generated using a multi-modal Vision-Language Model (VLM), providing the next best frontier to explore. Most contemporary approaches are derived or are influenced by this approach, which set a high benchmark and challenged the status quo with the usage of multi-modal models. Including this influential mapping method was crucial in promoting easy comparison to a well-established and recent baseline.

In VLFM, we deal with two main mappings (Figure 4.2):

- **Obstacle Map:** Similar to 2D Semantic Mapping, we obtain the point cloud from the depth input, and flatten along height dimension. The explored area is defined as the inverse of the obstacle region, while the frontiers are calculated in a straightforward manner using the *cv2* package's *contour* method on unexplored regions.

- **Value Map:** The RGB input and the target goal query is passed through an image-text matching VLM, which returns a similarity score. This value is projected over all pixels in the Field of Vision (FOV) (except on the objects), with the help of the depth input. Using this value map, we can score the active frontiers, and choose the next best region that might contain our target object goal.

With the help of these two maps, the agent is guided towards regions that have a high semantic possibility of containing the target object, all as a result of the richly developed multi-modal space of the VLM. Policies depending on multi-modal spaces have gained traction since the advent of this approach, and have now become the status quo in Embodied AI.

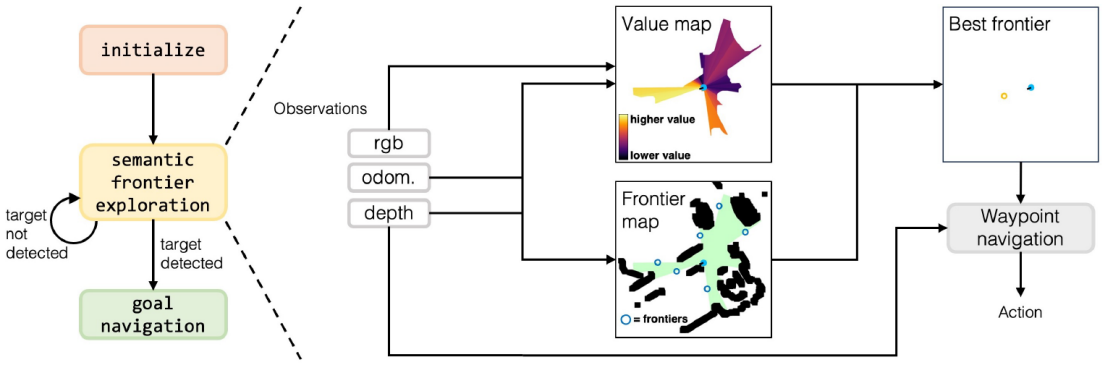

Figure 4.2: Pipeline for VLFM. Figure taken from [45].

4.2 Challenges

4.2.1 Preparing the Input Observations

Different mapping approaches impose distinct requirements on their input observations, especially regarding pose information. For example, 2D Semantic Mapping [9] updates the map using the relative difference between the agents current and previous local poses. On the other hand, VLFM [45] expects the absolute pose of the agent relative to its initial starting position.

RGB and depth images often need additional preprocessing, such as resizing, cropping, or normalization, to match each models input dimensions and value ranges. Because every method has its own preferred data format, one must carefully study its preprocessing pipeline (often only implicit in sparsely commented code) to ensure that the inputs are correctly prepared.

2D Semantic Mapping

In case of 2D Semantic Mapping, the relative pose difference is framed as follows. For a given global previous pose (x_1, y_1, o_1), and a current pose (x_2, y_2, o_2), we can express the latter in terms of the former as given in Equation 4.1.

$$\begin{aligned} dx &= dcos(360 - \theta - o_1) \\ dy &= dsin(360 - \theta - o_1) \\ do &= o_2 - o_1 \end{aligned} \tag{4.1}$$

Here (Figure 4.3), we consider the difference vector (d) and the angle at which it extends from the x-axis as $(360 - \theta)$. To work from the local frame, we need to take into account the orientation of the frame (o_1), which gives us the effective angle to be: $(360 - \theta - o_1)$. This effectively expresses the orientation of the difference vector with respect to the local frame. To confirm the validity of this derivation, refer to the Appendix section 9.1.

Vision-Language Frontier Mapping

VLFM expects the input pose to be the current local pose of the agent with respect to its initial starting pose. That is, in VLFM, the agent is initialized to

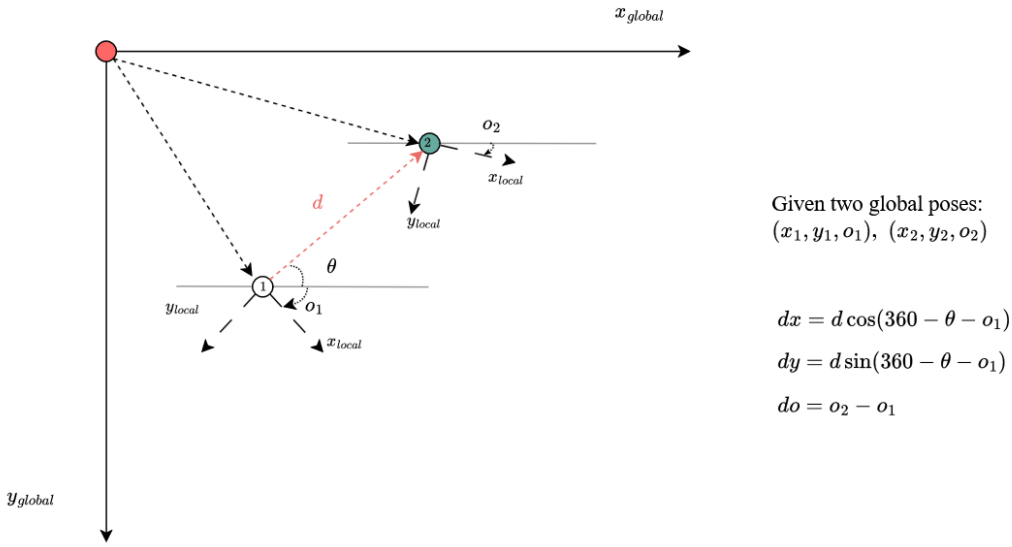

Figure 4.3: Obtaining relative local pose change from current and previous global poses. This frames the current pose in terms of the previous one, as required by 2D semantic mapping technique.

start at the position $(0, 0, 0)$ corresponding to $(pos_x, pos_y, orientation)$. Thus, all subsequent poses are calculated with respect to this initial pose. This is done using a four-dimensional transformation matrix that accounts for the change in position and orientation.

Given the current local agent pose, with the values x, y, yaw, the transformation from the initial pose to the current pose is expressed as the following transformation matrix.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{pmatrix} \cos(o_1) & -\sin(o_1) & 0 & dx \\ \sin(o_1) & \cos(o_1) & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{4.2}$$

Here, the agent is translated and oriented from the previous to the current local pose, where (dx, dy, dz) corresponds to the difference in local poses. In this way, the agent's position and orientation is tracked using this transformation matrix throughout.

Since the provided pose observation might be global, we need to localize it to the agent's initial pose, requiring a simple difference to be taken as the

preprocessing step.

4.2.2 Extracting the Mapping

This phase, in a nutshell, comprises of carefully studying the code, extracting the relevant parts that comprise the mapping technique, and wrapping it in an easy-to-access interface. All the challenges faced can be marked down to two major categories:

Tracking Coordinate and Convention Changes: Going through the paper and its implementation demands strict tracking of the various changes in coordinates and conventions that occur. A mapping technique generally deals with three different coordinate transformations: Simulation's (Habitat) Global coordinates, Agent's local coordinates, Map's pixel coordinates. Alongside this, we have the convention where arrays and plots are referenced with the x-axis flipped, producing a coordinate system where the x-axis increases towards the right, and the y-axis increases downwards. Loss of track of these changes often caused stark errors during mapping.

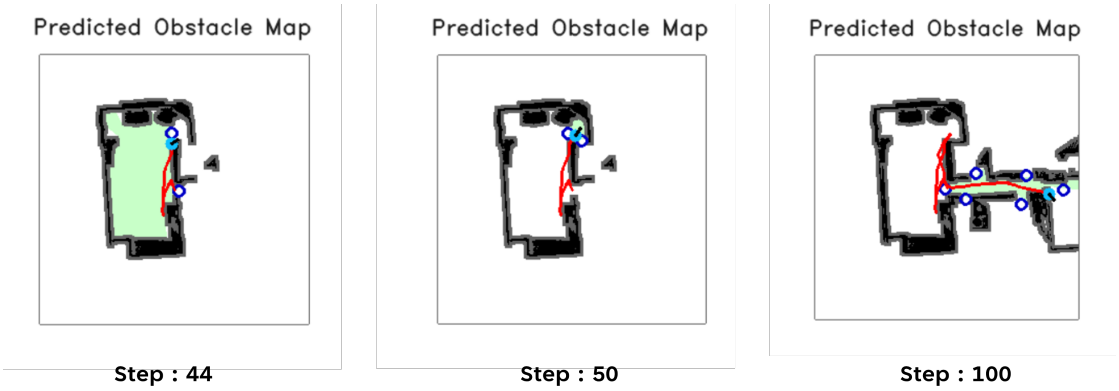

Figure 4.4: Loss of information regarding previous explored area (green-shaded regions) due to the agent going past the navigable buffer zone (grey-shaded regions).

Issues due to Generalized Agent: ModNav aims to build a generalizable agent, capable of varying in size and shape. However, such changes can adversely impact the mapping, requiring slight modifications to preserve the approach as intended. In VLFM, this occurs in the form of navigable areas, which

are buffer zones around obstacles meant to prevent the agent from getting too close to non-navigable regions. When the agent often enters this buffer zone, the explored regions are reset and information about previously tracked explored regions is lost in this bug (Figure 4.4). To fix this, we removed the explored region's dependence on the buffer zone, allowing the agent to get as close to the objects as possible.

Once such challenges are dealt with, we would be able to make the mapping approach accessible using the core methods mentioned before.

4.2.3 Dynamic Batching

To ensure parallelization in the case of multiple data batches, we implement modular dynamic batching where environments are run in parallel accounting for active and finished ones. This is done using PyTorch's *DataLoader* and *BatchSampler* classes. When dealing with datasets that are saved locally, we load the data into batches using *DataLoader* and keep track of the active batches with flags using *BatchSampler*.

```
from torch.utils.data import BatchSampler
import numpy as np

class DynamicBatchSampler(BatchSampler):
    def __init__(self, n_scenes, batch_size = 1):

        # List of lists representing batches
        self.batch_indices = np.expand_dims( np.arange(n_scenes),
axis = 1 )
        self.batch_size = batch_size

    def __iter__(self):
        for batch in self.batch_indices:
            if len(batch) == self.batch_size:
                yield batch

    def __len__(self):
        return len(self.batch_indices)

    def remove_batch(self, finished_batch):
        """Remove the finished batch from the list of batch
indices."""

```

```
            #True for finished batch
            del_mask = self.batch_indices != finished_batch

            #Applies finished mask to indices list
            new_batch_inds = self.batch_indices[del_mask]

            #Expands to previous list shape
            self.batch_indices = np.expand_dims(new_batch_inds, axis
    = 1)

```

Code 4.1: DynamicBatchSampler Class for keeping track of active and inactive batches

In summary, ModNav provides a unified and flexible framework that decouples mapping from policy learning, enabling comparison and integration of diverse navigation methods. This accessibility is extended to both classical approaches, like 2D semantic mapping, and also more relevant cutting-edge approaches, like VLFM. Ultimately, ModNav lays the groundwork for a comprehensive platform in Embodied AI, promoting collaboration and standardization across the community.

5

HSSD-rare: A Long-Tailed Object Dataset

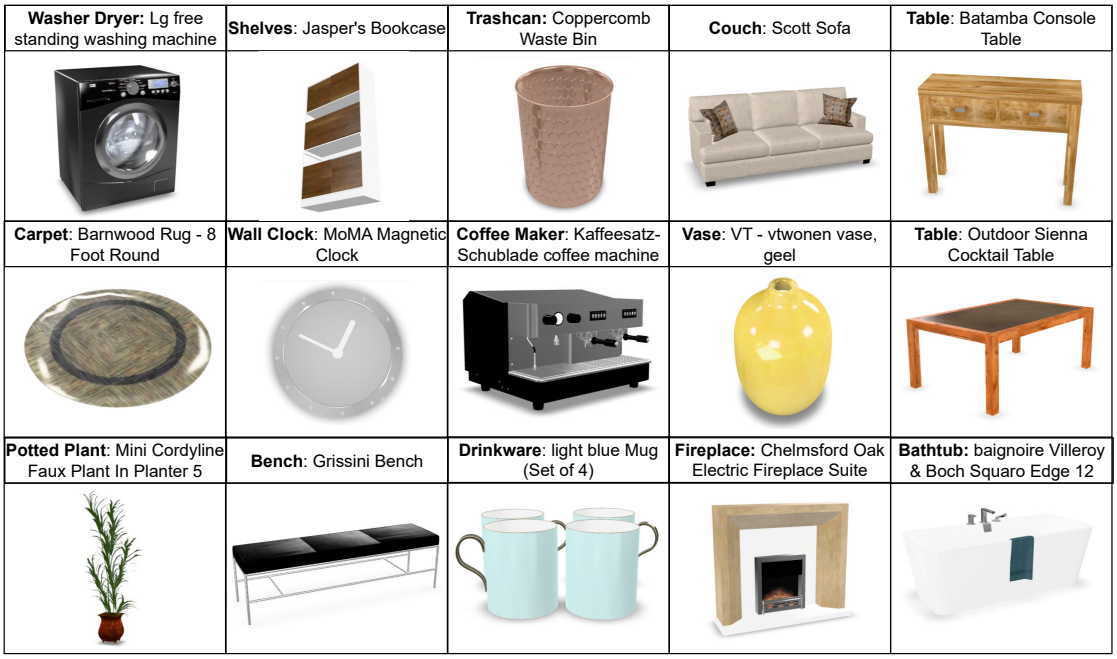

Figure 5.1: Qualitative Samples of rare, long-tail objects provided in HSSD dataset

In Embodied AI, although there are established approaches and datasets for various general tasks, there are few for navigating to rare and long-tail objects, a task that contemporary approaches find challenging. To progress in this, we require a dataset of objects on the long-tail along with their positions within

a given environment. OVON [44] provides test splits in which less frequent synonyms of the target objects are used as queries for an agent. Although the target objects are described using less common class labels (e.g., office chair instead of chair), they remain relatively general and simple to recognize.

In HSSD [22], a dataset of synthetic scenes, there are multitudes of rare objects provided in the source files (Figure 5.1). However, it lacks a large curated task-specific dataset (with episodes) for object goal navigation, where the currently available option only contains a subset of six objects. This necessitates the creation of HSSD-rare, a task-specific dataset which expands the scope to include all available HSSD objects, focusing specifically on long-tailed categories (Figure 5.1). This results in 1362 episodes spanning 17 different scenes ideal for zero-shot evaluation of queryable maps, and localization of rare objects.

There are two main phases to creating a task-specific dataset in the Habitat Environment, the first being generating valid viewpoints around a given object position, and the second being gathering relevant segments like the generated viewpoints, the episodes and the category to task_id mappings, to finally create the dataset. Both of these phases are explained in detail in the following sections.

5.1 Generating Viewpoints

Conventionally, datasets in the Habitat environment, like HM3D [30, 44] and HSSD [22], rely on semantic bounding boxes of the objects to generate viewpoints around them using IOU as a measure. However, such semantic information is lacking for most of the HSSD objects, prompting the need to either use a semantic segmentation model or develop an alternate novel method to determine the validity of a viewpoint. Since using the former would be time-intensive and impractical for generating a dataset with more than 1000 objects, we develop a novel algorithm for validating viewpoints.

We used the starting point provided by the HSSD data set [22] and used its curated list of rare objects. For a given object, we are able to access its position, dimensions, along with its main category and a related description. Now, for generating viewpoints, the core idea is to work with the depth observation to ensure the object's line of sight is clear and visible. If the object is obstructed, we claim that the viewpoint is invalid. This is implemented in two main phases, first finding the object's boundary, and second use this boundary to generate radial points and check if any of them qualify as valid viewpoints.

Algorithm 1 generate_view_pts

Require: $obj\_pos, obj\_dims, dist_{bound}, radius_{bound}, dist_{vp}, radius_{vp}$

$boundary\_points \leftarrow \emptyset$
for $\theta \in [0, 2\pi)$ **do**
  $pt_{bound} \leftarrow BinarySearch(dist_{bound}, radius_{bound})$
  if pt_{bound} exists **then**
    $boundary\_points \leftarrow boundary\_points \cup \{pt_{bound}\}$
  end if
end for
$view\_pts \leftarrow \emptyset$
for $pt_{bound} \in boundary\_points$ **do**
  $radial\_pts \leftarrow$ GenerateRadialPoints($dist_{vp}, radius_{vp}$)
  for $pt_{radial} \in radial\_pts$ **do**
    Continue if pt_{radial} is in previous pt_{bound} area
    Continue if pt_{radial} is not navigable
    Orient agent toward obj_pos from pt_{radial}
    $px_{obj\_dims} \leftarrow$ ProjectToPixels(obj_dims)
    $depth\_values \leftarrow$ GetDepthAt(px_{obj_dims})
    if $depth\_values >$ distance to obj_dims **then**
      $view\_pts \leftarrow view\_pts \cup \{pt_{radial}\}$
    end if
  end for
end for
return $view\_pts$

5.1.1 Finding Boundary Points

A given object position can exist in two possible zones: navigable or non-navigable. The way in which we ascertain the boundary for these two cases differs in their approach.

- **Navigable Zone:** In this case, we access the available object dimensions and scatter equidistant points along its outer boundary. These points mark the object boundary points. An example is a carpet, which is usually present in a navigable region (the floor).

- **Non-Navigable Zone:** This case is a bit tricky, as an object like a coffee cup can be placed on top of a table, which usually extends beyond the dimensions of the cup, preventing the agent to get close to the actual object. This prevents us from using the object dimensions as the sole reference, forcing us to account for other objects spanning over our target object.

  From the provided object position, we extend a $2m$ search radius and scan 360° to find transitionary points that mark the change from a non-

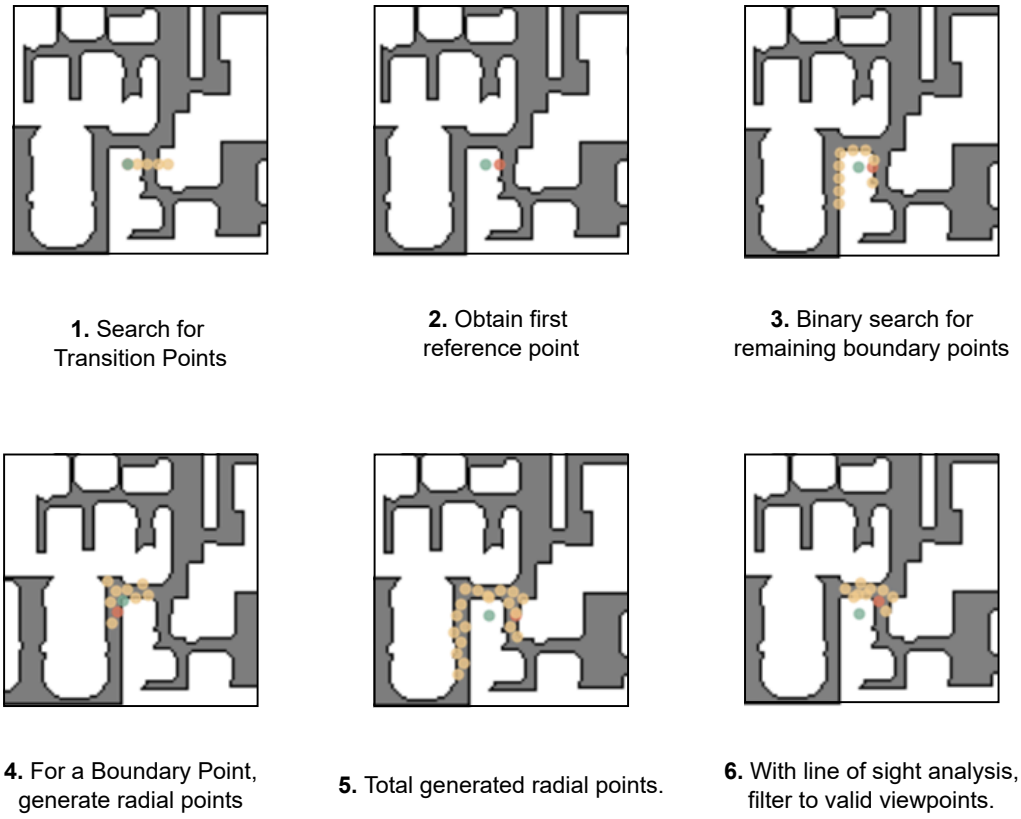

Figure 5.2: Pipeline for generating valid viewpoints around a given object position. The first three tasks (top row) aim to generate boundary points around the object. In the bottom row tasks, we iterate through each point, generating a cluster of radial points around it, and each radial point is validated to check if it is a viewpoint or not. This is done through line-of-sight analysis as explained in section 5.1.2

navigable to navigable region. This transitionary region provides a maximum threshold that the agent can explore till. Similar to the previous case, we consider equidistant points lying along the transitionary region, which is found through an efficient binary search, keeping a previously obtained point as reference. In this way, we obtain an effective outer-boundary for our target object.

5.1.2 Finding Valid Viewpoints

For each boundary point, we generate a cluster of radial points, extending a maximum of $1m$ away and spaced equidistantly from each other. Iterating through each of these radial points, we first confirm navigability. Then, the

Figure 5.3: Projecting positions of object's outer-dimensions onto the sensor pixel locations (marked red). These are used as reference to validate whether the object is in sight or not.

agent is placed at a specific radial point, and oriented towards the target object such that it is in the line-of-sight facing the object. From this point onwards, there are two major steps to find a viewpoint:

- **Pixel Locations of the Object:** Our aim here is to find the region in our sensor frame where the object is located. We know the object position and its dimensions, which provides us the absolute positions of its dimension points. From here on, we will identify the object with its spread out dimension points. With a matrix transformation, we can convert the dimension points from absolute coordinates to relative to the current position. The major conceptual step in this section is to utilize these relative dimension points, along with the sensor's focus, to obtain the exact sensor pixel locations of the object in the sensor's frame. This is done by the standard projection process from a three-dimensional point to a pixelated sensor frame [37]. This essentially provides us pixel coordinates encompassing the whole of the object.

$$\begin{aligned} \Delta p_y = \text{round}\left(\frac{f \cdot y}{|z|}\right),\ \Delta p_x = \text{round}\left(\frac{f \cdot x}{|z|}\right) \\ p_y = \frac{H}{2} - \Delta p_y,\ p_x = \frac{W}{2} - \Delta p_x, \end{aligned} \tag{5.1}$$

  Here, f represents the sensor's focus, while (x, y, z) corresponds to the relative position of the target object with respect to the current radial point. (H, W) refers to the sensor's dimensions. This provides a projection from relative three-dimension positions to the respective sensor's pixel coordinates.

- **Line-of-sight Analysis:** Checking the depth values at these pixel locations, we can test for meaningful visual access to the target object. If a depth value is lower than the metric distance (or depth) to the target point, we can claim that there is an obstacle obstructing the line-of-sight to the desired point, and vice-versa. In this way, we can check if any point on the object's

outer-dimension is visible without obstruction. If so, the radial point is deemed qualified as a valid viewpoint.

5.2 Generating Dataset

In this phase, we essentially iterate through each object, collecting all relevant information and formatting them as required, and finally saving them in files expected by the Habitat Environment. Since HSSD [22] follows the HM3D [30] formatting, we do the same since HSSD-rare is essentially an extension of the HSSD dataset. There are two main phases when generating a dataset in Habitat.

- **Object Goal Information:** In here, we save all relevant information regarding a given object, which constitute the following:
  - **Object Position:** Absolute position of the target object in habitat coordinates.
  - **Object ID**: Corresponds to a 12-digit unique identifier for an object category.
  - **Object Name**: A string made of the object ID and a numeric marker, intended to keep track of multiple occurrences of the same object category.
  - **Object Category**: Concatenation of the object main category and its description, which conveys additional context for the object going beyond a simple main category classification. This is particularly effective when dealing with long-tailed objects.
  - **Object Viewpoints**: List of valid viewpoints surrounding the object position. Each viewpoint consists of its position and the orientation to face the object.

- **Generating Episodes:** Here, we generate episodes for each object categories. Since we have a large number of objects available (1362), we stick to generating one episode per object. In here, we gather the following relevant information to construct an episode for an object:
  - **Episode ID**: Taken as one more than the number of episodes produced so far.
  - **Scene ID**: Obtained from the Habitat Scene ID.
  - **Scene Dataset Configuration**: Absolute path to the scene dataset config file, required for loading the scene and objects.
  - **Object Category**: Category name of the current object, which is a concatenation of the main category and its description.

  - **Start Position**: Episode start position, where the agent is initialized in the scene. A random navigable position is sampled within the range of $3m$ to $10m$ from the object position.
  - **Start Rotation**: Episode start rotation, where the agent is initialized with this rotation in the scene. For this, we generate a random orientation.
  - **Info**: This contains the geodesic and euclidean distance from the initial agent position to the target goal's nearest viewpoint.

- **Category to ID Mapping:** We also generate a mapping from the object category to a corresponding unique ID. This is a straightforward process of assigning the index value of the object category (found in the $objects.csv$ file in the HSSD directory) as the category ID.

With all these gathered information, we can generate a dataset for a scene by saving the above information (*object goal information, episodes, category to ID mapping*) as dictionaries, saved in a $json$ file format. In this way, we produce HSSD-rare, a task-specific dataset curated for long-tailed object navigation and localization.

Valid Viewpoints around Target Object (in green)

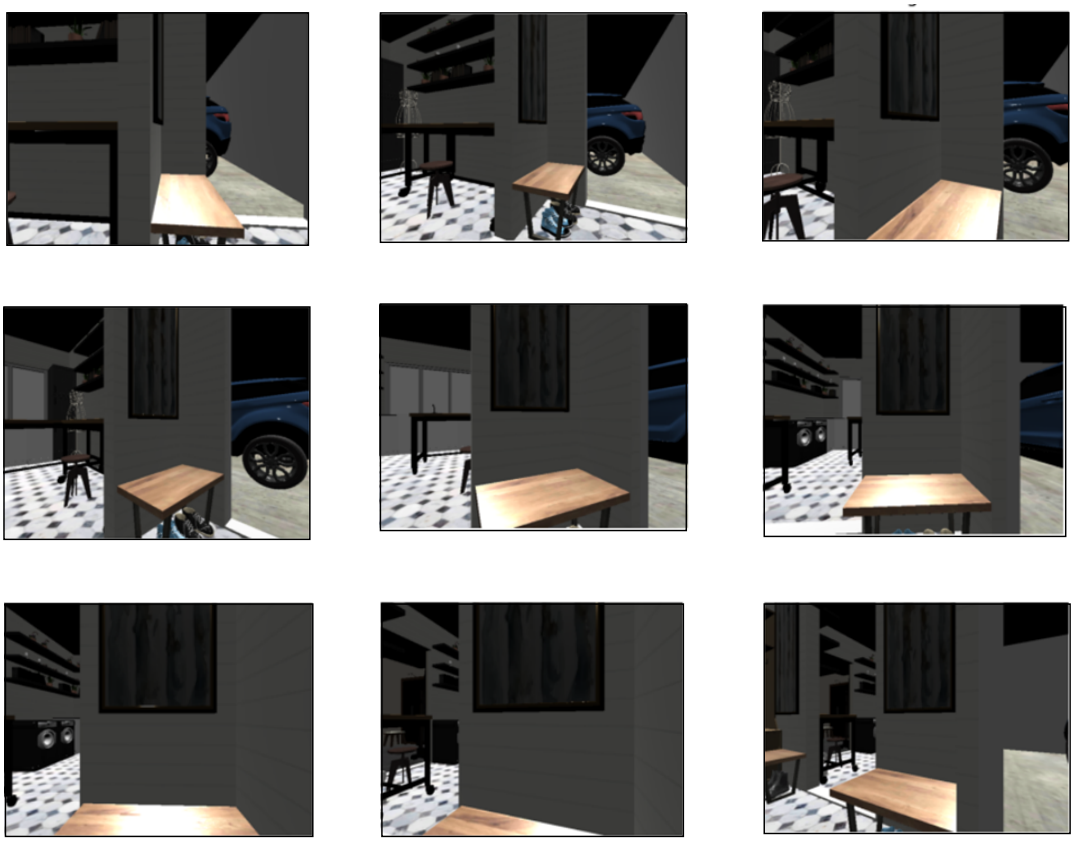

Figure 5.4: Viewpoint Samples. For a target object (Wall Painting), the top image contains all the valid viewpoints generated with a distance between viewpoints as 0.5 meters. Given below are 9 viewpoint samples from the left to right, and we note that viewpoint generation is successful as the target object is visible in each of them.

6

IMPRINT: Zero-Shot Navigation using Image Retrieval

Contemporary methods have leveraged Vision-Language Models (VLMs), such as BLIP2 [23] and SED [43], to understand open-vocabulary queries and navigate agents through complex real-world environments. This goes past the prior restriction faced by numerous datasets, like COCO [10], which confined target goals to a predefined set of 6-20 common categories ("chair", "TV", "bed", etc.). OVON [44] partially addresses this limitation by expanding to 379 categories, scattered through three sets varying by their specificity: *seen*, *seen synonyms*, *unseen*. However, this dataset too lacked truly rare or long-tailed objects, going only so far as providing a more specific description of a common object ("clothes hanger rod"). Additionally, relying solely on commonly used open-set object detectors, such as OWL-ViT2 [25] and GroundingDino [24], would frequently produce false positive predictions, leading to failures in locating the required target objects.

Due to the above reasons, we develop IMPRINT, a novel zero-shot approach that constructs a spatial queryable map by utilizing both VLMs and online image retrieval. We utilize a sparse grid-based map with a grid size of $1m \times 1m$, in contrast to the dense representation used by methods like VLFM [45]. Evaluation is conducted on the previously discussed (Chapter 5) HSSD-rare dataset, which contains 1362 episodes spanning over 17 scenes of the HSSD datasets, validating navigating capabilities to true long-tailed object categories.

The core idea behind IMPRINT is to build a queryable map, a grid-based map

that is populated with spatially-grounded feature vectors that represent the grid space it spans, which can be referenced to localize an open-vocabulary query. The map is built by passing the RGB observation at each step through a VLM feature extractor, after which the corresponding feature vector is projected onto all grids within the current field of vision. When localizing a query, instead of simply using the feature vector of the text query, we additionally retrieve relevant online images that provide visual context to the query, and use their corresponding feature vectors as reference against the backdrop of the scene's grid-embedding map.

This promotes zero-shot navigation capabilities on an open-vocabulary navigation task. In order to navigate to a specific grid or point, we assume a simple PointGoal Navigation policy, which achieves a 99% success rate [30] in reaching target waypoints. In the subsequent sections, we describe in detail the core components required to construct the IMPRINT pipeline: Building the Map, Retrieving Online Images, and the Pipelines.

6.0.1 Mapping the Scene

IMPRINT hinges on constructing a feature map that is representative of the scene's locations and the objects contained within it. Our approach, inspired by that of VLFM [45], prompts the agent to explore the environment based on scored frontiers, all the while mapping the scene at each step. We will focus on the latter, discussing the scoring and frontier planning later on.

The goal is to build and continuously update a feature map of the environment using embeddings extracted from the RGB observation at each step. A

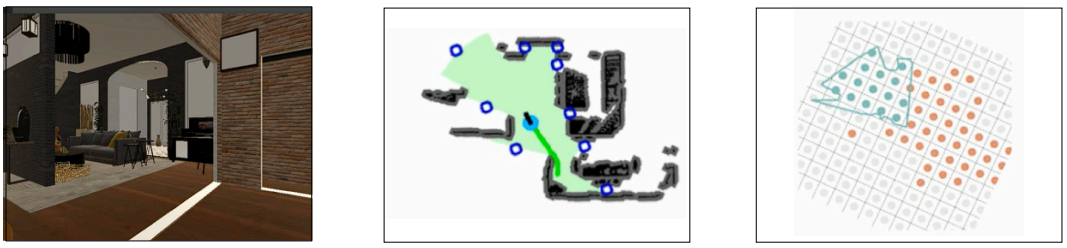

Figure 6.1: IMPRINT : Mapping the Scene. At each step, the agent extracts the feature vector corresponding to the RGB observation, and projects it onto all grid cells within the current field of vision (colored green in the feature map plot).

straightforward implementation would involve associating pixels in this map with an embedding vector that represents the scene's semantics at that location. However feature embeddings, like that of BLIP2 [23], are usualy high-dimensional (768-dimensional for BLIP2) vectors which would result in the final feature map occupying a significant amount of memory space ($\sim$ 2 GB), especially if we set a high pixel-per-meter value (e.g. 20).

To address this, we introduce a more efficient storage method by sectioning the map into predefined grid structures, each grid cell corresponding to a 1×1 meter square, and efficiently assigning embedding vectors to these cells. Specifically, given a map size of 1000×1000, a pixel-per-meter parameter of 20, and grid size of 20×20 pixels, we get a map with effective size $(50 \times 50 \times 768)$. This reduces the overall map size by roughly 400× with respect to the standard VLFM mapping (where the grid size would span one pixel). This produces a reduction of 99.75% in memory usage, enabling a significantly higher storage efficiency and subsequently quick retrieval when required. Such gains would be especially impactful when employed in real world experiments.

Therefore, we decided to implement this efficient storage method, and we additionally explored whether keeping the feature map size as small as possible would influence the mapping, or whether it might negatively affect web-retrieved image calculation, leading to degradation in results. For our standard implementation, we divided the map into square regions of 1×1 meter, but we also experimented with smaller square sizes, such as $20 \times 20cm$ and $50 \times 50cm$, which resulted in a greater number of regions, and a larger map size. Additionally, using smaller cells could fail to capture the complete semantic information of large objects that span multiple grids. We also avoided increasing the grid

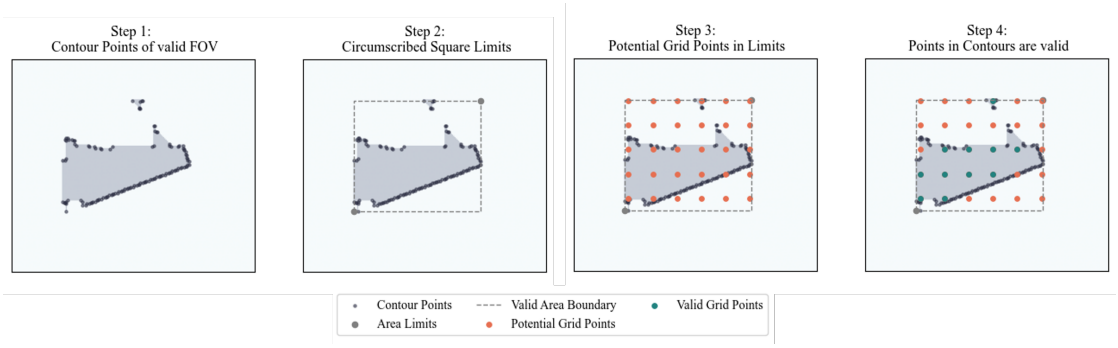

Figure 6.2: Projecting Embedding vectors onto Grid Cells. BLIP2 used this pipeline to cast vectors onto the relevant grid cells.

cell size due to:

- $1m$ being the predominant Habitat standard for recognizable maximum distance from a target object
- Spatial localization would be further lost as the grid cell size is increased.
- Increasing the grid cell size would mean burdening the sole feature vector, responsible for representing the spanned space, to represent further more context and scene relations within the confines of the same dimensional space as before.

Thus, to align with the objectives of this approach, we set the default implementation to use 1×1 meter grid cells.

BLIP2 Mapping: Constructing the map with the obtained feature vectors is surely dependent on the feature extractor used. In BLIP2, the feature vector obtained provides a holistic representation of the current RGB observation, encoding the scene semantics and relations in a single 768-dimensional vector. In IMPRINT, we project this feature vector onto all grid cells currently in the field of vision (Figure 6.1). Specifically, the current field of vision is obtained as a $cv2\ contour$, and passed through a closing morphology. This ensures the contour is closed and contains no undesirable holes inside. As shown in Figure 6.2.

Figure 6.3: Projecting SED patch embedding vectors onto spatially relevant grid cells. The patch corners are projected onto the pixel map, which are used to obtain overlapping grid cells. These cells are then assigned the corresponding feature vector.

SED Mapping: In case of SED, things get a lot more complicated. SED takes in an RGB observation and partitions it into 24×24 patches, assigning each patch a relevant embedding vector, taking into account its spatial relations to the other patches and its individual content. Thus, we obtain 24×24 feature vectors for an observation, each vector corresponding to a specific spatial region within the frame. This necessitates a projection more sophisticated than used for BLIP2. In IMPRINT, we do this by tracking the projection of the patch corners, from the sensor frame onto the pixel map. This provides a sense of where exactly the rectangular patch is located in our map. The grids overlapping over the projected map are assigned to contain the corresponding patch embedding. In this way, each patch embedding is projected onto grid cells that correspond spatially to the patched region. An illustration of this procedure is shown in Figure 6.3.

6.0.2 Online Image Retrieval

The core idea in this phase is to visually contextualize the text query, and subsequently provide more confident navigation and localization capabilities to the agent. For this, we leverage the large visual data present in Google Images. Hence, given a target object category, IMPRINT performs an online search and retrieves the top-n relevant images. This process scales linearly with the number of images requested (Table 6.1), and we set a maximum threshold of 15 images to ensure quick retrieval while gathering sufficient context regarding the object. Each image is sent through the VLM model and we obtain its corresponding feature embedding vector.

N° Images-per-request	**Time**(s)*	σ_t
1	2.1	±0.7
5	5.9	±1.4
15	15.1	±3.2

Table 6.1: Retrieval Time for Online Images. With linear scalability, this process has low time complexity and enables quick retrieval of required images.

To effectively localize the target query or object in the scene, we compute the similarity between each grid embedding in the feature map and the retrieved image embeddings. This produces a similarity grid for each of the retrieved images, signifying regions of interest that correspond highly to a specific image. We then take the mean of these similarity grids along the image dimensions,

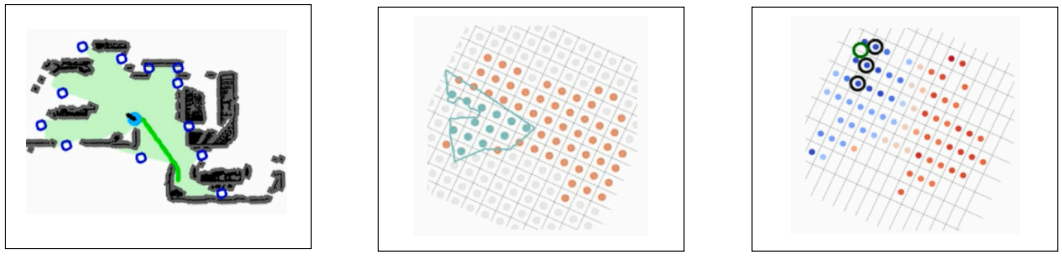

Figure 6.4: IMPRINT : Generating Similarity Grid. Using the query feature vectors as reference, these are compared against each grid embedding, generating a similarity grid for each of these queries. These similarity grids are then reduced by taking the arithmetic mean along generating a final similarity grid. Visible in the plot are the top-3 grid cells, along with the best frontier ranked by the similarity score.

resulting in a final similarity grid meant to represent an equally weighted map containing regions that are most relevant to all extracted images. This provides a single-value similarity map, similar to the VLFM approach. The top-k grids in this similarity map correspond to the locations where the target object is likely to be found.

This similarity grid can also be used for planning exploration before navigating to a target object, especially if the confidence level is lower than required. The agent can score the frontiers based on the similarity scores populating the grids containing them. This essentially ranks the frontier regions based on how likely the target is located in the vicinity. Once a frontier is selected, the agent can navigate to it using a straightforward, and established PointGoal Navigation policy, as mentioned earlier.

6.0.3 Vision-Language Feature Extraction

The core process in both the mapping and online retrieval phases is the embedding method. To implement this, we utilize two distinct feature extractors: BLIP2 [23] and SED [43]. BLIP2 has been an established vision-language model in the Embodied AI for a while, mainly due to its quick retrieval times and flexibility. It provides a holistic representation for a given RGB input, capturing the context in a single feature vector. On the other hand, SED operates by partitioning the RGB image into 24×24 number of patches, assigning a separate embedding for each patch. A patch embedding is meant to represent the physical

space spanned by its respective patch, providing a fine-grained representation to the frame in sight. For comparative analysis, we consider both BLIP2 and SED, and explore the effects on mapping and subsequent results.

While SED's approach offers potential benefits for mapping accuracy, it introduces challenges when applied to retrieved online images. As per its method, the online retrieved image would need to be divided into patches for processing by the SED feature extractor. The solution we implemented uses a select number (top-20) of highly relevant patches with the text query (chosen through similarity product scoring), and we take the mean of their corresponding patch embeddings to get a representative feature vector for the retrieved image. However, this process risks introducing inconsistencies due to the diverse semantic content and relations spread across the different patches, as noted in Section 7.

Our approach, instead of solely using retrieved images, also leverages a combination of text and image queries to further solidify and improve performance. In some cases, retrieved images alone provide insufficient or partial gains over text-only methods, particularly for categories that do not really belong to the long-tail definition. We address this limitation through multi-modal fusion. Specifically, we treat text related features as complementary to the image based features, and we experiment with different modes of fusion. Considering the similarity grids produced from each text or image queries, we consider reducing them through either an arithmetic mean, harmonic mean, or a fusion of both the arithmetic and harmonic mean. This integration enhances robustness and reliability by providing both textual and visual context to the query.

6.0.4 Pipelines: Static and Dynamic

To ensure proper and flexible evaluation of our approach, we combine all the core components and establish the IMPRINT pipeline in two major ways: Static and Dynamic. For smaller scenes that can be explored and mapped quickly, the static pipeline would be optimal, while for larger scenes where the agent might spend a significant time exploring the scene, the dynamic pipeline would be preferable.

Static Pipeline

This is done in two phases, with the first phase concerned with mapping the whole scene and the second phase deals with predicting the target object's

location.

- **Offline Phase:** In the offline phase, the agent is made to explore and map the entire scene. At each step, the RGB observation is passed through a VLM feature extractor, returning the corresponding feature vector, which is meant to be a high-dimensional representation of the spatial frame in sight. This feature vector is then projected out on all grid cells in the current field of vision (FOV). If a grid cell is already occupied, we simply consider the mean of the new and old vectors, forming the cell update rule. In this manner, the whole scene is populated with grid embedding vectors, providing a way to spatially localize any general open-vocabulary query.
- **Online Phase:** In the online phase (second phase), for a given text query, we retrieve relevant online images (upto 15 in number) and obtain the corresponding feature vectors by passing them through the VLM feature extractor. For each of the image feature vectors, we create a similarity grid by transforming the grid-map using similarity scores between the reference image feature vector and the grid embedding vector. This provides 15 similarity grids for 15 retrieved images, after which these are summed along this axis, providing a mean similarity grid. The highest scored similarity grid contains the most probable grid containing the target object, forming our prediction. This is compared against the true target position to provide us a quantifiable evaluation of our approach.

Dynamic Pipeline

Unlike static evaluation, where the agent maps the whole scene first and predictions are considered later on, here the agent explores, maps and navigates to a possible grid in real-time. Both the offline and online phases from before are combined to provide a dynamic navigation pipeline to make object goal navigation more efficient for the agent, which is useful when the scene is too large to be mapped first and queried later on.

As the agent moves around the environment, at each step, the RGB observation is used to extract the corresponding feature vector using the VLM, which is then projected onto the grid cells currently in the field of vision. To explore which region to explore next, the text query, along with the retrieved image queries, are used to create a similarity grid. This process is similar to the static case, where the feature vectors corresponding to each queries are used to create respective similarity grids, which are then reduced by taking the mean along the query dimension. The final similarity grid provides scores for each grid cell, including the ones containing possible frontiers. These are then used to rank frontiers, effectively sorting them by how probable each are to contain the target

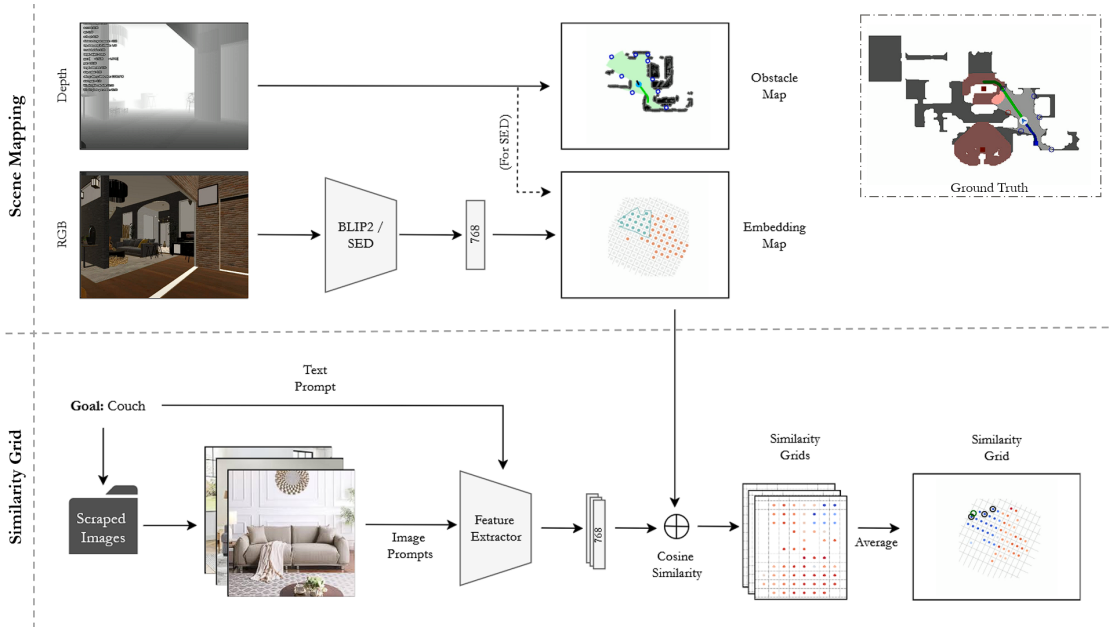

Figure 6.5: General Pipeline for IMPRINT. In Static mode, the scene mapping is done initially, mapping the entire scene and saving the embedding map, which is later used for creating the similarity grid and obtaining the prediction. In Dynamic mode, both the above steps are combined and the agent maps the scene and produces the prediction for the next frontier or target object in real-time fashion.

object in its vicinity. In this way, the agent chooses the next region to explore, all the while updating its map at each step.

When the similarity grid contains a grid scored above a certain confidence threshold (40% in our experiments), we infer the agent is confident about the presence of the target goal in the highly scored grid cell. The agent switches from exploring to navigating towards the grid cell, using the same PointGoal Navigation policy used in the other scenarios. If any other grid cell in the vicinity exceeds the target grid cell in similarity score, the agent is directed to reach the new target cell. In this way, the agent is forced to navigate to the best scored grid cell in a real-time, dynamic manner. Notably, this is similar to how the agent explores frontiers, where in this case, the highest grid cell effectively acts as a frontier to be explored.

7

Experiments and Results

IMPRINT offers a modular and flexible platform that can be seamlessly integrated into a number of embodied navigation pipelines. We focus on the task of Object Goal Navigation, where the agent is expected to navigate towards a given target object goal and call stop when sufficiently nearby (at least $1m$ close to the object is the standard threshold). Specifically, we target navigation towards rare, long-tailed object distributions as contemporary methods often struggle with such target goals. We discuss the datasets and metrics considered for evaluation, after which we move on to the results provided in the static and dynamic evaluation cases.

7.1 Datasets

Our goal is to select datasets that focus on long-tailed object goals, allowing us to evaluate agents based on its ability to navigate to such challenging target goals. For this reason, we selected OVON [44] and HSSD-rare.

Firstly, OVON provides three validation splits (*seen, seen synonyms and unseen*), with each split containing varying levels of object specificity. The intention of this dataset is to test a trained agent's capability to generalize beyond the training set of objects. In this regard, the val_seen split consists of objects seen during training, while the val_unseen split is made of categories not seen nor semantically similar to those during training. Both these splits contain the same amount of specificity, in contrast to the $val\_seen\_synonyms$ which contain categories synonymous to those seen during training, resulting in a more specific descrip-

tion of a category. Thus, we consider the *val_seen_synonyms* subsplit as the relatively optimal set for long-tailed categories. A sub-sample of the categories present in each split is given in Table 7.1.

val_seen	*val_seen_synonyms*	*val_unseen*
table	dining table	dresser
chair	folding chair	radiator
drawers	chest of drawers	nightstand
sink cabinet	kitchen cabinet	dishwasher

Table 7.1: Sample Categories in each sub-split in OVON Dataset. Notably, the synonyms split is more specific and arguably "long-tailed" compared to the other two splits.

However, merely considering synonyms does not constitute our definition of long-tailed, which constitues an object that is of a specific brand, shape, size or characteristic setting it apart from its generic category label. For example, "Ferri Upholstered Leather Metal Bench" instead of the generic "Bench".

The HSSD-Hab [22], unlike HM3D [30] scenes (to which OVON [44] belongs), offers a diverse set of long-tail objects because objects are directly rendered in the scene from their .glb files. This results in a high variety collection of uncommon and diverse objects. Therefore, as detailed in Section 5, we release a specific dataset for this, namely HSSD-rare. Overall, these features makes these datasets a strong testbed for evaluating IMPRINT, demonstrating its effectiveness and suitability for complex, open-set real-world scenarios.

7.2 Metrics

The results are evaluated using two main metrics:

7.2.1 Success Rate (SR)

$$\mathrm{SR}(th) = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \mathbf{1}\left(\text{dist}_i \leq th\right) \tag{7.1}$$

where $\mathrm{SR}(th)$ represents the number of successful location prediction episodes based on a given threshold (i.e. 1m, 2m, 3m in the experiments). We note that from a practical standpoint, defining a strict “correct” threshold is challenging, as predictions within the 2m range may still be valid if the agent can look

around and locate the target in real-world scenarios. However, from a methodological perspective, we acknowledge that in ObjectNav [3], a 1m threshold is the standard, and we primarily adhere to this convention. Nevertheless, we experimented with different thresholds to further demonstrate the robustness of the approach.

7.2.2 Distance to goal (DTG)

This is defined as:

$$\text{DTG} = \left\|\mathbf{p}_{\text{pred}} - \mathbf{p}_{\text{gt}}\right\|_2 = \sqrt{\left(\sum_{i=1}^{n} \left(p_{\text{pred},i} - p_{\text{gt},i}\right)^2\right)} \tag{7.2}$$

DTG is computed as the Euclidean distance between the predicted position and the ground truth target location. The navigable point associated with the predicted location is not explicitly considered, as the distance to it is minimal and is not the main scope of the work.

7.3 Static Evaluation Results

In Static Evaluation, the agent maps the entire scene and the saved feature map is then used to localize the target goal object. We do this using the BLIP2 and SED feature extractors on two different datasets, HSSD-rare and OVON synonyms. Additionally, we test two different standard baselines for comparision, VLFM and OneMap. These are passed through the same pipeline, and their respective saved feature maps are invoked for obtaining predictions and evaluating. The results for this static evaluation is shown in Table 7.2.

In the showcased results, for each prediction, we compare the performance across different fronts. We consider the effect of the used dataset (HSSD-rare, OVON), the feature extractor model (BLIP2, SED), and the mode of query used (Text, Image, Both). The baselines also follow this pattern, where VLFM uses the BLIP2 model, while OneMap uses the SED model. Also, for each prediction, we consider the top-k similarity scores in order to have a robust understanding of the framework behaviour.

Dataset	Encoder	Mode	SR (1m)↑			SR (2m)↑			SR (3m)↑			DTG (m)↓		
			$topk_1$	$topk_3$	$topk_5$	$topk_1$	$topk_3$	$topk_5$	$topk_1$	$topk_3$	$topk_5$	$topk_1$	$topk_3$	$topk_5$
HSSD-rare	**BLIP2**	Text	11.97	23.42	30.47	22.32	35.45	43.42	29.88	43.82	51.98	7.52	5.19	4.11
		Image	15.42	28.93	36.34	28.19	40.75	48.90	35.32	49.34	57.49	6.92	4.83	**3.75**
		Both	**15.49**	**29.07**	**36.64**	**28.41**	**40.82**	**49.12**	**35.46**	**49.49**	**57.86**	**6.91**	**4.81**	**3.75**
	SED	Text	8.00	14.76	19.53	15.46	25.62	33.11	21.07	34.65	44.27	8.18	5.47	4.43
		Image	10.5	20.26	26.87	22.1	33.11	41.04	29.44	42.0	50.44	7.09	**5.33**	4.34
		Both	**10.79**	**20.34**	**27.39**	**22.61**	**33.19**	**41.78**	**30.47**	**42.44**	**50.95**	**7.02**	**5.33**	**4.32**
OVON synonyms	**BLIP2**	Text	38.21	57.72	59.35	**57.72**	65.85	70.73	64.23	71.54	77.24	3.50	2.64	2.21
		Image	42.28	56.91	**65.85**	**57.72**	71.54	**77.24**	69.11	**77.20**	79.67	3.20	2.41	1.92
		Both	**43.09**	**59.36**	65.04	**57.72**	**72.36**	76.42	**69.92**	76.42	**80.49**	**3.08**	**2.34**	**1.89**
	SED	Text	**34.15**	**52.02**	56.10	47.90	62.60	69.11	56.10	73.17	78.86	**3.55**	**2.53**	2.04
		Image	29.27	47.15	**56.91**	**47.97**	**65.04**	**73.11**	**57.72**	73.18	**82.93**	3.87	2.77	**1.98**
		Both	32.52	47.15	54.47	47.0	64.23	71.54	51.72	**73.98**	81.30	3.92	2.74	2.08
HSSD-rare	**BLIP2 (VLFM)**	Text	13.78	18.05	19.12	23.75	28.27	29.22	33.14	36.82	38.72	6.76	6.36	6.25
		Image	17.81	21.38	23.28	**31.35**	**34.80**	**36.82**	39.31	42.28	**44.30**	6.09	5.78	**5.61**
		Both	**17.93**	**22.09**	**23.52**	**31.35**	**34.80**	36.22	**39.43**	**42.40**	44.06	**6.05**	**5.75**	5.62
	SED (OneMap)	Text	25.65	**36.1**	41.21	42.04	50.83	56.41	49.29	59.03	64.25	5.07	3.88	3.30
		Image	25.53	35.27	41.21	45.37	52.61	59.38	52.49	59.98	66.15	4.57	3.73	3.23
		Both	**26.13**	**36.1**	**42.04**	**45.61**	**53.80**	**60.21**	**52.97**	**61.28**	**67.34**	**4.51**	**3.64**	**3.12**

Table 7.2: IMPRINT Static Evaluation results with both feature extractor configurations, and two baselines (VLFM and OneMap) for comparison.

7.3.1 BLIP2 Experiments

BLIP2 [23] experiments, including the baseline VLFM, show a significant performance improvement of approximately 4-5% in Success Rate (within the standard 1m threshold) when going from the text-only baseline to the image-only retrieval method. Similarity, the average Distance to Goal measure also improves by 8-13%. This pattern is strong in both the HSSD-rare and OVON datasets, stressing the impactful improvement in incorporating image-based queries. The results improve further on when considering both the text and image queries, providing complementary contextual clues, further enhancing the long-tailed object localization. In BLIP2, almost all predictions trump the other two modes, and in some interesting cases, image-only predictions fare better. These outliers show that text-based queries can even be misleading, resulting in drop in performances. Overall, when using BLIP2, the results support our claim that using images or complementing text-based queries with relevant image queries can significantly improve the target localization capabilities of the agent, at least by 4-5%.

7.3.2 SED Experiments

SED [43] experiments, on the other hand, showcase interesting and varying results as we shift the concerned datasets. In HSSD-rare, a dataset specifically curated for long-tailed categories, both IMPRINT and the baseline OneMap showcase a general improvement of 1-3% in SR (1m) when going from text-based to other modes, albeit a smaller increase than seen in the BLIP2 experiments. This does provide support for our claim to include image-based queries, but not a strong one. When coming to the OVON dataset, the results waver from one side to the other, some supporting using text-only queries while others support image-only queries, with little to none supporting the use of both.

Firstly, IMPRINT's implementation of SED and its mapping pipeline is rudimentary, in the sense that all patch embeddings are mapped onto its corresponding spatial patches without any filtering or consideration about the reliability of the patch or its diverse space it spans. For example, patches spanning diverse and unrelated parts of the retrieved image are included for completion. Such considerations have been taken into account in the work of OneMap, and this can explain the higher SR scores and a clearer pattern favouring including image-based queries. Nonetheless, the gains are still small, possibly suggesting a case against the use of SED and its patch-based embeddings for constructing queryable feature maps.

Secondly, the confusing and fluctuating results occuring in the OVON case can be possibly due to its absence of true long-tailed categories, as opposed to HSSD-rare. Such fluctuating behaviour can be seen, to a smaller degree, in the case of BLIP2 experiments with OVON. Since SED might not be optimal for building feature maps, coupled with a low-quality dataset, performance might have been affected by such factors. This further reinforces the need for HSSD-rare, a task-specific and long-tailed dataset. Therefore, the OVON results can be interpreted as catering towards generic categories, rather than long-tailed ones, and we note that using BLIP2 and image-based queries does lead to a 4-5% increase, while using SED for the generic categories would not be ideal.

Overall, for our goal of enhancing long-tailed navigation, the results, particularly of HSSD-rare with BLIP2 encoder, showcase strong increase in performance and target localization, with at least 4-5% increase in the Success Rate (SR within 1m) and 8-13% increase in the Distance to Goal metric.

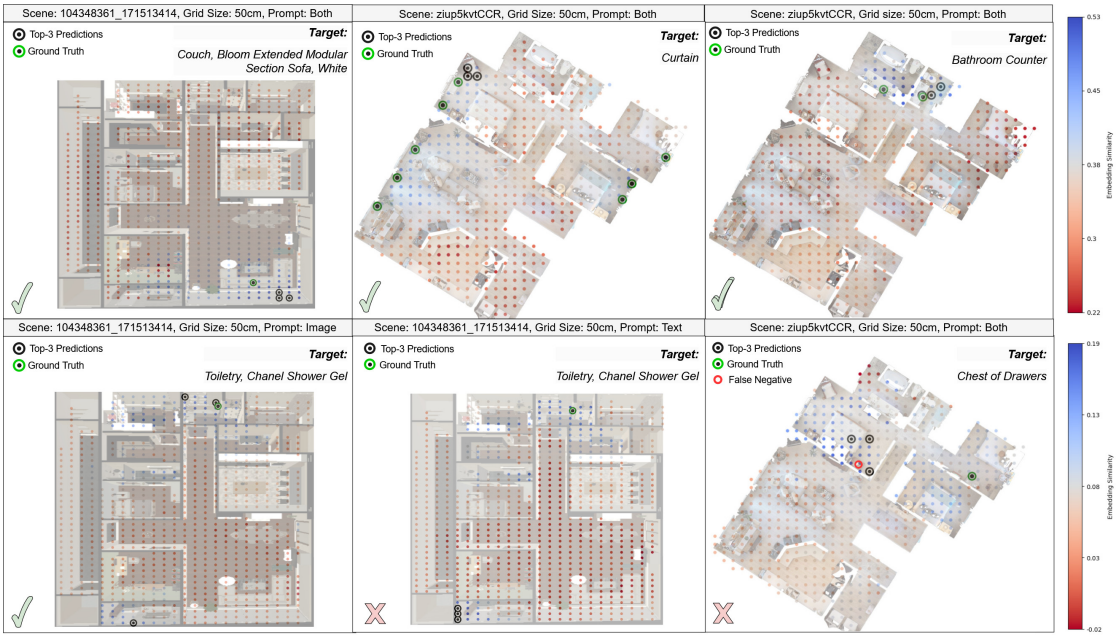

Figure 7.1: Qualitative Examples from IMPRINT Evaluation.

7.3.3 Ablations

We conducted ablation studies on our approach at multiple levels: Number of retrieved images, Grid Size, Reduction method for multiple similarity grids. All ablations were evaluated on Success Rate within $1m$ and Distance to Goal metrics, and the optimal values chosen to test IMPRINT and compare against baselines, as shown in Table 7.2 and discussed in the previous sections.

Ablation: Number of Images

We analyzed the impact of varying the number of online-retrieved images on the localization results, as summarized in Table 7.3. As we went increasing the retrieved number, we noticed a steady increase in performance, topping at the maximum number of 15 images. While the results suggest potential performance gains from increasing this limit to 20-30 images (as metrics do not fully saturate at 15 for BLIP2 configuration), we prioritized practical considerations (Table 6.1). To ensure seamless integration into the existing framework and maintain time efficiency for quick retrieval, we selected 15 images as the optimal solution. This threshold minimizes computational overhead while preserving competitive performance.

		SR (1m)↑			**DTG** (m)↓		
HSSD-rare	$N°$ **Images**	$topk_1$	$topk_3$	$topk_5$	$topk_1$	$topk_3$	$topk_5$
Text	-	11.97	23.42	30.46	7.52	5.19	4.11
BLIP2 [23]	1	12.33	24.67	31.50	7.40	5.09	4.04
	3	15.27	27.09	33.70	7.06	4.99	3.91
	5	14.90	27.39	35.02	7.10	4.88	3.78
	10	15.05	28.41	35.54	7.02	**4.80**	3.79
	15	**15.42**	**28.93**	**36.34**	**6.92**	4.83	**3.75**
Text	-	8.00	14.76	19.53	8.18	5.47	4.43
SED [43]	1	9.69	17.84	23.42	7.45	5.47	4.54
	3	**10.79**	**20.85**	26.06	7.21	**5.29**	4.34
	5	10.35	20.78	26.36	7.16	5.36	4.37
	10	10.21	20.48	**26.95**	7.18	5.34	4.34
	15	10.50	20.26	26.87	**7.09**	5.33	**4.33**

Table 7.3: HSSD-rare number of retrieved images ablation, with a grid size of 100cm.

An interesting pattern, visible both in the BLIP2 and SED cases in Table 7.3, is a performance peak when retrieveing only 3 images, and subsequent drop. This can possibly be the result of noise in web-queried images, as in images not corresponding to the target object or a crowded frame with the target object not particulary visible. Such images can adversely impact the results, particularly when extracted in bulk. This manifests as degraded feature embeddings, where the aggregated visual representation becomes less discriminative as more noisy samples are included. As expected, this is more start in SED due to its reliance on patch embeddings, and therefore the quality of the image and the semantic objects within its frame.

Additionally, we would like to point out that, in many cases, the most informative metric of an agent's behaviour is the Distace to Goal, rather than the reductive Success Rate. In this regards, for both BLIP2 and SED, a good choice for the image parameter is the number 15.

Ablation: Grid Size

To test for the optimal grid size, we considered 20cm, 50cm and 100cm as our possibile options. We fixed the upper threshold to be $1m$ as this is the standard threshold at which the agent's success is determined on reaching a target object.

		SR (1m)↑			**DTG** (m)↓		
HSSD-rare	**Grid Size**	$topk_1$	$topk_3$	$topk_5$	$topk_1$	$topk_3$	$topk_5$
BLIP2 [23]	20 (cm)	16.01	21.15	23.13	**6.70**	6.12	5.80
	50 (cm)	**16.96**	25.99	30.91	6.74	5.44	4.85
	100 (cm)	15.42	**28.93**	**36.34**	6.92	**4.83**	**3.75**
SED [43]	20 (cm)	**11.45**	15.35	17.77	7.22	6.51	6.17
	50 (cm)	11.16	**20.34**	24.3	**7.05**	5.96	5.34
	100 (cm)	10.50	20.26	**26.87**	7.09	**5.33**	**4.33**

Table 7.4: HSSD grid size ablation: The number of images for this configuration is set to 3 for SED and 15 for BLIP2.

A grid size larger than $1m$ would lead to information loss as a single feature vector can fail to capture all the semantics contained within, and also the spatial localization would be dispersed further more than is preferable.

Table 7.4 indicates that considering a large grid size, allowing the feature vector to capture a holistic picture while not focusing too much on the fine-grained details, performs the best. Using a grid size too small might fail to capture the full semantic context and relations spanning grid cells, which can in turn compromise the quality of the resulting embedding.

Ablation: Reduction Method

		SR (1m)↑ - $topk_1$		
		Method		
Encoder	Dataset	Mean	Harmonic mean	Hybrid
BLIP2 [23]	OVON [44]	**38.73**	25.35	37.32
	HSSD-rare	**15.49**	11.75	12.85
SED [43]	OVON [44]	**34.15**	33.33	33.33
	HSSD-rare	**10.50**	6.02	7.93

Table 7.5: Extraction of single embedding given multiple images as input to the VLM. Hybrid stands for a mixture of both simple mean and harmonic mean. Grid size is set to 100cm.

Table 7.5 shows the various reduction methods considered for reducing multiple similarity grids, as a result of multiple retrieved images or a combination of text and image queries, into one final similarity grid. In Mean and Harmonic Mean, all the grids are equally weighted and the corresponding mean is taken.

In the Hybrid method, we consider the harmonic mean of the text-derived grid and a collective image-derived grid, which is the result of an arithmetic mean through all image-derived similarity grids. This weights the text-query and the collective set of image-queries in equal footing, possibly leading to a more balanced similarity grid.

However, the method leading to the best result is that of the simple arithmetic mean with equal weightage through all considered grids.

7.4 Dynamic Evaluation Results

Object Goal Navigation, as a task, requires an agent to navigate an environment to eventually find the target goal object, and this implicitly contains the requirement that the agent is to conduct the exploration, mapping and navigation in real time. Therefore, although static evaluation has its merits, it is both economically preferable and time conserving for an agent to follow a real-time navigation pipeline, especially if the current scene is too large to be mapped within a short time span. Therefore, going by the progress we built from the static evaluation, we choose the following optimal parameter choices:

- **Dataset:** We consider HSSD-rare, since it is specifically created for our goal of improving long-tailed object localization and navigation. OVON Synonyms contained more generic object categories, and likewise, did not portray the strong performance improvement showcased when using HSSD-rare.
- **Vision-Language Encoder:** We consider BLIP2, since going by both the ablations and final results, this turned out to be the best feature extractor for constructing the queryable map, leading to larger gains in performance in both the datasets considered. Moreover, BLIP2 is also time-efficient, while SED requires patching the images and producing patch embeddings that capture the spatial and semantic relations between the patches.

Using the dynamic navigation pipeline, as described in Section 6.0.4, we evaluate IMPRINT on HSSD-rare using the BLIP2 feature extractor on a total of 17 scenes containing 700 episodes. The navigation results are given in Table 7.6, with the distance to viewpoints threshold varied by different amounts. In every case, we notice a 3-4% increase in SR (within $1m$), similar to that observed in the Static evaluation case. Notably, when the threshold is $0.5m$, we get navigation results that closely resemble the static results. This validates our approach, showcasing that our static results transfer even to the dynamic, real-time case.

Success Threshold	Query Prompt	Success Rate	SPL (Avg)
0.10m	Text	4.71	1.38
	Image	**6.14**	**1.90**
0.30m	Text	9.42	2.27
	Image	**12.28**	**3.63**
0.50m	Text	11.71	2.77
	Image	**15.14**	**4.31**
1.00m	Text	15.42	3.50
	Image	**19.57**	**5.31**

Table 7.6: IMPRINT Navigation results, while varying the distance to target viewpoints threshold, used for determining if an episode was a success or not. The results clearly show a preference for image-based queries, with a 4% SR increase.

In comparison with the VLFM implementation, IMPRINT is faster by 90.9% and consumes 99.9% less memory when building the feature maps, as is shown in Table 7.7. This confirms the validity and real-world benefits of IMPRINT over VLFM or other dense mappings.

Method*	**GFLOPs**↓	**Size** (GB)↓	**Time**(s)**
VLFM	~ 6.91	~ 2.86	5.66 ± 1.34
IMPRINT	~ 0.02	< 0.001	1.06 ± 0.09

Table 7.7: Time Ablation for IMPRINT against baseline VLFM. This proves IMPRINT is much more preferable than the latter, especially in real-world deployment.

However, in absolute terms, the success rates obtained are not on par with state-of-the-art methods, prompting possible areas of improvement within the static and dynamic pipeline. Notably, we have not employed the use of object detection models, which is the standard in contemporary models. Furthermore, the agent switches from exploration to navigating to the predicted target location based on a rudimentary threshold value (40%). This can be improved with the use of a more developed, sophisticated policy where the target location can be chosen with either a dynamic threshold or another suitable alternative.

8

Conclusions and Future Works

The field of Embodied AI has provided a lot of exciting challenges, ones which inch us closer to an embodied interpretation of artificial general intelligence. In this dissertation, working on a subset of this vast field, specifically in Object Goal Navigation, has been very fruitful on three fronts. On one front, we were able to make accessible two of the most influential and contemporary mapping methods active in the field. Developing ModNav, while ensuring parallel computation and an accessible interface, helped make scene mapping more approachable and enabled easy cross-experimentation with a mapping and policy of one's choice. Secondly, we were able to contribute towards and extend the widely used HSSD dataset, developing HSSD-rare, a task-specific dataset specifically curated towards rare, long-tailed object categories. Building this ObjectNav dataset required coming up with novel ways to generate viewpoints, while following the constraints in creating a dataset for the Habitat framework. On a final front, with reliance upon established approaches present in the community, we were able to develop IMPRINT, a novel approach, both time efficient and memory saving ($\sim$ 99.75%), that leverages the visual contextual clues present in online images to complement text queries when navigating towards a target goal. This idea was morphed into a queryable map, which can be used as a modular reference to create policies and path planning modules. This queryable map, constructed out of grid cells containing spatial feature vectors, was used by the agent for efficient spatial localization of the target query. This improved the success rate by 4-8% in HSSD-rare, confirming the better efficiency in navigating towards the challenging long-tailed object categories.

Although the task of navigating to long-tailed categories have not been explored previously, the results obtained with IMPRINT (in terms of success rates) are not on par with state-of-the-art approaches in other general tasks. One possible extension is to develop a better pipeline that leverages the image-based queries, like using a more refined projection algorithm that casts feature vectors to grids only corresponding to them, or considering an image-conditioned object detector for fine-grained object detections. Another possibility is to better the query contexts in the form of prompt engineering. The search queries to retrieve relevant images can be updated to include more context regarding the object or alternate angles of the object, like the back of a dining sofa. This can add more contextual clues to the feature vector extracted, and help in identifying objects better and from any angles. Additionally, apart from IMPRINT, ModNav can be expanded to include other influential methods like VLMaps [17] or OneMap [5].

In conclusion, our approach (IMPRINT), dataset (HSSD-rare) and framework (ModNav) have contributed to the field of Embodied AI in diverse fronts, each progressing on challenging and relevant tasks. Although further improvements can be done to extend our work on these fronts, this finished work forms a solid, reliable basis upon which such expansions can be built on.

9 Appendix

9.1 Using Relative Pose Change to reach next pose

Given a local frame with the current pose: (x_1, y_1, o_1), and the change in pose (with respect to to the same local frame): (dx, dy, do), we want to reach the next pose and shift to its local frame. Intuitively, we go in two steps:

- **Rotate Pose Change:** We first get the change in position: (dx, dy). However, this is in the general frame, which is the current local frame with orientation along the x-axis. We need to rotate it using the pose orientation o_1. This orients the position change to the current local frame.
- **Summation:** Now that both the current and relative change poses are in the same local frame, we can add it to get the new pose. Note that this new pose is also expressed in the current local frame.
- **New Orientation:** Now that we have reached the new position, we need to use do to orient the current local frame to the new orientation of the next local frame.

These steps can be expressed as follows:

$$\begin{bmatrix} x_2 \\ y_2 \\ o_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ o_1 \end{bmatrix} + \begin{pmatrix} \cos(o_1) & -\sin(o_1) & 0 \\ \sin(o_1) & \cos(o_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} dx \\ dy \\ do \end{bmatrix} \tag{9.1}$$

Using equations 4.1 and 9.1, we can find the result of matrix multiplication and confirm that this indeed results in global movement from one pose to the next.

$$
\begin{aligned}
x_2 &= x_1 + \left(dx \ \cos(o_1) - dy \ \sin(o_1) \right) \\
&= x_1 + \left(d \ \cos(360 - \theta - o_1) \cos(o_1) - d \ \sin(360 - \theta - o_1) \sin(o_1) \right) \\
&= x_1 + d \cos(360 - \theta - o_1 + o_1) \\
&= x_1 + d \cos(\theta)
\end{aligned}
$$

$$
\begin{aligned}
y_2 &= y_1 + \left(dx \ \sin(o_1) + dy \ \cos(o_1) \right) \\
&= y_1 + \left(d \ \cos(360 - \theta - o_1) \sin(o_1) + d \ \sin(360 - \theta - o_1) \cos(o_1) \right) \\
&= y_1 + d \sin(360 - \theta - o_1 + o_1) \\
&= y_1 - d \sin(\theta)
\end{aligned}
$$

$$
o_2 = o_1 + do
$$

Since the global change in pose is as expected, this confirms the validity of the earlier derivation for obtaining local relative difference of poses.

9.2 Calculating Vertical Field of Vision (VFOV)

We can directly obtain the vertical FOV from the horizontal FOV and the image sensor's height and width. The formula provided in the standard VLFM's code contained a slight error, as it uses the diagonal FOV to transition from the horizontal to the vertical FOV, which is unnecessary. I will mention both the direct method and the corrected method for the sake of completeness:

Without DFOV:

$$
\tan\left(\frac{h_{fov}}{2}\right) = \frac{W}{2f}, \quad \tan\left(\frac{v_{fov}}{2}\right) = \frac{H}{2f}, \quad \tan\left(\frac{d_{fov}}{2}\right) = \frac{\sqrt{H^2 + W^2}}{2f}
$$

$$
\frac{\tan\left(\frac{v_{fov}}{2}\right)}{\tan\left(\frac{h_{fov}}{2}\right)} = \frac{H}{W}
$$

$$v_{fov} = 2 \cdot \tan^{-1}\left(\tan\left(\frac{h_{fov}}{2}\right) \cdot \frac{H}{W}\right)$$

With DFOV: Similar to the above case, we get:

$$d_{fov} = 2 \cdot \tan^{-1}\left(\tan\left(\frac{h_{fov}}{2}\right) \cdot \frac{\sqrt{H^2 + W^2}}{W}\right)$$

$$v_{fov} = 2 \cdot \tan^{-1}\left(\tan\left(\frac{d_{fov}}{2}\right) \cdot \frac{H}{\sqrt{H^2 + W^2}}\right)$$

References

[1] Kush Agrawal. *To study the phenomenon of the Moravec's Paradox*. 2010. arXiv: 1012.3148 [cs.AI]. URL: https://arxiv.org/abs/1012.3148.

[2] Dong An et al. *1st Place Solutions for RxR-Habitat Vision-and-Language Navigation Competition (CVPR 2022)*. 2022. arXiv: 2206.11610 [cs.CV]. URL: https://arxiv.org/abs/2206.11610.

[3] Dhruv Batra et al. "Objectnav revisited: On evaluation of embodied agents navigating to objects". In: *arXiv preprint arXiv:2006.13171* (2020).

[4] Valts Blukis et al. *A Persistent Spatial Semantic Representation for High-level Natural Language Instruction Execution*. 2021. arXiv: 2107.05612 [cs.RO]. URL: https://arxiv.org/abs/2107.05612.

[5] J. Busch et al. "OneMap to Find Them All: A Spatial Semantic Map for Multi-Query Object Search". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.

[6] Murray Campbell, A.Joseph Hoane, and Feng-hsiung Hsu. "Deep Blue". In: *Artificial Intelligence* 134.1 (2002), pp. 57–83. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00129-1. URL: https://www.sciencedirect.com/science/article/pii/S0004370201001291.

[7] Angel Chang et al. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *International Conference on 3D Vision (3DV)*. 2017.

[8] Devendra Singh Chaplot et al. *Learning to Explore using Active Neural SLAM*. 2020. arXiv: 2004.05155 [cs.CV]. URL: https://arxiv.org/abs/2004.05155.

[9] Devendra Singh Chaplot et al. *Object Goal Navigation using Goal-Oriented Semantic Exploration*. 2020. arXiv: 2007.00643 [cs.CV]. URL: https://arxiv.org/abs/2007.00643.

[10] Xinlei Chen et al. "Microsoft coco captions: Data collection and evaluation server". In: *arXiv preprint arXiv:1504.00325* (2015).

[11] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

[12] Matt Deitke et al. *ProcTHOR: Large-Scale Embodied AI Using Procedural Generation*. 2022. arXiv: 2206.06994 [cs.AI]. URL: https://arxiv.org/abs/2206.06994.

[13] Matt Deitke et al. *Retrospectives on the Embodied AI Workshop*. 2022. arXiv: 2210.06849 [cs.CV]. URL: https://arxiv.org/abs/2210.06849.

[14] Jiafei Duan et al. *A Survey of Embodied AI: From Simulators to Research Tasks*. 2022. arXiv: 2103.04918 [cs.AI]. URL: https://arxiv.org/abs/2103.04918.

[15] David Hall et al. *The Robotic Vision Scene Understanding Challenge*. 2020. arXiv: 2009.05246 [cs.RO]. URL: https://arxiv.org/abs/2009.05246.

[16] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[17] C. Huang et al. "VLMaps: Vision-Language Maps for Robot Navigation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[18] Gabriel Ilharco et al. *General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping*. 2019. arXiv: 1907.05446 [cs.RO]. URL: https://arxiv.org/abs/1907.05446.

[19] John M. Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589. URL: https://api.semanticscholar.org/CorpusID:235959867.

[20] Abhishek Kadian et al. "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 6670–6677. ISSN: 2377-3774. DOI: 10.1109/lra.2020.3013848. URL: http://dx.doi.org/10.1109/LRA.2020.3013848.

[21] Apoorv Khandelwal et al. *Simple but Effective: CLIP Embeddings for Embodied AI*. 2022. arXiv: 2111.09888 [cs.CV]. URL: https://arxiv.org/abs/2111.09888.

[22] Mukul Khanna et al. "Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

[23] Junnan Li et al. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models". In: *International conference on machine learning*. PMLR. 2023.

[24] Shilong Liu et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *European Conference on Computer Vision*. Springer. 2024.

[25] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. "Scaling open-vocabulary object detection". In: *Advances in Neural Information Processing Systems* 36 (2023).

[26] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[27] Radford M. Neal. *Bayesian Learning for Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN: 0387947248.

[28] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[29] Rolf Pfeifer and Fumiya Iida. "Embodied Artificial Intelligence: Trends and Challenges". In: *Embodied Artificial Intelligence: International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003. Revised Papers*. Ed. by Fumiya Iida et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–26. ISBN: 978-3-540-27833-7. DOI: 10.1007/978-3-540-27833-7_1. URL: https://doi.org/10.1007/978-3-540-27833-7_1.

[30] Santhosh K Ramakrishnan et al. "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai". In: *arXiv preprint arXiv:2109.08238* (2021).

[31] Paul Resnick et al. "GroupLens: an open architecture for collaborative filtering of netnews". In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW '94. Chapel Hill, North Carolina, USA: Association for Computing Machinery, 1994, pp. 175–186. ISBN: 0897916891. DOI: 10.1145/192844.192905. URL: https://doi.org/10.1145/192844.192905.

[32] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[33] Manolis Savva et al. "Habitat: A Platform for Embodied AI Research". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[34] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (2016), pp. 484–503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[35] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435.

[36] Julian Straub et al. "The Replica Dataset: A Digital Replica of Indoor Spaces". In: *arXiv preprint arXiv:1906.05797* (2019).

[37] Richard Szeliski. *Computer vision algorithms and applications*. 2011. URL: http://dx.doi.org/10.1007/978-1-84882-935-0.

[38] Adly Templeton et al. "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet". In: *Transformer Circuits Thread* (2024). URL: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

[39] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[40] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[41] Luca Weihs et al. *Visual Room Rearrangement*. 2021. arXiv: 2103.16544 [cs.CV]. URL: https://arxiv.org/abs/2103.16544.

[42] Fei Xia et al. “Gibson env: Real-world perception for embodied agents”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[43] Bin Xie et al. “Sed: A simple encoder-decoder for open-vocabulary semantic segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024.

[44] Naoki Yokoyama et al. “HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation”. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024.

[45] Naoki Yokoyama et al. “Vlfm: Vision-language frontier maps for zero-shot semantic navigation”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024.