Autonomous Embodied Agents:
When Robotics Meets Deep Learning Reasoning

Ph.D. Dissertation

Roberto Bigazzi

Advisor:

    Prof. Rita Cucchiara   University of Modena and Reggio Emilia

Director of the School:

    Prof. Sonia Bergamaschi   University of Modena and Reggio Emilia

Review Committee:

    Prof. Shreyas Kousik   Georgia Institute of Technology
    Prof. Matteo Matteucci   Polytechnic University of Milan

This dissertation was typeset by the author using LaTeX 2_ε, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface.

To the most important women in my life, Gloria and my mother.

Gloria has been a constant source of love and unwavering support throughout my numerous ups and downs. My mother has always been my biggest fan and a source of inspiration for her strength.

None of this would have been possible without them.

Autonomous Embodied Agents:
When Robotics Meets Deep Learning Reasoning

Abstract

The increase in available computing power and the Deep Learning revolution have allowed the exploration of new topics and frontiers in Artificial Intelligence research. A new field called Embodied Artificial Intelligence, which places at the intersection of Computer Vision, Robotics, and Decision Making, has been gaining importance during the last few years, as it aims to foster the development of smart autonomous robots and their deployment in society. The recent availability of large collections of 3D models for photorealistic robotic simulation has allowed faster and safe training of learning-based agents for millions of frames and a careful evaluation of their behavior before deploying the models on real robotic platforms. These intelligent agents are intended to perform a certain task in a possibly unknown environment. To this end, during the training in simulation, the agents learn to perform continuous interactions with the surroundings, such as gathering information from the environment, encoding and extracting useful cues for the task, and performing actions towards the final goal; where every action of the agent influences the interactions. This dissertation follows the complete creation process of embodied agents for indoor environments, from their concept to their implementation and deployment.

In the first part of this work, we study the importance of building efficient representations of the agent's knowledge aimed at its understanding of the world and learning capabilities on the task to pursue. We devise and examine two alternative approaches to implicitly encode and maximize the information collected without the need for annotated data, which are usually costly and difficult to produce. The first explored method rewards actions that produce a significant

I

change in the agent's knowledge or internal representation of the environment and is called Impact. The second approach, instead, is called Curiosity, and as human curiosity does, it encourages the agent to explore states of the environment where it can see or learn new things.

The investigation of implicit representations for embodied agents is followed by a study of agents' behavior on various robotic tasks, both in simulated and real settings.

Following, we investigate the last step for a successful implementation of an autonomous agent: the deployment of the trained models on a real robot. We study how to transfer the knowledge acquired in simulation into the real world, considering and coping with the architectural discrepancies between those worlds to minimize the degradation caused by the simulation-to-reality transfer.

The final part of this work presents the acquisition and public release of a photo-realistic 3D model of an art gallery accompanied by a dataset for navigation. This contribution enlarges the number of datasets available in the literature and enables simulated robot navigation inside museums.

With this thesis, we aim to contribute to research in Embodied AI and autonomous agents, in order to foster future work in this field. We present a detailed analysis of the procedure behind implementing an intelligent embodied agent, comprehending a thorough description of the current state-of-the-art in literature, technical explanations of the proposed methods, and accurate experimental studies on relevant robotic tasks.

Agenti Incorporati Autonomi:
Quando la Robotica Incontra il Ragionamento con Apprendimento Profondo

Sommario

L'incremento della potenza di calcolo disponibile e la rivoluzione del Deep Learning hanno aperto nuovi temi e frontiere nella ricerca sull'Intelligenza Artificiale. Un nuovo campo chiamato Intelligenza Artificiale Incorporata (Embodied Artificial Intelligence), che si colloca al confine tra Computer Vision, Robotica e Decision Making, sta guadagnando importanza negli ultimi anni, in quanto mira a promuovere lo sviluppo e l'impiego nella società di robot autonomi intelligenti. La recente disponibilità di grandi collezioni di modelli 3D per la simulazione robotica fotorealistica ha permesso un addestramento più rapido e sicuro di agenti intelligenti usando milioni di fotogrammi, unito ad un'attenta valutazione del loro comportamento prima di distribuire i modelli su robot reali. Questi agenti intelligenti devono svolgere un determinato compito in un ambiente potenzialmente sconosciuto. A questo fine, durante l'allenamento in simulazione, gli agenti imparano ad eseguire interazioni continue con l'ambiente circostante, come la raccolta di informazioni dall'ambiente e la codifica ed estrazione di dati utili per l'esecuzione del compito assegnato; dove ogni azione dell'agente influenza tali interazioni. Questa tesi segue l'intero processo di creazione di agenti da ambienti interni, dalla loro concezione alla loro implementazione.

Nella prima parte di questo lavoro, studiamo l'importanza di costruire rappresentazioni efficienti della conoscenza dell'agente finalizzate alla sua comprensione del mondo e alle capacità di apprendimento del compito da perseguire. Abbiamo ideato ed esaminato due approcci alternativi per codificare implicitamente e massimizzare le informazioni raccolte senza la necessità di dati annotati, che di solito

sono costosi e difficili da produrre. Il primo metodo esplorato premia le azioni che producono un cambiamento significativo della conoscenza dell'agente o nella rappresentazione dell'ambiente, ed è chiamato Impact (Impatto). Il secondo approccio, invece, è chiamato Curiosity (Curiosità) e, come fa la curiosità umana, incoraggia l'agente a esplorare gli stati dell'ambiente in cui può vedere o imparare cose nuove.

L'indagine sulle rappresentazioni implicite per gli agenti embodied è seguita da uno studio del comportamento degli agenti in vari compiti robotici, sia in ambienti simulati che reali.

A seguire, proponiamo uno studio sull'ultimo passo per la creazione con successo di un agente autonomo: l'implementazione dei modelli addestrati su un robot reale. Investighiamo come trasferire nel mondo reale le conoscenze acquisite in simulazione, considerando e affrontando le discrepanze architettoniche tra questi due mondi in modo da minimizzare il peggioramento delle prestazioni causato dal trasferimento da simulato a reale.

La parte finale di questo lavoro presenta l'acquisizione e pubblicazione di un modello 3D fotorealistico di una galleria d'arte, accompagnato da un set di dati per la navigazione. Questo contributo amplia il numero di dataset disponibili in letteratura consentendo la navigazione simulata di robot all'interno dei musei.

Con questa tesi intendiamo contribuire alla ricerca sull'Embodied AI e sugli agenti autonomi, in modo da promuovere il lavoro futuro in questo campo. Presentiamo un'analisi dettagliata della procedura di implementazione di un agente intelligente, che comprende una descrizione approfondita dell'attuale stato dell'arte in letteratura, spiegazioni tecniche dei metodi proposti e accurati studi sperimentali su compiti robotici d'interesse.

Contents

1
Introduction

WHEN we think about life in 100 years from now, how do we imagine it? Most optimist viewers envision flying cars, autonomous transportation, and hi-tech cities, others instead, see a general improvement in life quality and automation in all aspects of everyday life. In any of the "good" scenarios, one detail that cannot be missed in the vision is the inclusion of ubiquitous robotic helpers interacting seamlessly with humans and the surrounding environment.

Recently, we have already seen robots gradually enter industry and society; for example, medical robots can assist surgeons during procedures, help patients with physical therapy, and perform diagnostic tests. Another example is shown in manufacturing settings, where robots perform tasks such as welding, painting, and assembly. At the same time, we have seen an increasing interest in service robots designed for interaction with humans in a variety of settings, such as hospitals, museums, and homes. These robots can assist with tasks such as delivering items, providing information, and performing basic maintenance.

However, robots that are able to really understand and learn, like the ones we see in some movies, are still a long way off. In fact, such robots can perform the

1

task they are programmed for, but they are not capable of reasoning about their surroundings, neither they learn new concepts or knowledge.

As a response to this lack, a new field called Embodied Artificial Intelligence (Embodied AI) gained attention in the research community. Such a field has the objective of fostering the development of the intelligent autonomous agents of the future by combining knowledge in various research areas, including Robotics, Computer Vision, and Natural Language Processing (NLP).

Embodied AI considers robots equipped with sensors and actuators that allow them to perceive, manipulate the environment, move around, and navigate; however, differently from pure Robotics, Embodied AI research mainly focuses on high-level interactions between the agents and the surrounding environment.

The reasoning behind Embodied AI is to create Artificial Intelligence systems that can interact with and operate within the physical world in a natural and intuitive way. Embodied agents can also be designed to be more efficient and effective at performing certain tasks, such as manufacturing or search and rescue, by taking advantage of their physical bodies and ability to move through space. In a nutshell, the goal of Embodied AI is to create intelligent systems that can operate and interact with the world in a manner similar to humans.

Regarding the factors that lead to the emergence of Embodied AI, we can identify some main causes: the advances brought by the so-called Deep Learning revolution, the release of new data devoted to robotic simulation, and the increasing availability of computational power.

By Deep Learning revolution, we refer to the recent breakthroughs in Artificial Intelligence (AI) research that have been made possible using Deep Learning techniques. In fact, Deep Learning lead to many of the major advances in AI over the past decade, for example in image and speech recognition, language translation, or self-driving cars. AI models can now effectively recognize objects, generate natural language sentences, and make decisions aimed at fulfilling a task. Speaking about robotic simulation, in the last years we have seen the introduction of large-scale datasets of real-life apartments, houses, and offices [30, 149, 126, 117], allowing researchers to use simulation to train agents for navigation and other tasks. Besides that, the growth in available computing power, boosted also

by the usage of dedicated general purpose Graphics Processing Units (GPUs), has allowed the development of models that can process and analyze large amounts of data in real-time, and be deployed on physical robots.

1.1 Problem Statement

In order to obtain systems and robots that behave and reason like humans, probably decades of research are still needed, requiring advances in all areas affecting Embodied AI before being able to reach such milestones. With the work presented in this dissertation, we aim to help future research on this topic and take a step toward real intelligent autonomous agents.

One of the problems presented in the previous section is the ineffective ability of embodied agents to reason about their surroundings. To start coping with this issue, an effective way is to design efficient representations to extract useful information from the agent's sensing. Finding the best way to process visual inputs or other auxiliary representations to improve agents' performance is an object of research, and finding efficient representations would allow faster learning phases using fewer data to train agents for a particular task.

Once the agent is capable of performing a task efficiently, it should be able to transfer its knowledge to different tasks without reinitialization. In light of this, a desired goal of our work is finding a task that can be used as a proxy for downstream.

At the same time, a smart agent should also be able to update its knowledge if something changes in the surrounding environment. We want to tackle this issue by finding effective ways to train the agent to search for fresh available information in the environment.

A totally perpendicular direction of research considers embodied agents' inability to provide feedback about their behavior and their perception. We study a way to overcome this problem by providing the robot with the capability to communicate with humans to describe its observations. As a part of this capability, the agent should consider the right moment for producing descriptions and interacting with humans without constantly outputting useless information.

How to choose when the agent should speak is also another interesting aspect we consider in this thesis.

A final problem that we want to address in this work is the correct deployment of agents trained in simulation, where the training is safer, faster, and cheaper, to real robotic platforms without major degradation in their performance. In this respect, we have to consider what discrepancies exist between simulation and reality that we need to address in order to transfer knowledge between simulation and the real world smoothly.

Addressing the aforementioned problems and weaknesses in current embodied agents is a small step towards robots that behave and operate in the real world seamlessly like humans. In the following pages of this dissertation, we will tackle these open questions trying to improve available agents.

1.2 ORGANIZATION

With this dissertation, we present a complete pipeline for the creation of a smart autonomous agent starting from the design of its knowledge representation to the real-world deployment and test. In the following, a brief description of the organization of the work is given.

Starting with Chapter 2, we present a detailed description of the current literature on Embodied AI and complementary research fields that are going to be relevant throughout the work proposed in the thesis.

In Chapter 3 and 4, we present two new implicit rewards for embodied agents for exploration that present interesting benefits also to downstream tasks. Specifically, in Chapter 3, the agent is enriched by a neural mapper that produces occupancy grid maps and we propose a more efficient intrinsic reward named Impact, which encourages the agent to perform actions that produce high variation in its internal representation of the environment. We evaluate agent's performance in navigation towards coordinates showing that an agent able to efficiently explore the environment can be used without major modifications for downstream tasks.

In Chapter 4 we present a new setting where the objective is to produce natural language descriptions while exploring the environment. We name this setting

Explore and Explain. To tackle this task, we propose an intrinsic reward based on artificial curiosity. This reward pushes the agent towards states where a model that generates future observations, produces wrong predictions.

Chapter 5 improves the work presented in the previous chapter testing different map-based exploration agents available in the literature on the new *Explore and Explain* setting, as well as adopting state-of-the-art captioning methods. To evaluate the performance of the models we present a novel metric considering both exploration and language generation measures. This metric highlights the agents that are able to explore effectively the environment and generate informative descriptions without repeating the same information multiple times.

In Chapter 6, we move away from the task of exploration by proposing a new task where the agent has already collected the map of the environment, but the layout of the map is changed from the state stored in the memory of the agent. Some objects are added, moved, or removed. The goal of the agent is to spot the outdated parts in the map and repair it in a given time window. We call this new task *Spot the Difference*.

In Chapter 7, we show that the models trained in simulation can be deployed on a Low-Cost Robot (LoCoBot) [88] without major redesigns. The performance degradation of the models trained in simulation and tested in the real world shows that simulation-to-reality (Sim2Real) transfer still poses a problem and further research is needed to fill the performance gap.

A final contribution is presented in Chapter 8, where we introduce a new dataset for photo-realistic robotic simulation that, instead of containing houses, flats, or offices as the available benchmark datasets, has been collected in an art museum. We show that the peculiar topology of such a building increases the difficulty of embodied tasks such as coordinate-driven navigation, enabling further research on the task in large environments.

Chapter 9 presents the conclusions of the dissertation with some personal considerations and some possible future work and directions of research.

1

2

Literature Survey

In this chapter, we present a literature overview of the relevant work that is related to the tasks and settings proposed in this thesis. We first review the literature on visual exploration, listing related work in both traditional Robotics and Embodied AI fields (Section 2.1). Following, Section 2.2 presents implicit rewards for exploration agents with the work done in the Machine Learning field and the recent work in Embodied AI. Section 2.3 reviews the recent advances in embodied navigation, listing both map-based and map-less methods. In Section 2.4, we show some of the advances in simulating platforms for research, from the simulators used in traditional Reinforcement Learning (RL) to the photo-realistic simulators for Embodied AI agents, including also complementary datasets of 3D models of environments and task-specific datasets. Section 2.5 presents the research on Sim2Real transfer of smart autonomous agents or work that includes real-world deployment. The last part of this chapter contains three sections to briefly present literature on the knowledge of the environment exploited by embodied agents (Section 2.6), Image Captioning (Section 2.7), and deep generative models (Section 2.8) that is related to the work presented in this dissertation.

2.1 Embodied Agents for Exploration

Classical heuristic and geometric-based exploration methods rely on two main strategies: frontier-based exploration [152] and next-best-view planning [50].

The former entails iteratively navigating towards the closest point of the closest frontier, which is defined as the boundary between the explored free space and the unexplored space. The latter entails sequentially reaching cost-effective unexplored points, *i.e.* points from where the gain in the explored area is maximum, weighed by the cost to reach them.

These methods have been largely used and improved [58, 22, 99] or combined in a hierarchical exploration algorithm [162, 130]. However, when applied with noisy odometry and localization sensors or in highly complex environments, geometric approaches tend to fail [38, 99, 118]. In light of this, increasing research effort has been dedicated to the development of learning-based approaches, which usually exploit Deep Reinforcement Learning (DRL) to learn robust and efficient exploration policies.

In fact, even if the final goal of current research on Embodied AI mainly focuses on tasks that require navigating to indoor locations or coordinates, such as Vision-and-Language Navigation (VLN) [9, 80, 78, 35, 53, 37, 36], Point Goal navigation (PointNav), and Object Goal navigation (ObjectNav) [147, 5, 164], Ramakrishnan *et al.* [118] highlighted the importance of visual exploration in order to pretrain a generic embodied agent and identified four paradigms for learning-based visual exploration: novelty-based, curiosity-based, reconstruction-based, and coverage-based. Each paradigm is characterized by a different reward function used as a self-supervision signal for optimizing the exploration policy. A coverage-based reward, considering the area seen, is also used in the modular approach to Active Neural SLAM (ANS) presented in [32], which combines a neural mapper module with a hierarchical navigation policy. To enhance exploration efficiency in complex environments, Ramakrishnan *et al.* [115] resorted to an extrinsic reward by introducing the occupancy anticipation reward, which aims to maximize the agent accuracy in predicting occluded unseen areas, and a method combining frontier-based exploration with a learning-based approach

[116]. A different approach is followed by Georgakis *et al*. [48] that implies an uncertainty-based exploration agent.

We contribute to current research on exploration methods for embodied agents in Chapter 3 and Chapter 4.

2.2 IMPLICIT REWARDS FOR EXPLORATION

The lack of ground truth in the exploration task forces the adoption of Reinforcement Learning (RL) for training exploration methods. Unfortunately, RL methods have low sample efficiency, even when applied to tasks different from robot exploration. Thus, they require designing intrinsic reward functions that encourage visiting novel states or learning the environment dynamics. Furthermore, the use of intrinsic motivation is beneficial also in case the external task-specific rewards are sparse or absent.

In this context, Oudeyer and Kaplan [104] provide a summary of early work on intrinsic motivation. Among them, Schmidhuber [128] and Sun *et al*. [138] proposed to use information gain and compression as intrinsic rewards, while Klyubin *et al*. [74], and Mohamed and Rezende [95] adopted the concept of empowerment as reward during training.

Among the intrinsic rewards that motivate the exploration of novel states, Bellemare *et al*. [15] introduced the notion of pseudo visitation count by using a Context-Tree Switching (CTS) density model to extract a pseudo-count from raw pixels and applied count-based algorithms. Similarly, Ostrovski *et al*. [103] applied the autoregressive deep generative model PixelCNN* [100] to estimate the pseudo-count of the visited state. Recently, Zhang *et al*. [158] proposed a criterion to mitigate common issues in count-based methods.

Rewards that promote the learning of the environment dynamics comprehend Random Network Distillation (RND) [27], Disagreement [109], and Curiosity. Among curiosity-driven exploration methods, the strategy of jointly training forward and backward dynamics models for learning a feature space has been demonstrated to be effective in Atari games [16] and other exploration games

---

*Related work on deep generative models is presented in Section 2.8.

2

[2, 108, 26]. Differently, Houthooft *et al.* [60] presented an exploration strategy based on the maximization of information gain about the agent's belief of environment dynamics. Another common approach for exploration is that of using state visitation counts as intrinsic rewards [15, 139]. Recently, Raileanu *et al.* [114] proposed to jointly encourage both the visitation of novel states and the learning of the environment dynamics, by devising a paradigm that rewards the agent proportionally to the change in the state representation caused by its actions. However, their approach is developed for grid-like environments with a finite number of states, where the visitation count can be easily employed as a discount factor.

In Chapter 3 and Chapter 4, we present two intrinsic reward paradigms that are suited for robotic exploration in continuous photo-realistic environments.

2.3 Embodied Agents for Navigation

After having presented the literature on embodied exploration in Section 2.1, in this section we discuss relevant research work on navigation. The tasks of embodied exploration and navigation are very closely related in Embodied AI literature, in fact, several research papers on exploration methods also validate their approaches on the downstream task of Point Goal navigation (PointNav).

In the last few years, performance on PointNav has improved significantly saturating very quickly the task of noise-free navigation. For example, Wijmans *et al.* [147] achieved 95% of success rate (SR) on Habitat Challenge 2019 test dataset. However, moving on to noisy navigation the task becomes problematic.

Literature on Embodied AI methods for navigation can be divided in two main categories: map-based and map-less approaches.

Regarding the map-based approaches, Active Neural SLAM (ANS) [32] was among the first methods to adopt a hierarchical planning module coupled with a learning-based neural mapper. Ramakrishnan *et al.* [115] improved ANS results by inferring unseen or occluded regions in the scene.

Among the mapless methods, Sax *et al.* [127] transferred visual features from Taskonomy [156], showing that using diverse representation sets for downstream

tasks improves performance. Ye *et al.* [154, 153] takes inspiration from this approach, devising a method that involves learning auxiliary tasks to tackle both Point Goal navigation and Object Goal navigation. Instead, Karkus *et al.* [68] took inspiration from traditional Robotics proposing a learning-based particle filter to improve agents' pose estimation during navigation. Zhao *et al.* [161] is the first to apply visual odometry combined with a learning-based approach effectively. Recently, Partsey *et al.* [107] developed data-augmentation techniques that do not require human annotations to train models for visual odometry, achieving significant results on the task of PointNav.

We tackle the task of embodied navigation in Chapter 5, 7, and 8.

2.4 Interactive Environments and Datasets

When it comes to training intelligent agents, the underlying environment plays an important role. A first testbed for research in RL has been provided by the Atari games of the Arcade Learning Environment (ALE) [16, 25, 93]. Using these environments researchers were able to make remarkable progress on learning-based agents surpassing human performance on most games [59]. However, these kinds of settings are not suitable for navigation and exploration in general. To solve this problem, many maze-like environments have been proposed [71, 14], but nevertheless, agents trained in synthetic environments hardly adapt to real-world scenarios, because of the drastic change in terms of visual appearance.

Simulating platforms like Habitat [126] and Matterport3D simulator [9] provide a photo-realistic environment to train navigation agents. Some of these simulators only provide RGB equirectangular images as visual input [9], while others employ the full 3D model and implement physic interactions with the environment [149, 126, 137, 148, 82]. In these simulating platforms, algorithms for intelligent exploration and navigation can be developed safely and more quickly than in the real world, before being easily deployed on real robotic platforms [67, 8].

Alongside the introduction of these simulators, a large number of task-specific datasets for Embodied AI has been released fostering the research on multiple aspects of Embodied AI. The main tasks covered by these datasets are PointNav

2

and ObjectNav, VLN [9, 65, 76, 75, 112, 163], and household tasks [132, 45].

Among the other datasets, a relevant contribution is given by the release of photo-realistic indoor environments that contain semantic annotations of the objects or regions in the scene [30, 11, 151]. Such datasets enabled multiple tasks that involve learning semantic relations and characteristics of the environment.

We aim to enlarge the possibilities enabled by the presence of specific data for Embodied AI research by introducing two new datasets in Chapter 6 and Chapter 8.

2.5 Simulation-to-Reality Transfer

As mentioned in the previous sections, simulation allowed an impressive boost in training efficiency and final performance of embodied agents on a multitude of tasks. Some of these embodied tasks were solved, *i.e.* Wijmans *et al*. [147] achieved nearly perfect results on noise-free Point Goal navigation. Nevertheless, their model is trained using 2.5 billion frames and requires experience acquired over more than half a year of GPU time, and unfortunately, models tend to learn simulator-specific tricks to circumvent navigation difficulties [67]. Since such shortcuts do not work in the real world, there is a significant simulation-to-reality performance gap.

Recent work has studied how to deploy models trained on simulation to the real world [44, 67, 123]. In their work, Kadian *et al*. [67] make a 3D acquisition of a real-world scene and study the Sim2Real gap for various setups and metrics. However, their environment is very simple as the obstacles are large boxes, the floor has an even and regular surface in order to facilitate the actuation system, and there are no doors or other navigation bottlenecks. Another recent study on Sim2Real transfer is done by Truong *et al*. [143] employing complex legged robots. Gervet *et al*. [49], instead, extensively study the simulation-to-reality transfer of agents for visual semantic navigation in a real-world apartment.

In Chapter 7, we present some guidelines for the deployment of models that are trained in simulation to real robotic platforms and we test our method in a realistic environment with multiple rooms and typical office furniture.

2.6   ENVIRONMENT KNOWLEDGE IN EMBODIED AGENTS

Current research on Embodied AI for navigation agents can be categorized according to the quantity of knowledge about the environment provided to the agent prior to performing the task [5]. The most common approaches focus on the scenario in which the agent is deployed in a completely new environment for which it has no prior knowledge [54, 32, 68, 147].

In a different but similar approach, the agent has no knowledge of the environment, but the exploration is run in parallel with a target-driven navigation task resulting in an effective approach to solving the latter (*e.g.*, ObjectNav [33] and PointNav [115]).

Other approaches consider the case in which the agent is able to exploit preacquired information about the environment [43, 34] when performing a navigation task. Such preacquired information can be either partial [125, 134, 160] or complete [38, 28, 118]. A major limitation of such approaches is that the obtained map representation is assumed to conform perfectly with the environment where the downstream task will be performed.

A last option is the case in which the preacquired map provided to the agent is incomplete or incorrect due to changes that occurred in the environment over time. Common strategies to deal with changing environments entail disregarding dynamic objects as landmarks when performing simultaneous localization and mapping (SLAM) [124, 23] and applying local policies to avoid them when navigating [92]. An alternative strategy is learning to predict geometric changes based on experience, as done by Nardi and Stachniss [98], where the environment is represented as a traversability graph. The main limitation of this strategy is its computational intractability when considering dense metric maps of wide areas.

In Chapter 6 we present a learning-based method to efficiently update an obsolete map of the environment when using occupancy grids.

2.7 Image Captioning

In order to enable the capability of autonomous agents to provide user-understandable representations of the perceived environment [111], in the form of natural language descriptions, Image Captionings methods need to be implemented on the robot. Image Captioning is a task at the intersection between vision and language whose goal is to generate a natural language description of a given image. To represent images, convolutional neural network (CNN) can be used to extract global features [69, 121, 146], grids of features [150, 89], or features for image regions containing visual entities [7, 90]. In most cases, attention mechanisms are applied to enhance the visual input representation. More recent approaches employ fully-attentive Transformer-like architectures [144] as visual encoders, which can also be applied directly to image patches [87, 41, 13]. The image representation is used to condition a language model that generates the caption. The language model can be implemented as a recurrent neural network [69, 121, 62, 79] or Transformer-based fully-attentive models [42, 159].

We adopt captioning models in Chapter 4 and Chapter 5 to generate descriptions from the point of view of the embodied agent.

2.8 Deep Generative Models

Deep generative models are trained to approximate high-dimensional probability distributions by means of a large set of training samples. In recent years, literature on deep generative models followed three main approaches: latent variable models like VAE [73], implicit generative models like GANs or the more recent Style-GAN and VQ-GAN [51, 70, 47], and exact likelihood models. Exact likelihood models can be classified in non-autoregressive flow-based models, like RealNVP [46] and Flow++ [57], and autoregressive models, like PixelCNN [100] and Image Transformer [106]. Non-autoregressive flow-based models consist of a sequence of invertible transformation functions to compose a complex distribution modeling the training data. Autoregressive models decompose the joint distribution of images as a product of conditional probabilities of the single pix-

els. Usually, each pixel is computed using as input only the previously predicted ones, following a raster scan order.

Regarding the work presented in this dissertation, we use a generative model in the architecture presented in Chapter 3.

2

2

3

Focus on Impact:
Embodied Exploration with Intrinsic Motivation

R OBOTIC exploration can be defined as the task of autonomously navigating an unknown environment with the goal of gathering sufficient information to represent it, often via a spatial map [136]. This ability is key to enabling many downstream tasks such as planning [130] and goal-driven navigation [126, 147, 96]. Although a vast portion of existing literature tackles this problem [102, 32, 38, 115], it is not yet completely solved, especially in complex indoor environments. The introduction of large datasets of photo-realistic indoor environments [30, 149] has eased the development of robust exploration strategies[119], which can be validated safely and quickly thanks to powerful simulating platforms [44, 126]. Moreover, exploration algorithms developed on simulated environments can be deployed in the real world with little hyperparameter tuning [67, 142], if the simulation is sufficiently realistic.

Most of the recently devised exploration algorithms exploit Deep Reinforce-

---

This Chapter is related to the publication "R. Bigazzi *et al*., Focus on Impact: Indoor Exploration with Intrinsic Motivation, RA-L 2022" [18]. See the list of Publications on page 133 for more details.

3
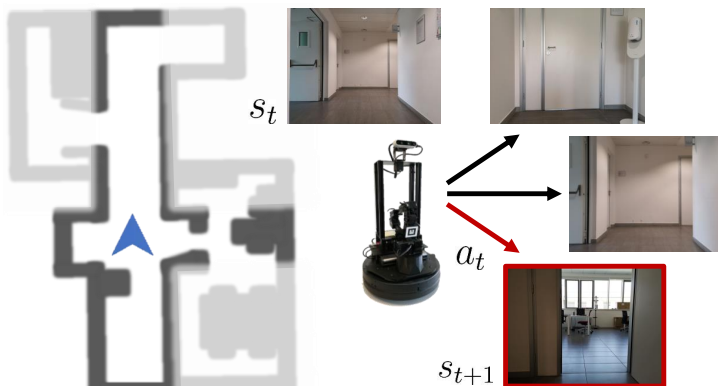
Figure 3.1: We propose an impact-based reward for robot exploration of continuous indoor spaces. The robot is encouraged to take actions that maximize the difference between two consecutive observations.

ment Learning (DRL) [162], as learning-based exploration and navigation algorithms are more flexible and robust to noise than geometric methods [38, 99, 118]. Despite these advantages, one of the main challenges in training DRL-based exploration algorithms is designing appropriate rewards.

In this work, we propose a new reward function that employs the impact of the agent actions on the environment, measured as the difference between two consecutive observations [114], discounted with a pseudo-count [15] for previously-visited states (see Fig 3.1). So far, impact-based rewards [114] have been used only as an additional intrinsic reward in procedurally-generated (*e.g.* grid-like mazes) or singleton (*i.e.* the test environment is the same employed for training) synthetic environments. Instead, our reward can deal with photo-realistic non-singleton environments.

Recent research on robot exploration proposes the use of an extrinsic reward based on the prediction of the occupancy of the surrounding environment [115]. This type of reward encourages the agent to navigate toward areas that can be easily mapped without errors. Unfortunately, this approach presents a major drawback, as this reward heavily depends on the mapping phase, rather than focusing on what has been already seen. In fact, moving towards new places that are difficult to map would produce a low occupancy-based reward. Moreover,

the precise layout of the training environments is not always available, especially in real-world applications. To overcome these issues, a possible solution is given by the use of intrinsic reward functions, so that the agent does not need to rely on annotated data and can compute its reward by means of its observations. Some examples of recently proposed intrinsic rewards for robot exploration are based on curiosity [20], novelty [118], and coverage [32]. All these rewards, however, tend to vanish with the length of the episode because the agent quickly learns to model the environment dynamics and appearance (for curiosity and novelty-based rewards) or tends to stay in previously-explored areas (for the coverage reward). Impact, instead, provides a stable reward signal throughout the episode [114].

Since robot exploration takes place in complex and realistic environments that can present an infinite number of states, it is impossible to store a visitation count for every state. Furthermore, the vector of visitation counts would consist of a very sparse vector, and that would cause the agent to give the same impact score to nearly identical states. To overcome this issue, we introduce an additional module in our design to keep track of a pseudo-count for visited states. The pseudo-count is estimated by a density model that is trained end-to-end and together with the policy. We integrate our newly-proposed reward in a modular embodied exploration and navigation system inspired by that proposed by Chaplot *et al*. [32] and consider two commonly adopted collections of photo-realistic simulated indoor environments, namely Gibson [149] and Matterport 3D (MP3D) [30]. Furthermore, we also deploy the devised algorithm in the real world. The results in both simulated and real environments are promising: we outperform state-of-the-art baselines in simulated experiments and demonstrate the effectiveness of our approach in real-world experiments. We make the source code of our approach and pretrained models publicly available*.

---

*https://github.com/aimagelab/focus-on-impact
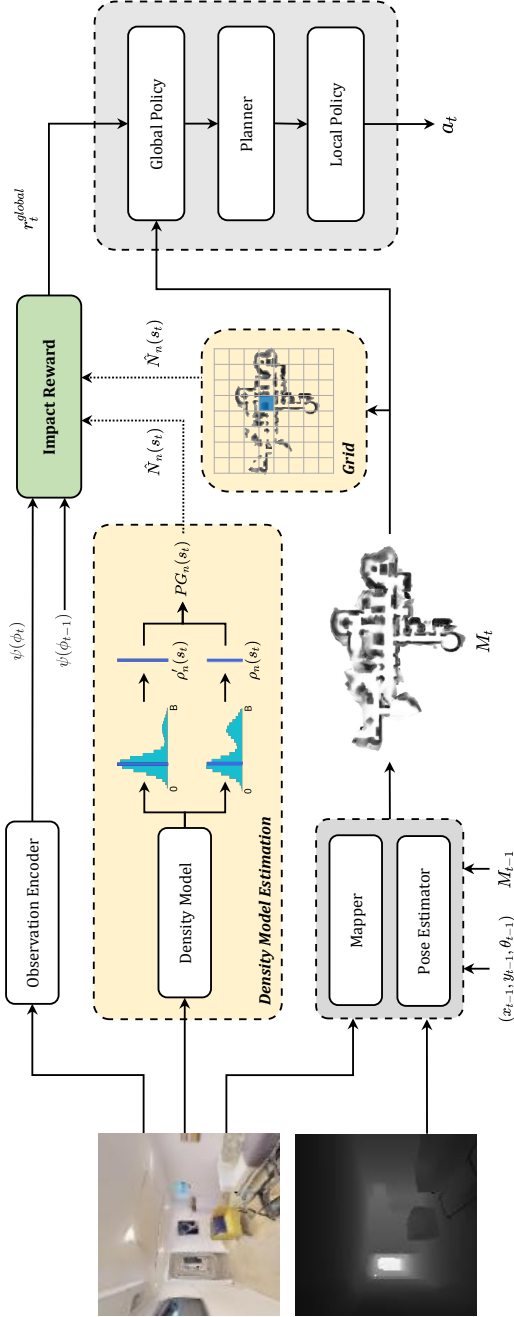
Figure 3.2: Our modular exploration architecture consists of a Mapper that iteratively builds a top-down occupancy map of the environment, a Pose Estimator that predicts the pose of the robot at every step, and a hierarchical self-supervised Navigation Module in charge of sequentially setting exploration goals and predicting actions to navigate towards it. We exploit the impact-based reward to guide the exploration and adapt it for continuous environments, using an Observation Encoder to extract observation features and depending on the method, a Density Model or a Grid to compute the pseudo-count.

3.1 Proposed Method

Following the current state-of-the-art architectures for navigation for embodied agents [32, 115], the proposed method comprises three major components: a CNN-based mapper, a pose estimator, and a hierarchical navigation policy. The navigation policy defines the actions of the agent, the mapper builds a top-down map of the environment to be used for navigation, and the pose estimator locates the position of the agent on the map. Our architecture is depicted in Fig. 3.2 and described below.

3.1.1 Mapper

The mapper generates a map of the free and occupied regions of the environment discovered during the exploration. At each timestep, the RGB observation φ_t^{rgb} and the depth observation φ_t^d are processed to output a two-channel $V \times V$ local map m_t depicting the area in front of the agent, where each cell describes the state of a 5×5 cm area of the environment, the channels measure the probability of a cell being occupied and being explored, as in [32].

Specifically, the RGB observation φ_t^{rgb} is encoded using the first two blocks of ResNet-18 [55] pretrained on ImageNet, followed by a three-layered CNN. We project the depth image φ_t^d using the camera intrinsics [38] and obtain a preliminary map for the visible occupancy. We name the obtained feature representations $\hat{\varphi}_t^{rgb}$ and $\hat{\varphi}_t^d$, respectively. We then encode the two feature maps using a U-Net [122]:

$$f_\mu(\hat{\varphi}_t^{rgb}, \hat{\varphi}_t^d) = \text{U-Net}_{\text{enc}}(\hat{\varphi}_t^{rgb}, \hat{\varphi}_t^d, \mu_{\text{enc}}), \tag{3.1}$$

and decode the $2 \times V \times V$ matrix of probabilities as:

$$v_t = \sigma(\text{U-Net}_{\text{dec}}(f_\mu(\hat{\varphi}_t^{rgb}, \hat{\varphi}_t^d), \mu_{\text{dec}})), \tag{3.2}$$

where μ_{enc} and μ_{dec} represent the learnable parameters in the U-Net encoder and decoder, respectively, and σ is the sigmoid activation function.

Please note that this module performs anticipation-based mapping, as defined in [115], where the predicted local map m_t includes also unseen/occluded por-

tions of space. The local maps are aggregated and registered to the $W \times W \times 2$ global map M_t of the environment. To that end, we use a geometric transformation to project v_t in the global coordinate system, for which we need a triple (x, y, θ) corresponding to the agent position and heading in the environment. This triple is estimated by a specific component that tracks the agent displacements across the environment, as discussed in the following paragraph. The resulting global map is used by the navigation policy for action planning and in this setting, it is initially empty and is built incrementally with the exploration of the environment.

3.1.2 POSE ESTIMATOR

The pose estimator is used to predict the displacement of the agent as a consequence of an action. The considered atomic actions a_t of the agent are: *go forward 0.25m, turn left 10°, turn right 10°*. However, the noise in the actuation system and the possible physical interactions between the agent and the environment could produce unexpected outcomes causing positioning errors. The pose estimator reduces the effect of such errors by predicting the real displacement $(\Delta x_t, \Delta y_t, \Delta \theta_t)$. According to [115], the input of this module consists of the RGB-D observations $(\varphi_{t-1}^{rgb}, \varphi_t^{rgb})$ and $(\varphi_{t-1}^d, \varphi_t^d)$ and the local maps (m_{t-1}, m_t). Each modality $i = \{0, 1, 2\}$ is encoded singularly to obtain three different estimates of the displacement:

$$p_i(e_{t-1}, e_t) = W_1 \max(W_2(e_{t-1}, e_t) + b_2, 0) + b_1, \quad (3.3)$$

where $e_t \in \{\varphi_t^{rgb}, \varphi_t^d, m_t\}$ and $W_{1,2}$ and b_2 are weights matrices and bias. Eventually, the displacement estimates are aggregated with a weighted sum:

$$\alpha_i = \text{softmax}(\text{MLP}_i([\bar{\varphi}_t^{rgb}, \bar{\varphi}_t^d, \bar{m}_t])), \quad (3.4)$$

$$(\Delta x_t, \Delta y_t, \Delta \theta_t) = \sum_{i=0}^{2} \alpha_i \cdot p_i, \quad (3.5)$$

where multilayer perceptron (MLP) is a three-layered fully-connected network, $(\bar{\varphi}_t^{rgb}, \bar{\varphi}_t^{d}, \bar{l}_t)$ are the inputs encoded by a CNN, and $[\cdot, \cdot, \cdot]$ denotes tensor concatenation. The estimated pose of the agent at time t is given by:

$$(x_t, y_t, \theta_t) = (x_{t-1}, y_{t-1}, \theta_{t-1}) + (\Delta x_t, \Delta y_t, \Delta \theta_t). \qquad (3.6)$$

Note that, at the beginning of each exploration episode, the agent sets its position to the center of its environment representation, *i.e.* $(x_0, y_0, \theta_0) = (0, 0, 0)$.

3.1.3 NAVIGATION POLICY

The sampling of the atomic actions of the agent relies on the hierarchical navigation policy that is composed of the following modules: the global policy, the planner, and the local policy. This architecture is in line with current literature on embodied exploration [32, 115, 116]. The hierarchical policy is adopted to decouple high-level and low-level concepts like moving across rooms and avoiding obstacles. It samples a goal coordinate on the map, while the deterministic planner uses the global goal to compute a local goal in close proximity of the agent. The local policy then predicts actions to reach the local goal.

The global policy samples a point on an augmented global map of the environment, M_t^+, that represents the current global goal of the agent. The augmented global map M_t^+ is a $W \times W \times 4$ map obtained by stacking the two-channel global map M_t from the Mapper with the one-hot representation of the agent position (x_t, y_t) and the map of the visited positions, which collects the one-hot representations of all the positions assumed by the agent from the beginning of the exploration. Moreover, M_t^+ is in parallel cropped with respect to the position of the agent and max-pooled to a spatial dimension $H \times H$ where $H < W$. These two versions of the augmented global map are concatenated to form the $H \times H \times 8$ input of the global policy that is used to sample a goal in the global action space $H \times H$. The global policy is trained with reinforcement learning with our proposed impact-based reward r_t^{global}, defined below, that encourages exploration.

The deterministic planner adopts the A* algorithm to compute a feasible tra-

jectory from the agent's current position to the global goal using the current state of the map m_t. A point on the trajectory within 1.25m from the agent is extracted to form the local goal l_t.

The local policy outputs the atomic actions needed to reach the local goal and is trained to minimize the euclidean distance to the local goal, which is expressed via the following reward:

$$r_t^{local} = d(\omega_{t-1}, l_t, M_{t-1}) - d(\omega_t, l_t, M_t), \qquad (3.7)$$

where $d(\cdot, \cdot, \cdot)$ is the function computing the euclidean distance between two positions using the map M_t to take into account possible obstacles on the way, while $\omega = (x_t, y_t, \theta_t)$ and l_t are respectively, the pose of the agent and the local goal at timestep t. Note that the output actions in our setup are discrete. These platform-agnostic actions can be translated into signals for specific robot actuators, as we do in this work. Alternatively, based on the high-level predicted commands, continuous actions can be predicted, *e.g.* in the form of linear and angular velocity commands to the robot, by using an additional, lower-level policy, as done in [64]. The implementation of such a policy is beyond the scope of our work.

Following the hierarchical structure, the global goal is reset every η steps, and the local goal is reset if at least one of the following conditions verifies: a new global goal is sampled, the agent reaches the local goal, the local goal location is discovered to be in an occupied area.

3.1.4 Impact-Driven Exploration

The exploration ability of the agent relies on the design of an appropriate reward for the global policy. In this setting, the lack of external rewards from the environment requires the design of a dense intrinsic reward. To the best of our knowledge, our proposed method presents the first implementation of impact-driven exploration in photo-realistic environments. The key idea of this concept is encouraging the agent to perform actions that have impact on the environment and the observations retrieved from it, where the impact at timestep t is measured

as the l_2-norm of the encodings of two consecutive states $\psi(s_t)$ and $\psi(s_{t+1})$, considering the RGB observation ϕ_t^{rgb} to compute the state s_t. Following the formulation proposed in [114], the reward of the global policy for the proposed method is calculated as:

$$r_t^{global}(s_t, s_{t+1}) = \frac{\|\psi(s_{t+1}) - \psi(s_t)\|_2}{\sqrt{N(s_{t+1})}}, \tag{3.8}$$

where $N(s_t)$ is the visitation count of the state at timestep t, *i.e.* how many times the agent has observed s_t. The visitation count is used to drive the agent out of regions already seen in order to avoid trajectory cycles. Note that the visitation count is episodic, *i.e.* $N_{ep}(s_t) \equiv N(s_t)$. For simplicity, in the following, we denote the episodic visitation count as $N(s_t)$.

VISITATION COUNTS. The concept of normalizing the reward using visitation count, as in [114], fails when the environment is continuous since during exploration is unlikely to visit exactly the same state more than once. In fact, even microscopic changes in terms of translation or orientation of the agent cause shifts in the values of the RGB observation, thus resulting in new states. Therefore, using a photo-realistic continuous environment nullifies the scaling property of the denominator of the global reward in Eq. 3.8 because every state s_t during the exploration episode is, most of the times, only encountered for the first time. To overcome this limitation, we implement two types of pseudo-visitation counts $\hat{N}(s_t)$ to be used in place of $N(s_t)$, which extend the properties of visitation counts to continuous environments: *Grid* and *Density Model Estimation*.

GRID. With this approach, we consider a virtual discretized grid of cells with fixed size in the environment. We then assign a visitation count to each cell of the grid. Note that, different from approaches working on procedurally-generated environments like [114], the state space of the environment we consider is continuous also in this formulation, and depends on the pose of the agent (x, y, θ). The grid approach operates a quantization of the agent's positions, and that allows to cluster observation made from similar positions. To this end, we take the global map of the environment and divide it into cells of size $G \times G$. The estimated pose of the agent, regardless of its orientation θ_t, is used to select the cell that the agent occupies at time t. In the *Grid* formulation, the visitation count of the selected

cell is used as $N(s_t)$ in Eq. 3.8 and is formalized as:

$$\hat{N}(s_t) = \hat{N}(\text{grid}(x_t, y_t)), \tag{3.9}$$

where $\text{grid}(\cdot)$ returns the block corresponding to the estimated position of the agent.

3

Density Model Estimation (Density Model Estimation (DME)). Let ρ be an autoregressive density model defined over the states $s \in S$, where S is the set of all possible states. We call $\rho_n(s)$ the probability assigned by ρ to the state s after being trained on a sequence of states $s_1, \ldots, s_n$, and $\rho'_n(s)$, or recoding probability [15, 103], the probability assigned by ρ to s after being trained on $s_1, \ldots, s_n, s$. The prediction gain PG of ρ describes how much the model has improved in the prediction of s after being trained on s itself, and is defined as

$$PG_n(s) = \log \rho'_n(s) - \log \rho_n(s). \tag{3.10}$$

In this work, we employ a lightweight version of Gated PixelCNN [100] as density model. This model is trained from scratch along with the exploration policy using the states visited during the exploration, which are fed to PixelCNN one at a time, as they are encountered. The weights of PixelCNN are optimized continually over all the environments. As a consequence, the knowledge of the density model is not specific for a particular environment or episode. To compute the input of the PixelCNN model, we transform the RGB observation φ_t^{rgb} to grayscale and we crop and resize it to a lower size $P \times P$. The transformed observation is quantized to B bins to form the final input to the model, s_t. The model is trained to predict the conditional probabilities of the pixels in the transformed input image, with each pixel depending only on the previous ones following a raster scan order. The output has shape $P \times P \times B$ and each of its elements represents the probability of a pixel belonging to each of the B bins. The joint distribution of

the input modeled by PixelCNN is:

$$p(s_t) = \prod_1^{P^2} p(\chi_i | \chi_1, ..., \chi_{i-1}), \tag{3.11}$$

where χ_i is the ith pixel of the image s_t. ρ is trained to fit $p(s_t)$ by using the negative log-likelihood loss.

Let $\hat{n}$ be the pseudo-count total, *i.e.* the sum of all the visitation counts of all states during the episode. The probability and the recoding probability of s can be defined as:

$$\rho_n(s) = \frac{\hat{N}_n(s)}{\hat{n}}, \qquad\qquad \rho_n'(s) = \frac{\hat{N}_n(s) + 1}{\hat{n} + 1}. \tag{3.12}$$

Note that, if ρ is learning-positive, *i.e.* if $PG_n(s) > 0$ for all possible sequences $s_1, ..., s_n$ and all $s \in S$, we can approximate $\hat{N}_n(s)$ as:

$$\hat{N}_n(s) = \frac{\rho_n(s)(1 - \rho_n'(s))}{\rho_n'(s) - \rho_n(s)} \approx (e^{PG_n(s)} - 1)^{-1}. \tag{3.13}$$

To use this approximation in Eq. 3.8, we still need to address three problems: it does not scale with the length of the episode, the density model could be not learning-positive, and $\hat{N}_n(s)$ should be large enough to avoid the reward becoming too large regardless the goal selection. In this respect, to take into account the length of the episode, we introduce a normalizing factor $n^{-1/2}$, where n is the number of steps done by the agent since the start of the episode. Moreover, to force ρ to be learning-positive, we clip $PG_n(s)$ to 0 when it becomes negative. Finally, to avoid small values at the denominator of r_t^{global} (Eq. 3.8), we introduce a lower bound of 1 to the pseudo visitation count. The resulting definition of $\hat{N}_n(s)$ in the *Density Model Estimation* formulation is:

$$\widetilde{PG}_n = c \cdot n^{-1/2} \cdot (PG_n(s))_+, \tag{3.14}$$

$$\hat{N}_n(s) = \max\left\{ \left(e^{\widetilde{PG}_n(s)} - 1\right)^{-1}, 1 \right\}, \tag{3.15}$$

where c is a term used to scale the prediction gain. It is worth noting that, unlike the Grid approach that can be applied only when s_t is representable as the robot location, the DME can be adapted to a wider range of tasks, including settings where the agent alters the environment.

3.2 Experimental Setup

3.2.1 Datasets

For comparison with state-of-the-art DRL-based methods for embodied exploration, we employ the photo-realistic simulated 3D environments contained in the Gibson dataset [149] and the MP3D dataset [30]. Both these datasets consist of indoor environments where different exploration episodes take place. In each episode, the robot starts exploring from a different point in the environment. Environments used during training do not appear in the validation/test split of these datasets. Gibson contains 106 scans of different indoor locations, for a total of around 5M exploration episodes (14 locations are used in 994 episodes for test in the so-called Gibson Val split). MP3D consists of 90 scans of large indoor environments (11 of those are used in 495 episodes for the validation split and 18 in 1008 episodes for the test split).

3.2.2 Evaluation Protocol

We train our models on the Gibson train split. Then, we perform model selection basing on the results obtained on Gibson Val. We then employ the MP3D validation and test splits to benchmark the generalization abilities of the agents. To evaluate exploration agents, we employ the following metrics. Intersection-over-union (IoU) between the reconstructed map and the ground-truth map of the environment: here we consider two different classes for every pixel in the map (free or occupied). Similarly, the map accuracy (Acc, expressed in m^2) is the portion of the map that has been correctly mapped by the agent. The area seen (AS, in m^2) is the total area of the environment observed by the agent. For both the IoU and the area seen, we also present the results relative to the two different

Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Grid									
G = 2	0.726	0.721	0.730	51.41	61.88	34.17	27.71	0.240	4.450
G = 4	0.796	0.792	0.801	54.34	61.17	33.74	27.42	0.079	1.055
G = 5	**0.806**	**0.801**	**0.813**	**55.21**	**62.17**	**34.31**	**27.87**	**0.077**	**0.881**
G = 10	0.789	0.784	0.794	54.26	61.67	34.06	27.61	0.111	1.434
DME									
B = 64	0.773	0.768	0.778	53.58	61.00	33.79	27.21	0.131	2.501
B = 128	**0.796**	**0.794**	**0.799**	**54.73**	**62.07**	**34.27**	**27.79**	**0.095**	**1.184**
B = 256	0.685	0.676	0.695	49.27	61.40	33.95	27.45	0.311	6.817

Table 3.1: Results for our model selection on Gibson validation split for $T = 500$. $G = 5$ and $B = 128$ are the best models for the Grid-based and the DME-based models, respectively.

classes: free space and occupied space respectively (FIoU, OIoU, FAS, OAS). Finally, we report the mean positioning error achieved by the agent at the end of the episode. A larger translation error (TE, expressed in m) or angular error (AE, in degrees) indicates that the agent struggles to keep a correct estimate of its position throughout the episode. For all the metrics, we consider episodes of length $T = 500$ and $T = 1000$ steps.

For our comparisons, we consider five baselines trained with different rewards. *Curiosity* employs a surprisal-based intrinsic reward as defined in [108]. *Coverage* and *Anticipation* are trained with the corresponding coverage-based and accuracy-based rewards defined in [115]. For completeness, we include two count-based baselines, obtained using the reward defined in Eq. 3.8, but ignoring the contribution of impact (*i.e.* setting the numerator to a constant value of 1). These are *Count (Grid)* and *Count (DME)*. All the baselines share the same overall architecture and training setup of our main models.

3.2.3 Implementation Details

The experiments are performed using the Habitat Simulator [126] with observations of the agent set to be 128×128 RGB-D images and episode length during training set to $T = 500$. Each model is trained with the training split of the Gibson dataset [149] with 40 environments in parallel for $\approx$ 5M frames.

3

	Gibson Val (T = 500)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.678	0.669	0.688	49.35	61.67	34.16	27.51	0.330	7.430
Coverage	0.721	0.715	0.726	51.47	61.13	34.07	27.06	0.272	5.508
Anticipation	0.783	0.778	0.789	54.68	60.96	34.15	26.81	0.100	1.112
Count (Grid)	0.714	0.706	0.721	50.85	61.61	34.17	27.44	0.258	5.476
Count (DME)	0.764	0.757	0.772	52.81	60.69	33.68	27.01	0.148	2.888
Impact (Grid)	**0.803**	**0.797**	**0.809**	54.94	61.90	34.07	27.83	**0.079**	**0.878**
Impact (DME)	0.800	0.796	0.803	**55.10**	**62.59**	**34.45**	**28.14**	0.095	1.166

	MP3D Val (T = 500)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.339	0.473	0.205	97.82	118.13	75.73	42.40	0.566	7.290
Coverage	0.352	0.494	0.210	102.05	120.00	76.78	43.21	0.504	5.822
Anticipation	0.381	0.530	0.231	106.02	114.06	72.94	41.13	0.151	1.280
Count (Grid)	0.347	0.488	0.206	99.00	116.77	75.00	41.76	0.466	5.828
Count (DME)	0.359	0.493	0.225	101.73	112.65	72.22	40.43	0.268	3.318
Impact (Grid)	0.383	0.531	**0.234**	107.41	116.60	74.44	42.17	**0.120**	**0.860**
Impact (DME)	**0.396**	**0.560**	0.233	**111.61**	**124.06**	**79.47**	**44.59**	0.232	1.988

	MP3D Test (T = 500)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.362	0.372	0.352	109.66	130.48	85.98	44.50	0.620	7.482
Coverage	0.390	0.401	0.379	116.71	134.89	88.15	46.75	0.564	5.938
Anticipation	0.424	0.433	**0.415**	117.87	124.24	81.31	42.93	0.151	1.306
Count (Grid)	0.364	0.381	0.348	117.50	134.85	89.81	45.05	0.525	5.790
Count (DME)	0.391	0.397	0.385	114.02	123.86	81.86	42.00	0.287	3.322
Impact (Grid)	0.420	0.430	0.409	124.44	130.98	86.08	44.90	**0.124**	**0.834**
Impact (DME)	**0.426**	**0.444**	0.409	**133.51**	**144.64**	**95.70**	**48.94**	0.288	2.312

Table 3.2: Exploration results on Gibson and MP3D datasets, at $T = 500$ timesteps. On the small environments of Gibson Val, Impact (Grid) is the best-performing model, while on larger environments like those of MP3D, Impact (DME) achieves the best results.

NAVIGATION MODULE. The reinforcement learning algorithm used to train the global and local policies is PPO [129] with Adam optimizer and a learning rate of 2.5×10^{-4}. The global goal is reset every $\eta = 25$ timesteps and the global action space hyperparameter H is 240. The local policy is updated every η steps and the global policy is updated every 20η steps.

	Gibson Val (T = 1000)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.560	0.539	0.581	45.71	67.64	37.19	30.45	0.682	14.862
Coverage	0.653	0.641	0.664	50.10	66.15	36.77	29.38	0.492	10.796
Anticipation	0.773	0.763	0.782	56.37	66.61	37.17	29.44	0.155	1.876
Count (Grid)	0.608	0.592	0.624	48.22	67.80	37.31	30.50	0.520	10.996
Count (DME)	0.708	0.694	0.722	52.67	66.91	36.81	30.12	0.282	5.802
Impact (Grid)	**0.802**	**0.793**	**0.811**	**57.21**	67.74	37.04	30.69	**0.119**	**1.358**
Impact (DME)	0.789	0.783	0.796	56.77	**68.34**	**37.42**	**30.92**	0.154	1.958

	MP3D Val (T = 1000)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.336	0.449	0.223	109.79	157.27	100.07	57.20	1.322	14.540
Coverage	0.362	0.492	0.232	116.58	158.83	100.76	58.07	1.072	11.624
Anticipation	0.420	0.568	0.272	126.86	147.33	93.56	53.78	0.267	2.436
Count (Grid)	0.350	0.474	0.226	112.75	157.13	100.03	57.10	1.074	11.686
Count (DME)	0.379	0.505	0.254	119.07	149.62	95.16	54.46	0.590	6.544
Impact (Grid)	**0.440**	**0.595**	**0.285**	**133.97**	157.19	99.61	57.58	**0.202**	**1.294**
Impact (DME)	0.427	0.587	0.268	133.27	**166.20**	**105.69**	**60.50**	0.461	3.654

	MP3D Test (T = 1000)								
Model	IoU ↑	FIoU ↑	OIoU ↑	Acc ↑	AS ↑	FAS ↑	OAS ↑	TE ↓	AE ↓
Curiosity	0.361	0.365	0.357	130.10	185.36	121.65	63.71	1.520	14.992
Coverage	0.409	0.418	0.399	142.86	193.20	126.21	66.99	1.240	11.814
Anticipation	0.484	0.491	0.478	153.83	174.76	114.29	60.47	0.289	2.356
Count (Grid)	0.377	0.391	0.363	144.26	194.76	129.22	65.53	1.246	11.608
Count (DME)	0.418	0.419	0.418	140.21	172.44	113.25	59.19	0.657	6.572
Impact (Grid)	**0.502**	**0.510**	**0.494**	168.55	190.03	124.44	65.60	**0.218**	**1.270**
Impact (DME)	0.481	0.498	0.464	**174.18**	**212.00**	**140.10**	**71.90**	0.637	4.390

Table 3.3: Exploration results on Gibson and MP3D datasets, at $T = 1000$ timesteps. Impact (Grid) is the best-performing model with respect to IoU, while Impact (DME) achieves the best results on AS.

MAPPER AND POSE ESTIMATOR. These models are trained with a learning rate of 10^{-3} with Adam optimizer, the local map size is set with $V = 101$ while the global map size is $W = 961$ for episodes in the Gibson dataset and $W = 2001$ in the MP3D dataset. Both models are updated every 4η timesteps, where η is the reset interval of the global policy.

DENSITY MODEL. The model used for density estimation is a lightweight version of Gated PixelCNN [100] consisting of a 7×7 masked convolution followed by two residual blocks with 1×1 masked convolutions with 16 output channels, a 1×1 masked convolutional layer with 16 output channels, and a final 1×1 masked convolution that returns the output logits with shape $P \times P \times B$, where B is the number of bins used to quantize the model input. We set $P = 42$ for the resolution of the input and the output of the density model, and $c = 0.1$ for the prediction gain scale factor.

3.3 EXPERIMENTAL RESULTS

3.3.1 EXPLORATION RESULTS

As a first step, we perform model selection using the results on the Gibson Val split (Table 3.1). Our agents have different hyperparameters that depend on the implementation for the pseudo-counts. When our model employs grid-based pseudo-counts, it is important to determine the dimension of a single cell in this grid-based structure. In our experiments, we test the effects of using $G \times G$ squared cells, with $G \in \{2, 4, 5, 10\}$. The best results are obtained with $G = 5$, with small differences among the various setups. When using pseudo-counts based on a density model, the most relevant hyperparameters depend on the particular model employed as density estimator. In our case, we need to determine the number of bins B for PixelCNN, with $B \in \{64, 128, 256\}$. We find out that the best results are achieved with $B = 128$.

In Table 3.2 and 3.3, we compare the Impact (Grid) and Impact (DME) agents with the baseline agents previously described on the considered datasets. For each model and each split, we test 5 different random seeds and report the mean result for each metric. For the sake of readability, we do not report the standard deviations for the different runs, which we quantify in around 1.2% of the mean value reported.

As can be seen, results achieved by the two proposed impact-based agents are
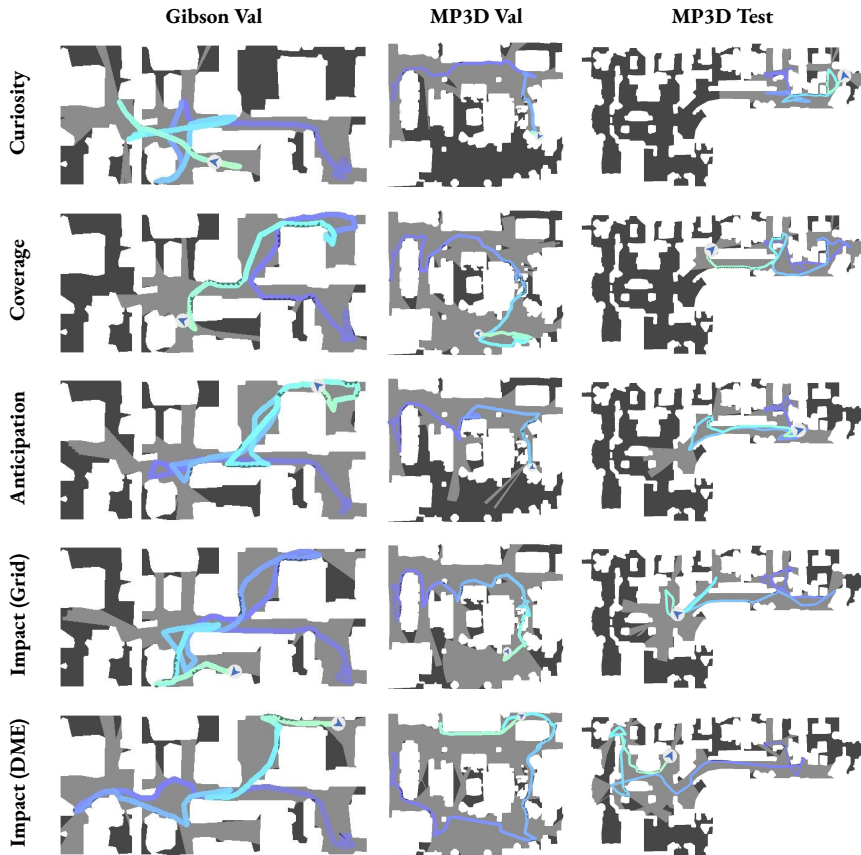
Figure 3.3: Qualitative results. For each model, we report three exploration episodes on Gibson and MP3D datasets for $T = 500$. The exploration capabilities of the Impact-based models are higher than the baselines, in particular in larger environments.

constantly better than those obtained by the competitors, both for $T = 500$ and $T = 1000$. It is worth noting that our intrinsic impact-based reward outperforms strong extrinsic rewards that exploit information computed using the ground-truth layout of the environment. Moreover, the different implementations chosen for the pseudo-counts affect final performance, with Impact (DME) bringing the best results in terms of AS and Impact (Grid) in terms of IoU metrics. From the results, it also emerges that, although the proposed implementations for the pseudo-count in Eq. 3.8 lead to comparable results in small envi-

ronments as those contained in Gibson and MP3D Val, the advantage of using DME is more evident in large, complex environments as those in MP3D Test.

In Fig. 3.3, we report some qualitative results displaying the trajectories and the area seen by different agents in the same episode. Also from a qualitative point of view, the benefits given by the proposed reward in terms of exploration trajectories and explored areas are easy to identify.

3.3.2 Point Goal Navigation Results

One of the main advantages of training deep modular agents for embodied exploration is that they easily adapt to perform downstream tasks, such as Point Goal navigation [126]. Recent literature [32, 115] has discovered that hierarchical agents trained for exploration are competitive with state-of-the-art architecture tailored for PointGoal navigation and trained with strong supervision for 2.5 billion frames [147]. Additionally, the training time and data required to learn the policy are much more limited (2 to 3 orders of magnitude smaller). In Table 3.4, we report the results obtained using two different settings. The *noise-free pose sensor* setting is the standard benchmark for Point Goal navigation in Habitat [126]. In the *noisy pose sensor* setting, instead, the pose sensor readings are noisy, and thus the agent position must be estimated as the episode progresses. We consider four main metrics: the average distance to the goal achieved by the agent (D2G) and three success-related metrics. The success rate (SR) is the fraction of episodes terminated within 0.2 meters from the goal, while the SPL and SoftSPL (sSPL) weigh the distance from the goal with the length of the path taken by the agent in order to penalize inefficient navigation. As can be seen, the two proposed agents outperform the main competitors from the literature: Active Neural SLAM (ANS) [32] and OccAnt [115] (for which we report both the results from the paper and the official code release).

When comparing with our baselines in the noise-free setting, the overall architecture design allows for high-performance results, as the reward influences map estimation only marginally. In fact, in this setting, the global policy and the pose

---

[†]https://github.com/facebookresearch/OccupancyAnticipation

Model	Noise-free Pose Sensor				Noisy Pose Sensor			
	D2G ↓	SR ↑	SPL ↑	sSPL ↑	D2G ↓	SR ↑	SPL ↑	sSPL
ANS [32]	-	0.950	0.846	-	-	-	-	-
OccAnt [115]	-	0.930	0.800	-	-	-	-	-
OccAnt [115]†	-	-	<u>0.911</u>	-	-	-	-	-
Curiosity	**0.238**	**0.970**	**0.914**	**0.899**	0.302	0.861	0.822	<u>0.890</u>
Coverage	<u>0.240</u>	**0.970**	0.909	<u>0.895</u>	0.288	0.827	0.788	0.886
Anticipation	0.285	0.965	0.906	0.892	0.309	0.885	0.835	0.884
Impact (Grid)	0.252	<u>0.969</u>	0.908	0.894	**0.226**	**0.923**	**0.867**	**0.893**
Impact (DME)	0.264	0.967	0.907	<u>0.895</u>	<u>0.276</u>	<u>0.913</u>	<u>0.859</u>	**0.893**
DD-PPO [147]	-	*0.967*	*0.922*	-	-	-	-	-

Table 3.4: Point Goal navigation results on the validation subset of the Gibson dataset. Underlined denotes second best.

estimation module are not used, as the global goal coincides with the episode goal coordinates, and the agent receives oracle position information. Thus, good results mainly depend on the effectiveness of the mapping module. Instead, in the noisy setting, the effectiveness of the reward used during training influences navigation performance more significantly. In this case, better numerical results originate from a better ability to estimate the precise pose of the agent during the episode.

For completeness, we also compare with the results achieved by DD-PPO [147], a method trained with reinforcement learning for the PointNav task on 2.5 billion frames, 500 times more than the frames used to train our agents.

3.3.3 REAL-WORLD DEPLOYMENT

As agents trained in realistic indoor environments using the Habitat simulator are adaptable to real-world deployment [67], we also deploy the proposed approach on a LoCoBot robot‡. We employ the PyRobot interface [97] to deploy code and trained models on the robot. To enable the adaptation to the real-world environment, there are some aspects that must be taken into account during training. As a first step, we adjust the simulation in order to reproduce realis-

---

‡https://locobot-website.netlify.com

3

tic actuation and sensor noise. To that end, we adopt the noise model proposed in [32] based on Gaussian Mixture Models fitting real-world noise data acquired from a LoCoBot. Additionally, we modify the parameters of the RGB-D sensor used in simulation to match those of the RealSense camera mounted on the robot. Specifically, we change the camera resolution and field of view, the range of depth information, and the camera height. Finally, it is imperative to prevent the agent from learning simulation-specific shortcuts and tricks. For instance, the agent may learn to slide along the walls due to imperfect dynamics in simulation [67]. To prevent the learning of such dynamics, we employ the *bump* sensor provided by Habitat and block the agent whenever it is in contact with an obstacle. When deployed in the real world, our agent is able to explore the environment without getting stuck or bumping into obstacles.

4

Explore and Explain:
Embodied Exploration and Recounting

I**N** Chapter 3 we have studied novel exploration methods for embodied agents in photo-realistic environments, in this chapter we investigate the ability of a simple exploration agent to communicate its perception in an intelligent way.

Speaking about how intelligent robots that could autonomously walk and talk were imagined a few decades ago, people used to think about Artificial Intelligence exclusively as a fictional feature, as the available machines they interacted with were purely reactive and showed no form of autonomy. Nowadays, intelligent systems are everywhere, with Deep Learning being the main engine of the AI revolution. With the research in the Embodied AI field, the capabilities of robotic agents were significantly improved especially in visual navigation and instruction following [9] tasks. At the same time, tasks at the intersection of Computer Vision and Natural Language Processing (NLP) are of particular interest

---

This Chapter is related to the publication "R. Bigazzi *et al*., Explore and Explain: Self-Supervised Navigation and Recounting, ICPR 2020" [20]. See the list of Publications on page 133 for more details.
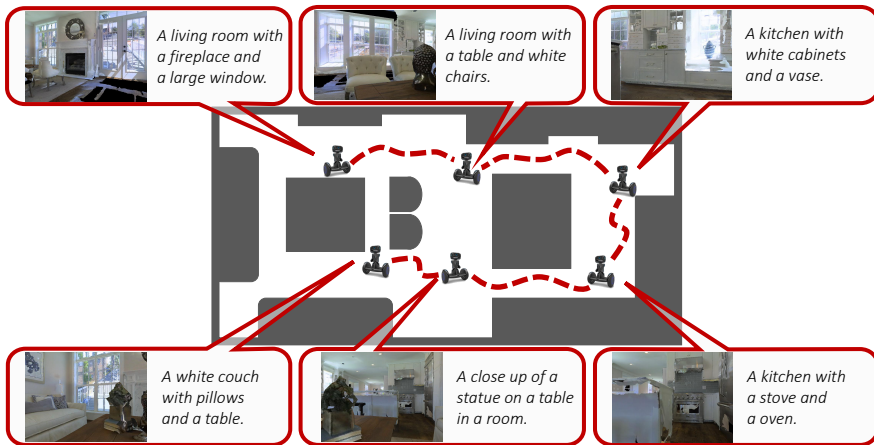
Figure 4.1: We propose a novel setting in which an embodied agent performs joint curiosity-driven exploration and explanation in unseen environments. While navigating the environment, the agent must produce informative descriptions of what it sees, providing a means of interpreting its internal state.

to the community, with Image Captioning being one of the most active areas [69, 7, 42]. By describing the content of an image or a video, captioning models can bridge the gap between the black-box architecture and the user.

In this work, we propose a new task at the intersection of Embodied AI, Computer Vision, and NLP, and aim to create a robot that can navigate through a new environment and describe what it sees. We call this new task *Explore and Explain* since it tackles the problem of joint exploration and captioning (Fig. 4.1). In this schema, the agent needs to perceive the environment around itself, navigate it driven by an exploratory goal, and describe salient objects and scenes in natural language. Beyond navigating the environment and translating visual cues into natural language, the agent also needs to identify appropriate moments to perform the explanation step.

It is worthwhile to mention that both exploration and explanation feature significant challenges. Effective exploration without any previous knowledge of the environment can not exploit a reference trajectory and the agent cannot be trained with classic methods from reinforcement learning [147]. To overcome this problem, we design a self-supervised exploration module that is driven solely

by curiosity toward the new environment. In this setting, rewards are more sparse than in traditional setups and encourage the agent to explore new places and interact with the environment.

While we are motivated by recent works incorporating curiosity in Atari and other exploration games [2, 108, 26], the effectiveness of a curiosity-based approach in a photo-realistic, indoor environment has not been tested extensively. Some preliminary studies [118] suggest that curiosity struggles with embodied exploration. In this work, we show that a simple modification of the reward function can lead to striking improvements in the exploration of unseen environments.

Additionally, we encourage the agent to produce a description of what it sees throughout the navigation. In this way, we match the agent's internal state (the measure of curiosity) with the variety and relevance of the generated captions. Such matching offers a proxy for the desirable by-product of interpretability. In fact, by looking at the caption produced, the user can more easily interpret the navigation and perception capabilities of the agent, and the motivations of the actions it takes [40]. In this sense, our work is related to goal-driven Explainable AI, *i.e.* the ability of autonomous agents to explain their actions and the reasons leading to their decisions [10].

Previous work on Image Captioning has mainly focused on recurrent neural networks. However, the rise of Transformer [144] and the great effectiveness shown by the use of self-attention have motivated a shift towards recurrent-free architectures. Our captioning algorithm builds upon the importance of fully-attentive networks for Image Captioning and incorporates self-attention both during the encoding of the image features and in the decoding phase. This also allows for a reduction in computational requirements.

Finally, to bridge exploration and recounting, our model can count on a novel speaker policy, which regulates the speaking rate of our captioner using information coming from the agent perception. We call our architecture eX^2, from the name of the task: *Explore and Explain*.

4

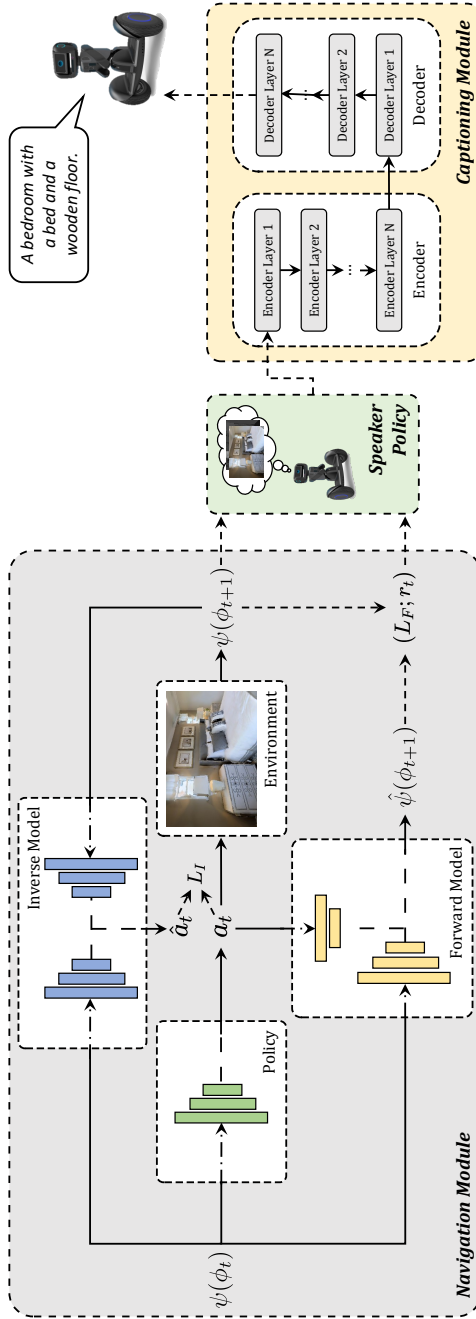**Figure 4.2:** Overview of our eX² framework for navigation and captioning. Our model is composed of three main components: a navigation module which is in charge of exploring the environment, a captioning module that produces a textual sentence describing the agent point of view, and a speaker policy that connects the previous modules and activates the captioning component based on the information collected during the navigation.

Our main contributions are as follows. We propose a new setting for Embodied AI, *Explore and Explain* in which the agent must jointly deal with two challenging tasks: exploration and captioning of unseen environments. We devise a novel solution involving curiosity for exploration. Thanks to curiosity, we can learn an efficient policy that can easily generalize to unseen environments. We apply a captioning algorithm exclusively to indoor environments for robotic exploration.

4.1 PROPOSED METHOD

The proposed method consists of three principal parts: a navigation module, a speaker policy, and a captioning module. The last two components constitute the speaker module, which is used to explain the agent's first-person point of view. The explanation is elicited by our speaker module basing on the information gathered during the navigation. Our architecture is depicted in Fig. 4.2 and detailed in the following sections. Note that, in this work, differently from the previous chapter we are investigating a navigation method that does not build a map along the way but predicts directly the actions to perform from the input.

4.1.1 NAVIGATION MODULE

The navigation policy takes care of the agent displacement inside the environment. At each timestep t, the agent acquires an observation φ_t from the surroundings, performs an action a_t, and gets the consequent observation φ_{t+1}. In this setting, the moves available to the agent are atomic actions such as *rotate 15 degrees* and *step ahead*. Our navigation module consists of three main components: a feature embedding network, a forward model, and an inverse model. The discrepancy of the predictions of dynamics models with the actual observation is measured by a reward signal r_t, which is then used to stimulate the agent to move towards more informative states.

EMBEDDING NETWORK. At each timestep t, the agent observes the environment and gathers $\varphi_t = (\varphi_t^{rgb}, \varphi_t^d)$. This observation corresponds to the raw RGB-D pixels coming from the forward-facing camera of the agent. Yet, raw

pixels are not optimal to encode the visual information [26]. For this reason, we employ a convolutional neural network (CNN) ψ to encode a more efficient and compact representation of the surrounding environment. We call this embedded representation $\psi(\varphi_t)$. To ensure that the features observed by the agent are stable throughout the training, we do not change the set of parameters of the CNN θ_ψ during training. This approach is shown to be efficient for generic curiosity-based agents [26].

FORWARD DYNAMICS MODEL. Given an agent with policy $\pi(\psi(\varphi_t); \theta_\pi)$, represented by a neural network with parameters θ_π, the selected action at timestep t is given by:

$$a_t \sim \pi\Big(\psi(\varphi_t); \theta_\pi\Big). \tag{4.1}$$

After executing the chosen action, the agent can observe a new visual stimulus $\varphi(x_{t+1})$. The problem of predicting the next observation given the current input and action to be performed can be defined as a forward dynamics problem:

$$\hat{\psi}(\varphi_{t+1}) = f\Big(\psi(\varphi_t), a_t; \theta_F\Big), \tag{4.2}$$

where $\hat{\varphi}(x_{t+1})$ is the predicted visual embedding for the next observation x_{t+1} and f is the forward dynamics model with parameters θ_F. The forward model is trained to minimize the following loss function:

$$L_F = \frac{1}{2} \left\| \hat{\psi}(\varphi_{t+1}) - \psi(\varphi_{t+1}) \right\|_2^2 \tag{4.3}$$

INVERSE DYNAMICS MODEL. Given two consecutive observations $(\varphi_t, \varphi_{t+1})$, the inverse dynamics model aims to predict the action performed at timestep t:

$$\hat{a}_t = g\Big(\psi(\varphi_t), \psi(\varphi_{t+1}); \theta_I\Big), \tag{4.4}$$

where $\hat{a}_t$ is the predicted estimate for the action a_t and g is the inverse dynamics model with parameters θ_I. In our work, the inverse model g predicts a probability distribution over the possible actions and it is optimized to minimize the cross-

entropy loss with the ground-truth action a_t performed in the previous timestep:

$$L_I = y_t \log \hat{a}_t, \tag{4.5}$$

where y_t is the one-hot representation for a_t.

CURIOSITY-DRIVEN EXPLORATION. The agent exploration policy $\pi(\psi(\varphi_t); \theta_\pi)$ is trained to maximize the expected sum of rewards:

$$\max_{\theta_\pi} \mathbb{E}_{\pi(\psi(\varphi_t); \theta_\pi)} \left[\sum_t r_t \right], \tag{4.6}$$

where the exploration reward r_t at timestep t, also called surprisal [1], is given by our forward dynamics model:

$$r_t = \frac{\eta}{2} \left\| f(\psi(\varphi_t), a_t) - \psi(\varphi_{t+1}) \right\|_2^2, \tag{4.7}$$

with η being a scaling factor. The overall optimization problem can be written as a composition of Eq. 4.3, 4.5, and 4.6:

$$\min_{\theta_\pi, \theta_F, \theta_I} \left[-\lambda \mathbb{E}_{\pi(\psi(\varphi_t); \theta_\pi)} \left[\sum_t r_t \right] + \beta L_F + (1 - \beta) L_I \right] \tag{4.8}$$

where λ weights the importance of the intrinsic reward signal *w.r.t.* the policy loss, and β balances the contributions of the forward and inverse models.

PENALTY FOR REPEATED ACTIONS. To encourage diversity in our policy, we devise a penalty that triggers after the agent has performed the same move for $\tilde{t}$ timesteps. This prevents the agent from always picking the same action and encourages the exploration of different combinations of atomic actions.

We can thus rewrite the surprisal in Eq. 4.7 as:

$$r_t = \frac{\eta}{2} \left\| f(\psi(\varphi_t), a_t) - \psi(\varphi_{t+1}) \right\|_2^2 - p_t, \tag{4.9}$$

where p_t is the penalty at timestep t. In the simplest formulation, p_t can be modeled with a scalar that is either 0 or equal to a constant $\tilde{p}$, after an action has been

repeated $\tilde{t}$ times.

4.1.2 SPEAKER POLICY

As the navigation proceeds, new observations φ_t are acquired and rewards r_t are obtained at each timestep. Based on these, a speaker policy can be defined, that activates the captioning module. Different types of information from the environment and the navigation module allow for defining different policies. In this work, we consider three policies, namely: object-driven, depth-driven, and curiosity-driven.

OBJECT-DRIVEN POLICY. Given the RGB component of the observation φ_t, relevant objects can be recognized. When at least a minimum number O of such objects is observed, the speaker policy triggers the captioning module. The idea behind this policy is to let the captioner describe the scene only when objects that allow connoting the different views are present.

DEPTH-DRIVEN POLICY. Given the depth component of the observation φ_t, the speaker policy activates the captioner when the mean depth value perceived D is above a certain threshold. This way, the captioner is triggered only depending on the distance of the agent from generic objects, regardless of their semantic category.

CURIOSITY-DRIVEN POLICY. Given the surprisal reward defined as in Eq. 4.7 and possibly cumulated over multiple timesteps, S, the speaker policy triggers the captioner when S is above a certain threshold. This policy is independent of the type of information perceived from the environment but is instead closely related to the navigation module. Thus, it helps to match the agent's internal state with the generated captions more explicitly than the other policies.

4.1.3 CAPTIONING MODULE

When the speaker policy activates, a captioning module is in charge of producing a description in natural language given the current RGB observation φ_t^{rgb}. Following recent literature on the topic, we here employ a visual encoder based on image regions [120], and a decoder that models the probability of generating

one word given previously generated ones. In contrast to previous captioning approaches based on recurrent networks, we propose to use a fully-attentive model for both the encoding and the decoding stage, building on the Transformer model [144].

REGION ENCODER. Given a set of features from image regions $R = \{r_1, ..., r_N\}$ extracted from the agent visual view, our encoder applies a stack of self-attentive and linear projection operations. As the former be seen as convolutions on a graph, the role of the encoder can also be interpreted as that of learning visual relationships between image regions. The self-attention operator S builds upon three linear projections of the input set, which are treated as queries, keys and values for an attention distribution. Stacking region features R in matrix form, the operator can be defined as follows:

$$S(R) = \text{Attention}(W_q R, W_k R, W_v R), \qquad (4.10)$$
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V.$$

The output of the self-attention operator is a new set of elements $S(R)$, with the same cardinality as R, in which each element of R is replaced with a weighted sum of the values, *i.e.* of linear projections of the input.

Following the structure of the Transformer model, the self-attention operator S is followed by a position-wise feed-forward layer, and each of these two operators is encapsulated within a residual connection and a layer norm operation. Multiple layers of this kind are then applied in a stack fashion to obtain the final encoder.

LANGUAGE DECODER. The output of the encoder module is a set of region encodings $\tilde{R}$ with the same cardinality of R. We employ a fully-attentive decoder which is conditioned on both previously generated words and region encodings and is in charge of generating the next tokens of the output caption. The structure of our decoder follows that of the Transformer [144], and thus relies on self-attentive and cross-attentive operations.

Given a partially decoded sequence of words $W = \{w_0, w_1, ..., w_\tau\}$, each represented as a one-hot vector, the decoder applies a self-attention operation in which W is used to build queries, keys and values. To ensure the causality of this sequence encoding process, we purposely mask the attention operator so that each word can only be conditioned to its left-hand sub-sequence, *i.e.* word w_t is conditioned on $\{w_{t'}\}_{t' \leq t}$ only. Afterwards, a cross-attention operator is applied between W and $\tilde{R}$ to condition words on regions, as follows:

$$C(W, \tilde{R}) = \text{Attention}(W_q W, W_k \tilde{R}, W_v \tilde{R}). \qquad (4.11)$$

As in the Transformer model, after a self-attention and a cross-attention stage, a position-wise feed-forward layer is applied, and each of these operators is encapsulated within a residual connection and a layer norm operation. Finally, our decoder stacks together multiple decoder layers, helping to refine the understanding of the textual input.

Overall, the decoder takes as input word vectors, and the t-th element of its output sequence encodes the prediction of a word at time $t + 1$, conditioned on $\{w_t\}_{\leq t}$. After taking a linear projection and a softmax operation, this encodes a probability over words in the dictionary. During training, the model is trained to predict the next token given previous ground-truth words; during decoding, we iteratively sample a predicted word from the output distribution and feed it back to the model to decode the next one, until the end of the sequence is reached. Following the usual practice in Image Captioning literature, the model is trained to predict an end-of-sequence token to signal the end of the caption.

4.2   EXPERIMENTAL SETUP

4.2.1   DATASET

The main testbed for this work is Matterport 3D [30], a photo-realistic dataset of indoor environments. Some of the buildings in the dataset contain outdoor components like swimming pools or gardens, raising the difficulty of the exploration task. The dataset is split into 61 scenes for training, 11 for validation, and 18 for

testing. It also provides instance segmentation annotations that we use to evaluate the captioning module. Overall, the dataset is annotated with 40 different semantic categories. For both training and testing, we use the episodes provided by Habitat Simulator [126] for the Point Goal navigation task, employing only the starting point of each episode. The size of the training set amounts to a total of 5M episodes, while the test set is composed of 1008 episodes.

4.2.2 Evaluation Protocol

Navigation Module. To quantitatively evaluate the navigation module, we do not use quantitative metrics as the model employs a map-less approach. Instead, we use a curiosity-based metric: we extract the sum of the surprisal values defined in Eq. 4.7 every 20 steps performed by the agent, and then we compute the average over the number of test episodes.

Captioning Module. Standard captioning methods are usually evaluated by comparing each generated caption against the corresponding ground-truth sentences. However, in this setting, only the information on which objects are contained in the scene is available, thanks to the semantic annotations provided by the Matterport 3D dataset. Therefore, to evaluate the performance of our captioning module, we define two different metrics: a soft coverage measure that assesses how the predicted caption covers all the ground-truth objects, and a diversity score that measures the diversity in terms of described objects of two consecutively generated captions.

In detail, for each caption generated according to the speaker policy, we compute the soft coverage measure between the ground-truth set of semantic categories and the set of nouns in the caption. Given a predicted caption, we firstly extract all nouns $\boldsymbol{n}$ from the sentence and we compute the optimal assignment between them and the set of ground-truth categories $\boldsymbol{c}^*$, using distances between word vectors and the Hungarian algorithm [77]. We then define an intersection score between the two sets as the sum of assignment profits. Our coverage measure is computed as the ratio of the intersection score and the number of ground-

truth semantic classes:

$$\mathsf{Cov}(\boldsymbol{n}, \boldsymbol{c}^*) = \frac{\mathrm{I}(\boldsymbol{n}, \boldsymbol{c}^*)}{\#\boldsymbol{c}^*}, \tag{4.12}$$

where $\mathrm{I}(\cdot, \cdot)$ is the intersection score, and the $\#$ operator represents the cardinality of the set of ground-truth categories.

Since images may contain small objects which not necessarily should be mentioned in a caption describing the overall scene, we define a variant of the coverage measure by thresholding over the minimum object area. In this case, we consider $\boldsymbol{c}^*$ as the set of objects whose overall areas are greater than the threshold.

For the diversity measure, we consider the sets of nouns extracted from two consecutively generated captions, indicated as $\boldsymbol{n}_t$ and $\boldsymbol{n}_{t+1}$, and we define a soft intersection over union score between the two sets of nouns. Also in this case, we compute the intersection score between the two sets using word distances and the Hungarian algorithm to find the optimal assignment. Recalling that set union can be expressed in function of an intersection, the final diversity score is computed by subtracting the intersection over union score from 1 (*i.e.* the Jaccard distance between the two sets):

$$\mathsf{Div}(\boldsymbol{n}_t, \boldsymbol{n}_{t+1}) = 1 - \frac{\mathrm{I}(\boldsymbol{n}_t, \boldsymbol{n}_{t+1})}{\#\boldsymbol{n}_t + \#\boldsymbol{n}_{t+1} - \mathrm{I}(\boldsymbol{n}_t, \boldsymbol{n}_{t+1})}, \tag{4.13}$$

where $\mathrm{I}(\cdot, \cdot)$ is the intersection score previously defined, and the $\#$ operator represents the cardinality of the sets of nouns.

We evaluate the diversity of generated captions with respect to the three speaker policies described in Sec. 4.1.2 and consider different thresholds for each policy (*i.e.* number of objects, mean depth value, and surprisal score). For each speaker policy and selected threshold, the agent is triggered a different number of times thus generating a variable number of captions during the episode. We define the agent's overall loquacity (Loq) as the number of times it is activated by the speaker policy according to a given threshold. In the experiments, we report the loquacity values averaged over the test episodes.

4.2.3 IMPLEMENTATION DETAILS

NAVIGATION MODULE. Navigation agents are trained using only visual inputs, with each observation converted to grayscale, cropped, and re-scaled to an 84×84 size. A stack of four historical observations $[\varphi_{t-3}, \varphi_{t-2}, \varphi_{t-1}, \varphi_t]$ is used for training in order to model temporal dependencies. We adopt PPO [129] as learning algorithm and employ Adam [72] as optimizer. The learning rate for all networks is set to 10^{-4} and the length of rollouts is equal to 128. For each rollout, we make 3 optimization epochs. The features $\psi(\varphi_t)$ used by the forward and backward dynamics networks are 512-dimensional and are obtained using a randomly initialized convolutional network ψ with fixed weights θ_ψ, following the approach in [26].

The model is trained using the splits described in Sec. 4.2.1, stopping the training after 10000 updates of the agent. The length of an exploration episode is 1000 steps. In our experiments, we set the parameters reported in Eq. 4.8 to $\lambda = 0.1$ and $\beta = 0.2$, respectively. Concerning the penalty p_t given to the agent to stimulate diversity (Eq. 4.9), we set $p_t = \tilde{p} = 0.01$ after the same action is repeated for $\tilde{t} = 5$ times.

SPEAKER POLICY. For the object-driven policy, we use the instance segmentation annotations provided by the Matterport3D simulator. For this policy, we select 15 of the 40 semantic categories in the dataset, discarding the contextual ones, which would not be discriminative for the different views acquired by the agent, as for example *wall*, *floor*, and *ceiling*. This way, we can better evaluate the effect of the policy without it being affected by the performance of an underlying object detector of recognizing objects in the agent's current view. Also for the depth-driven policy, we obtain the depth information of the current view from the Matterport3D simulator, averaging the depth values to extract a single score. In the curiosity-driven policy, we consider the sum of surprisal scores extracted over the last 20 steps, obtained by the agent during navigation.

CAPTIONING MODULE. To represent image regions, we use Faster R-CNN finetuned on the Visual Genome dataset [120, 7], thus obtaining a 2048-dimensional feature vector for each region. To represent words, we use one-hot vectors and lin-

Navigation Module	Surprisal
Random Exploration	0.333
eX^2 w/o Penalty for repeated actions (RGB only)	0.193
eX^2 w/o Penalty for repeated actions (Depth only)	0.361
eX^2 w/o Penalty for repeated actions (RGB + Depth)	0.439
eX^2	**0.697**

Table 4.1: Surprisal scores for different navigation policies obtained during the agent exploration of the environment. The final model achieves a higher score than the baselines.

early project them to the input dimensionality of the model, *d*. We also employ sinusoidal positional encodings [144] to represent word positions inside the sequence and sum the two embeddings before the first encoding layer. In both the region encoder and language decoder, we set the dimensionality *d* of each layer to 512, the number of heads to 8, and the dimensionality of the inner feed-forward layer to 2048. We use dropout with keep probability 0.9 after each attention layer and after position-wise feed-forward layers.

Following standard practice in image captioning [121, 7], we train our model in two phases using image-caption pairs coming from the COCO dataset [86]. Firstly, the model is trained with cross-entropy loss to predict the next token given previous ground-truth words. Then, we further optimize the sequence generation using reinforcement learning employing a variant of the self-critical sequence training [121] on sequences sampled using beam search [7]. Pretraining with cross-entropy loss is done using the learning rate scheduling strategy defined in [144] with a warmup equal to 10000 iterations. Then, during finetuning with reinforcement learning, we use the CIDEr-D score [145] as reward and a fixed learning rate equal to 5^{-6}. We train the model using the Adam optimizer [72] and a batch size of 50. During CIDEr-D optimization and caption decoding, we use beam search with a beam size equal to 5. To compute coverage and diversity metrics and for extracting nouns from predicted captions, we use the spaCy NLP toolkit[*]. We use GloVe word embeddings [110] to compute word similarities between nouns and semantic class names.

---

[*]https://spacy.io/
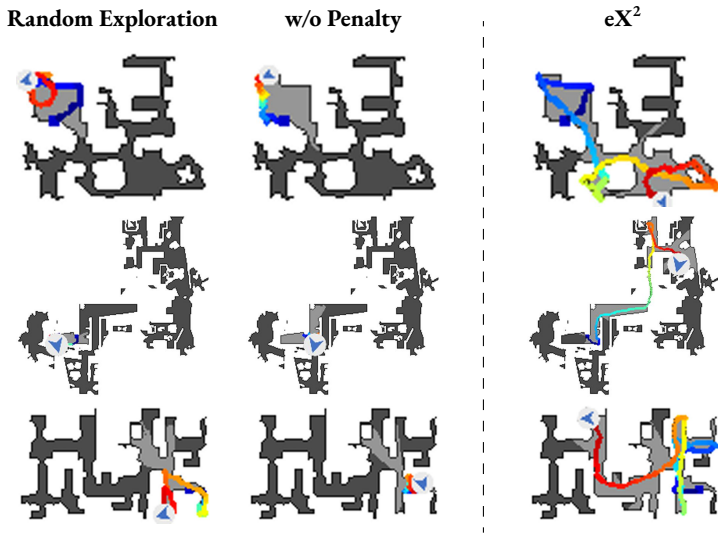
Figure 4.3: Qualitative results of the agent trajectories in sample navigation episodes.

4.3   Experimental Results

4.3.1   Exploration Results

As defined in Sec. 4.2.2, we evaluate the performance of our navigation agents by computing the average surprisal score over test episodes. Results are reported in Table 4.2 and show that our complete method (eX^2) outperforms all other variants, achieving a significantly greater surprisal score than our method without penalty. In particular, the final performance greatly benefits from using both visual modalities (RGB and depth), instead of using a single visual modality to represent the scene. Notably, random exploration (*e.g.* sampling a_t from a uniform distribution over the available actions at each timestep t) proves to be a strong baseline for this task, performing better than our single-modality RGB agent. Nonetheless, our final agent greatly outperforms the baselines, scoring 0.364 and 0.258 above the random policy and the vanilla curiosity-based agent respectively.

Qualitative Analysis. In Fig. 4.3, we report some top-down views from the testing scenes, together with the trajectory from three different navigation

agents: the random baseline, our approach without the penalty for repeated action described in Sec. 4.1.1, and our full model. We notice that the agent without penalty usually remains in the starting area and thus has some difficulties in exploring the whole environment. Instead, our complete model demonstrates better results as it is able to explore a much wider area within the environment. Thus, we conclude that the addition of a penalty for repeated actions in the final reward function is of central importance when it comes to stimulating the agent toward the exploration of regions far from the starting point.

4.3.2   Captioning Results

Here, we provide quantitative and qualitative results for our captioning module that is helped by speaker policy. In fact, the captioning module generates a description of the first-person view of the agent when the policy is triggered. For better readability, results are reported in two tables, Table 4.2 and Table 4.3, and are discussed below.

Speaker Policy. Among the three different policies, the object-driven speaker performs the best in terms of coverage and diversity. In particular, setting a low threshold ($O \geq 1$) provides the highest scores. At the same time, the agent tends to speak more often, which is desirable in a visually rich environment. As the threshold for O gets higher, performances get worse. This indicates that, as the number of objects in the scene increases, there are many details that the captioner cannot describe. The same applies to the depth-driven policy: while the agent tends to describe well items that are closer, it experiences some troubles when facing an open space with more distant objects ($D \geq 0.75$).

   Instead, our curiosity-driven speaker shows a more peculiar behavior: as the threshold grows, results get better in terms of diversity, while the coverage scores are quite stable (only -0.005% in terms of $Cov_{>1\%}$). It is also worth mentioning that our curiosity-based speaker can be adopted in any kind of environment, as the driving metric is computed from the raw RGB-D input. The same does not apply in an object-driven policy, since the agent needs semantic information. Further, the curiosity-driven policy employs a learned metric, hence being more

Object ($O \geq 1$) Loq = 43.3	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.468	0.564	0.623	0.720	**0.394**
$\mathbf{eX^2}$ (2 lay.)	**0.485**	**0.579**	**0.637**	**0.727**	0.368
$\mathbf{eX^2}$ (3 lay.)	0.474	0.558	0.612	0.701	0.372
$\mathbf{eX^2}$ (6 lay.)	0.456	0.550	0.609	0.706	0.386

Object ($O \geq 3$) Loq = 27.4	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.400	0.519	0.593	0.713	**0.377**
$\mathbf{eX^2}$ (2 lay.)	**0.416**	**0.534**	**0.607**	**0.721**	0.349
$\mathbf{eX^2}$ (3 lay.)	0.384	0.497	0.571	0.691	0.350
$\mathbf{eX^2}$ (6 lay.)	0.387	0.502	0.576	0.696	0.363

Object ($O \geq 5$) Loq = 15.8	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.356	0.479	0.560	0.702	**0.373**
$\mathbf{eX^2}$ (2 lay.)	**0.373**	**0.497**	**0.577**	**0.713**	0.340
$\mathbf{eX^2}$ (3 lay.)	0.347	0.467	0.546	0.688	0.338
$\mathbf{eX^2}$ (6 lay.)	0.348	0.468	0.549	0.691	0.352

Depth ($D > 0.25$) Loq = 38.5	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.448	0.548	0.613	0.723	**0.371**
$\mathbf{eX^2}$ (2 lay.)	**0.463**	**0.562**	**0.625**	**0.730**	0.341
$\mathbf{eX^2}$ (3 lay.)	0.427	0.524	0.588	0.700	0.349
$\mathbf{eX^2}$ (6 lay.)	0.433	0.532	0.600	0.705	0.360

Depth ($D > 0.5$) Loq = 31.1	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.434	0.536	0.603	0.719	**0.359**
$\mathbf{eX^2}$ (2 lay.)	**0.449**	**0.550**	**0.612**	**0.726**	0.330
$\mathbf{eX^2}$ (3 lay.)	0.413	0.511	0.577	0.695	0.335
$\mathbf{eX^2}$ (6 lay.)	0.420	0.519	0.585	0.701	0.346

Depth ($D > 0.75$) Loq = 14.8	$Cov_{>1\%}$	$Cov_{>3\%}$	$Cov_{>5\%}$	$Cov_{>10\%}$	Div
$\mathbf{eX^2}$ (1 lay.)	0.412	0.513	0.583	0.708	**0.355**
$\mathbf{eX^2}$ (2 lay.)	**0.425**	**0.525**	**0.595**	**0.715**	0.325
$\mathbf{eX^2}$ (3 lay.)	0.394	0.491	0.559	0.685	0.330
$\mathbf{eX^2}$ (6 lay.)	0.399	0.497	0.566	0.691	0.339

Table 4.2: Coverage and diversity results for different versions of our captioning module. Results are reported for our object-based and depth-based speaker policies using different thresholds to determine the agent's loquacity inside the episode. The model using 2 Transformer layers returns the best results.

4

Curiosity (S > 0.7)					
Loq = 27.2	$\text{Cov}_{>1\%}$	$\text{Cov}_{>3\%}$	$\text{Cov}_{>5\%}$	$\text{Cov}_{>10\%}$	Div
eX^2 (1 lay.)	0.438	0.539	0.604	0.719	**0.370**
eX^2 (2 lay.)	**0.453**	**0.552**	**0.617**	**0.726**	0.340
eX^2 (3 lay.)	0.418	0.514	0.578	0.694	0.348
eX^2 (6 lay.)	0.425	0.523	0.588	0.703	0.356
Curiosity (S > 0.85)					
Loq = 18.2	$\text{Cov}_{>1\%}$	$\text{Cov}_{>3\%}$	$\text{Cov}_{>5\%}$	$\text{Cov}_{>10\%}$	Div
eX^2 (1 lay.)	0.433	0.530	0.597	0.716	**0.373**
eX^2 (2 lay.)	**0.448**	**0.545**	**0.611**	**0.724**	0.342
eX^2 (3 lay.)	0.413	0.506	0.571	0.691	0.350
eX^2 (6 lay.)	0.421	0.515	0.581	0.699	0.360
Curiosity (S > 1.0)					
Loq = 6.4	$\text{Cov}_{>1\%}$	$\text{Cov}_{>3\%}$	$\text{Cov}_{>5\%}$	$\text{Cov}_{>10\%}$	Div
eX^2 (1 lay.)	0.434	0.532	0.597	0.717	**0.380**
eX^2 (2 lay.)	**0.448**	**0.545**	**0.610**	**0.723**	0.349
eX^2 (3 lay.)	0.413	0.506	0.570	0.690	0.361
eX^2 (6 lay.)	0.422	0.518	0.583	0.702	0.364

Table 4.3: Coverage and diversity results for different versions of our captioning module. Results are reported for our curiosity-based speaker policy using different thresholds to determine the agent's loquacity inside the episode.

related to the exploration module.

From all these observations, we can conclude that curiosity not only helps train navigation agents but also represents an important metric when bridging cross-modal components in embodied agents.

Captioning Module. When evaluating the captioning module, we compare the performance using a different number of encoding and decoding layers. As it can be seen from Table 4.2 and 4.3, the captioning model achieves the best results when composed of 2 layers for coverage and 1 layer for diversity. While this is in contrast with traditional Transformer-based models [144], which employ 6 or more layers, it is in line with recent research on image captioning [42], which finds it beneficial to adopt fewer layers. At the same time, a more lightweight network can possibly be embedded in many embodied agents, thus being more appropriate for our task.

Qualitative Analysis. We report some qualitative results for eX^2 in Fig. 4.4. To ease visualization, we underline the items mentioned by the captioner in the
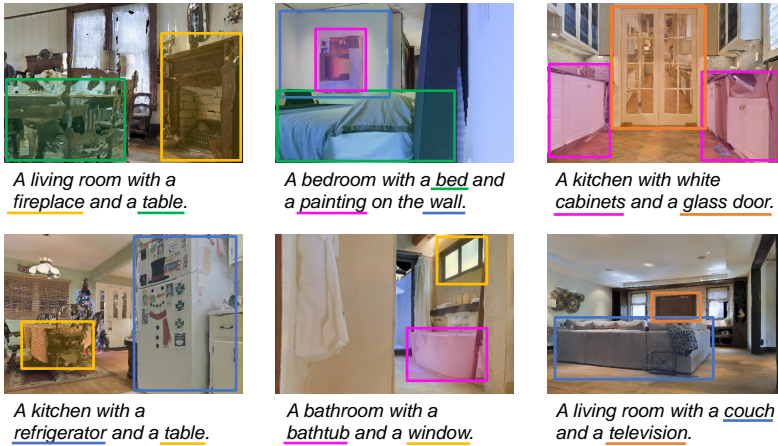
Figure 4.4: Sentences generated on sample images extracted from eX² navigation trajectories. For each image, we report the relevant objects present on the scene and we underline their mentions in the caption.

sentence and highlight them with a bounding box of the same color in the corresponding input image. Our agent can explain the scene perceived from a first-person, egocentric point of view. We can notice that eX² identifies all the main objects in the environment and produces a suitable description even when the view is partially occluded or the object presents artifacts due to the 3D model reconstruction.

4

5

Efficient Exploration and Smart Scene Description

T HE path explored with the work described in Chapter 4, adopting language to facilitate human understanding of the behavior and perception of a robotic agent, poses as a first step towards fluid interaction between robotic agents and humans. Nevertheless, even with the introduction of *Explore and Explain*, the task of joint exploration and scene description, and the implementation of eX^2 architecture, there is still a long way to go before this ambitious objective becomes a reality. In fact, eX^2 could be improved in each one of the three main components of the method, namely the navigation module, the speaker policy, and the captioning module.

For example, a significant improvement in the exploration capabilities of the agent could be enabled by the adoption of a map-based navigation policy, in this way the agent is capable of remembering areas already observed. In this regard,

---

This Chapter is related to the publication "R. Bigazzi *et al*., Embodied Agents for Efficient Exploration and Smart Scene Description, ICRA 2023" [17]. See the list of Publications on page 133 for more details.

a plethora of exploration rewards and strategies is available in the representation learning community, and generally, those strategies could be adapted to work on board smart autonomous agents to improve their perception of the surrounding.

At the same time, apart from the depth-based and the object-based speaker policies, the curiosity-based policy was specifically designed for a curiosity-based navigation policy, going against the modularity of the proposed method. Instead, on the captioning side, the advent of foundation models like CLIP [113] in the last few years brought significant advances to the methods available in the literature.

Furthermore, the absence of a metric designed to evaluate agents on *Explore and Explain* was a major lack.

We present a revised pipeline for an autonomous agent for efficient exploration and mapping of unseen environments, providing user-understandable representations of the perceived environment and avoiding unnecessary repetitions.

The key contributions of this work are twofold. First, we combine state-of-the-art approaches for image captioning and visual exploration to tackle *Explore and Explain* task with the aim of improving human understanding of robotic perception. Second, we devise a novel metric, called episode description score (ED-S), that evaluates the exploration and the ability to cover objects in the environment avoiding repetitions. We extensively test the performance of the proposed approach and validate the value of ED-S on both Gibson [149] and Matterport 3D [30] datasets. Furthermore, while our approach is trained and evaluated in simulation, the proposed architecture is designed for the final deployment on a real robotic platform.

5.1 Proposed Method

Our proposed architecture remains composed of three main components: a navigation module, in charge of the exploration, a captioning module, that describes interesting scenes, and the speaker policy that decides when the captioner should be activated. An overview of the updated complete architecture is shown in Fig. 5.1.
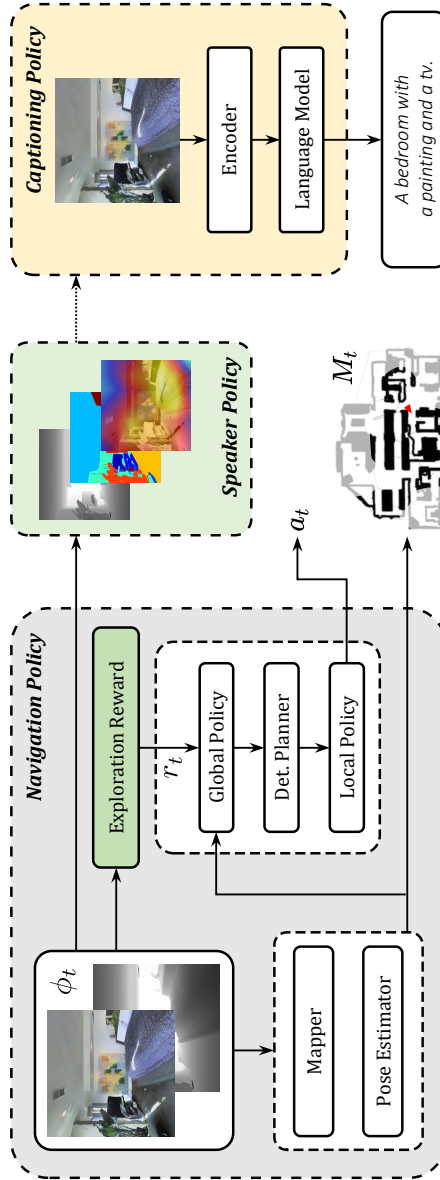
Figure 5.1: Overview of the proposed approach for smart scene description, comprising a navigation module, a speaker policy, and a captioning module.

5.1.1 NAVIGATION MODULE

The exploration capabilities of the agent are strictly dependent on the performance of the navigation module, therefore relying on a proper navigation approach is of fundamental importance. Following recent literature on embodied visual navigation [32, 115, 116], we adopt a hierarchical policy coupled with a learned neural occupancy mapper and a pose estimator. The hierarchical policy sets long and short-term navigation goals, while the neural mapper builds an occupancy grid map representation of the environment and the pose estimator locates the agent on such map.

MAPPER. In order to track explored and unexplored regions of the environment over time, the mapper is a fundamental component. In this work, as presented in Chapter 3, we use a neural-based mapper that allows inferring region occupancy beyond the observable area in front of the agent, facilitating the planning phase [115]. The output of the mapper is a $W \times W \times 2$ global map of the environment M_t that keeps track of the non-traversable space in its first channel and the area explored by the agent in the second one. The mapper processes the RGB-D observation $\varphi_t = (\varphi_t^{rgb}, \varphi_t^d)$ coming from the agent and predicts a $V \times V \times 2$ egocentric local map m_t representing the state in front of the agent. More details on how the local map is output are described in Section 3.1.1. At every timestep t, the local map m_t is transformed using the estimated pose of the agent $\omega_t = (x_t, y_t, \theta_t)$ and registered to the global map M_t with a moving average. The global map is initially empty and is built incrementally with the exploration of the environment.

POSE ESTIMATOR. Relying on a global map requires a robust pose estimator in order to build geometrically coherent and precise maps. Indeed, an inaccurate pose estimate would rapidly diverge from the ground-truth pose, and loop closure is inapplicable if previous knowledge of the environment is not available. Furthermore, directly using the pose sensor of the robot is not sufficient since sensor noise, slipping wheels, and collisions with obstacles would not be taken into consideration. The adopted approach uses the difference between consecutive pose sensor readings $\Delta \omega_t' = \omega_t' - \omega_{t-1}'$ as a first estimate of the motion of the agent, where $\omega_t' = (x_t', y_t', \theta_t')$, with (x_t', y_t') being the coordinates on the map, and
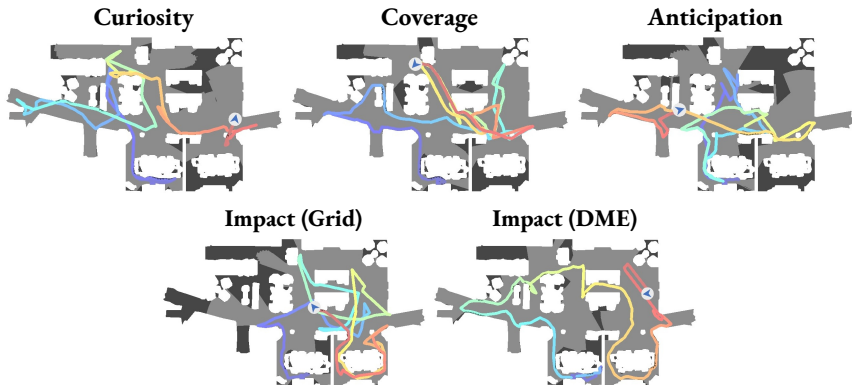
Figure 5.2: Qualitative exploration trajectories of different navigation agents on the same episode.

θ'_t the orientation of the agent. In order to correct eventual inaccuracies, we use local maps m_t, m_{t-1} extracted from the respective observations as feedback. The local map m_{t-1} is rototranslated with respect to the current position of the agent using $\Delta\omega'_t$. Transformed m_{t-1} and m_t are concatenated and fed to a CNN to output a corrected displacement $\Delta\omega_t$. At every timestep, $\Delta\omega_t$ is used to compute the pose of the agent with respect to the pose at the previous step:

$$\omega_t = \omega_{t-1} + \Delta\omega_t \quad \text{where} \quad \omega_t = (x_t, y_t, \theta_t). \tag{5.1}$$

Without loss of generality, we consider the agent starting from $\omega_0 = (0, 0, 0)$, *i.e.* the center of the global map M_t.

NAVIGATION POLICY. The navigation policy adopts a hierarchical structure as used in embodied literature [32, 115, 116]. Specifically, the navigation policy comprehends three modules: a high-level global policy, a deterministic planner, and an atomic local policy. In this way, the hierarchical policy considers both high-level and low-level concepts like moving across rooms and avoiding obstacles. The implementation of the navigation policy follows the same architecture presented in Section 3.1.3.

The global policy takes as input an enriched version of the current global map m_t and outputs the global goal g_t, that is a coordinate to be reached. The global

policy is trained with reinforcement learning using the global reward r_t^{global}.

The planner consists of an A* algorithm. It uses the global map to plan a path towards the global goal and samples a local goal l_t within 1.25m from the position of the agent.

The local policy takes as input the current RGB observation φ_t^{rgb} as well as the relative displacement of the local goal l_t from agent's position ω_t, and predicts the atomic action needed to reach the local goal. The output of the local policy corresponds to one of the following atomic actions: *move forward 0.25m*, *turn left 10°*, and *turn right 10°*. This policy is trained with a reward r_t^{local} that encourages the decrease in the geodesic distance between the agent and the local goal.

5.1.2 Exploration Rewards

We compare various global exploration rewards such as curiosity [108], coverage [33], anticipation [115], and impact [18]. All the considered methods obtain the reward by exploiting visual input sensors only. Exemplar exploration trajectories resulting from the different rewards are reported in Fig. 5.2.

Curiosity. The curiosity reward follows the same paradigm presented in Chapter 4, but in this case, the penalty presented in Section 4.1.1 is not used. When using the curiosity-based reward the navigation policy adopts two additional neural networks that learn the environment dynamics. The forward model trained to predict the encoding of the future RGB observation given the encoding of the current observation and action and the inverse model trained to infer the action a_t performed between consecutive observations $(\varphi_t^{rgb}, \varphi_{t+1}^{rgb})$. These models are trained by minimizing the following losses:

$$L_F = \frac{1}{2} \left\| \hat{\psi}(\varphi_{t+1}^{rgb}) - \psi(\varphi_{t+1}^{rgb}) \right\|_2^2 \quad \text{and} \quad L_I = y_t \log \hat{a}_t, \qquad (5.2)$$

where $\hat{\psi}(\varphi^{rgb})$ and $\psi(\varphi^{rgb})$ denote predicted and ground-truth encodings of the observation φ^{rgb}, y is the one-hot encoding of the ground-truth action a, and $\hat{a}$ is the predicted action probability distribution. The global reward for the curiosity-driven model is given by the error of the forward dynamics model prediction dur-

ing the navigation:

$$r_t^{global} = \frac{\eta}{2} \| \hat{\psi}(\varphi_{t+1}^{rgb}) - \psi(\varphi_{t+1}^{rgb}) \|_2^2, \tag{5.3}$$

where η is a normalizing term set to 0.01.

COVERAGE. The coverage-based reward maximizes the information gathered at each timestep, being the number of objects or landmarks reached or area seen. In this work, we consider the area seen definition, as proposed in [32]:

$$r_t^{global} = AS_t - AS_{t-1}, \tag{5.4}$$

where AS indicates the number of pixels explored in the ground truth map.

ANTICIPATION. The occupancy anticipation reward [115] aims to maximize accuracy in the prediction of the map including occluded unseen areas, *i.e.*,

$$r_t^{global} = Acc(M_t, M^*) - Acc(M_{t-1}, M^*), \tag{5.5}$$

$$Acc(M, M^*) = \sum_{i=1}^{W^2} \sum_{j=1}^{2} \mathbb{1}[M_{ij} = M_{ij}^*], \tag{5.6}$$

where M is the predicted global map, M^* is the ground truth global map, and $\mathbb{1}[\cdot]$ is the indicator function.

IMPACT. The impact reward encourages actions that modify agent's internal representation of the environment, with impact at timestep t that is measured as the l_2-norm of the encodings of the two consecutive states $\psi(s_t)$ and $\psi(s_{t+1})$. However, using the formulation of impact as it is, could lead to trajectory cycles with high impact but low exploration. To overcome this issue, Raileanu *et al.* [114] uses the state visitation count $N(s_t)$ to scale the reward. Unfortunately in our setting the concept of the visitation count is not directly applicable, due to the continuous space of the photo-realistic environment. Hence, we adopt and evaluate the impact-based methods proposed in [18]. Such methods formalize a pseudo-visitation count $\widehat{N}(s_t)$ in continuous environments with two different approaches: grid and density model estimation. The final global reward for the

impact-driven model becomes:

$$r_t^{global} = \left\| \psi(\varphi_{t+1}^{rgb}) - \psi(\varphi_t^{rgb}) \right\|_2 \Big/ \sqrt{\widehat{N}(s_t)}, \tag{5.7}$$

where $\psi(\varphi_t^{rgb})$ and $\widehat{N}(s_t)$ are the encoding and the estimated pseudo-visitation count at timestep t.

5.1.3 CAPTIONING MODULE

The goal of the captioning module is that of modeling an autoregressive distribution probability $p(\boldsymbol{w}_t | \boldsymbol{w}_{\tau < t}, V)$, where V is an image captured from the agent and $\{\boldsymbol{w}_t\}_t$ is the sequence of words comprising the generated caption. This is usually achieved by training a language model conditioned on visual features to mimic ground-truth descriptions. For multimodal fusion, we employ an encoder-decoder Transformer [144] architecture. Each layer of the encoder employs multi-head self-attention (MSA) and feed-forward layers, while each layer of the decoder employs multi-head self- and cross-attention (MSCA) and feed-forward layers. For enabling text generation, sequence-to-sequence attention masks are employed in each self-attention layer of the decoder.

To obtain the set of visual features V for an image, we employ a visual encoder that is pretrained to match vision and language (*i.e.* CLIP [113]). Compared to using features extracted from object detectors [7, 157], our strategy is beneficial in terms of both computational efficiency and feature quality. The visual descriptors $V = \{\boldsymbol{v}_i\}_{i=1}^N$ are encoded via bi-directional attention in the encoder, while the token embeddings of the caption $W = \{\boldsymbol{w}_i\}_{i=1}^L$ are inputs of the decoder, where N and L indicate the number of visual embeddings and caption tokens, respectively. The overall network operates according to the following schema:

$$\begin{aligned} \text{encoder} \quad & \tilde{\boldsymbol{v}}_i = \text{MSA}(\boldsymbol{v}_i, V) \\ \text{decoder} \quad & O_{\boldsymbol{w}_i} = \text{MSCA}(\boldsymbol{w}_i, \tilde{V}, \{\boldsymbol{w}_t\}_{t=1}^i), \end{aligned} \tag{5.8}$$

where O is the network output, $\text{MSA}(\boldsymbol{x}, Y)$ is a self-attention with $\boldsymbol{x}$ mapped to

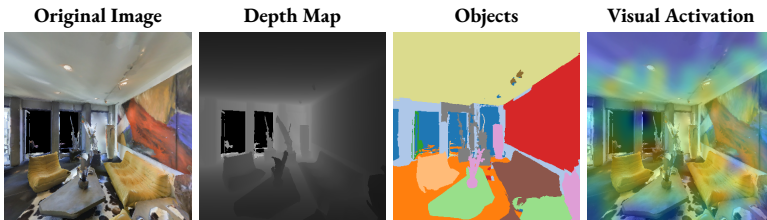| Original Image	Depth Map	Objects	Visual Activation

Figure 5.3: A sample of agent observation and corresponding images used by the speaker policy to trigger the captioner.

query and Y mapped to key-values, and $\text{MSCA}(\boldsymbol{x}, Y, Z)$ indicates a self-attention with $\boldsymbol{x}$ as query and Z as key-values, followed by cross-attention with $\boldsymbol{x}$ as query and Y as key-values. We omit feed-forward layers and the dependency between consecutive layers for ease of notation.

5.1.4 Speaker Policy

While exploring the environment, the agent sees various RGB observations φ_t^{rgb}. Even if the agent is navigating efficiently, the majority of the observations will be overlapped with each other, and the same objects will be observed at multiple consecutive timesteps. Since the agent should describe only relevant scenes during exploration and avoid uninformative captions or unnecessary repetitions, a criterion for making the captioner generate a description becomes necessary. Similarly to [20], we use a speaker policy responsible for triggering the captioner. We compare three approaches that exploit different modalities and can be used regardless of the methods used for the other components of architecture: a depth-based policy, an object-based policy, and a visual activation-based policy.

An example of the considered modalities for the same observation is reported in Fig. 5.3.

DEPTH-BASED POLICY. The depth-driven policy uses the current depth observation φ_t^d and computes the mean depth value. The captioner is activated if the mean depth value is above a predetermined threshold D. High mean depth values indicate a larger area observed by the agent, and potentially, a richer scene to be described. In fact, when the field of view of the agent is occluded by an obstacle,

the mean depth value of the observation φ_t^d is typically low.

OBJECT-BASED POLICY. Considering that the description of the scene will concentrate on relevant objects, the object-driven policy uses the number of relevant objects in the RGB observation φ_t^{rgb} to decide if the captioner should generate the description. Specifically, the captioner is triggered only if at least a number O of objects are being observed in the scene since using observations with multiple objects allows a larger variety of generated captions.

VISUAL ACTIVATION-BASED POLICY. Another possible strategy to implement the speaker policy entails exploiting the activation maps of the same visual encoder used by the captioner (which is a CLIP-like [113] encoder, as detailed in Sec. 5.1.3). Such a speaker policy is more closely related to the captioning module and provides a means to interpret the image regions that are more relevant to the agent. In particular, in this work, we consider a CNN-based visual encoder and thus take the feature maps from the last convolutional block, projected into a d-dimensional vector. This vector is then averaged, and the speaker policy is activated if its average is above a certain threshold A, thus indicating the presence of sufficient semantic content in the image.

5.2 EXPERIMENTAL SETUP

5.2.1 IMPLEMENTATION DETAILS

NAVIGATION MODULE. All the exploration models are trained for 5M frames on Gibson Dataset [149] environments using Habitat simulator [126]. The evaluation is performed using the test split of Matterport 3D (MP3D) dataset [30] and the validation split of Gibson tiny dataset, because they contain object annotations that are used to evaluate the generated captions.

The RGB-D input to the components of the navigator is resized to 128 × 128 pixels, and the global map size is $W = 2001$ for MP3D and $W = 961$ and Gibson environments. The size of the local map predicted by the mapper is $V = 101$, and each pixel in the maps describes a 5 × 5 cm^2 of the environment. Regarding the global policy, the global goal is sampled every $N_G = 25$ timesteps.

Both the global and local policies are trained using PPO algorithm [129] with a learning rate of 2.5×10^{-4}, while the mapper and the pose estimator use a learning rate of 10^{-3}. Episode length is set to 500 and 1000 for the training and evaluation phases, respectively.

SPEAKER POLICY. The performance of the navigation module, which is able to move efficiently and capture interesting observations most of the time made it necessary to choose appropriate thresholds for the activation of the policy. Moreover, since the MP3D dataset has richer object annotations and larger environments than the Gibson dataset, we compare two different sets of threshold values depending on the evaluation dataset for the depth- and object-based policies for triggering the captioner. In particular, the threshold values for MP3D dataset are $\mathsf{D} = (1.0, 2.0, 3.0)$ and $\mathsf{O} = (1, 3, 5)$ for depth- and object-based policies. Depth and object thresholds are $\mathsf{D} = (1.0, 1.5, 2.0)$ and $\mathsf{O} = (1, 2, 3)$ for Gibson dataset. On the other hand, for the activation-based criterion, we use the same set of threshold values for both the evaluation datasets, *i.e.* $\mathsf{A} = (4.5, 5.0, 5.5)$.

CAPTIONING MODULE. As training and evaluation dataset, we employ COCO [86] following the splits defined in [69]. To improve the generalization abilities of the model, we also train a variant on a combination of 35.7M images taken from both human-collected datasets (*i.e.* COCO [86]) and web-collected sources (*i.e.* SBU [101], CC3M [131], CC12M [31], WIT [135], and a subset of YFCC100M [141]).

We consider three configurations of the captioner, varying the number of decoding layers l, model dimensionality d, and number of attention heads h: Tiny ($l = 3, d = 384, h = 6$), Small ($l = 6, d = 512, h = 8$), and Base ($l = 12, d = 768, h = 12$). For all models, we employ CLIP-ViT-L/14 [113] as visual feature extractor and three layers in the visual encoder. To assess the effectiveness of CLIP-based features, we also consider a variant of the Tiny model that employs region-based visual features, extracted from Faster R-CNN [120, 7]. We train all captioning variants with cross-entropy loss using LAMB [155] as optimizer. We employ the learning rate scheduling strategy proposed in [144], with a warmup

of 6000 iterations and a batch size equal to 1080. We additionally finetune the models with the SCST strategy [121], by using the Adam optimizer [72], a fixed learning rate equal to 5×10^{-6}, and a batch size of 80.

5.2.2 Evaluation Protocol

As the task considered requires both exploration and description capabilities, for evaluation we propose to use both exploration and captioning-specific metrics, and a specific score devised for the task.

Navigation Module. As for the performance of the navigation module, we express them in terms of metrics that are commonly used for evaluating embodied exploration agents. In particular, we consider the intersection-over-union between the ground-truth map of the environment and the map reconstructed by the agent (IoU), the extent of correctly mapped area (*i.e.* the map accuracy Acc), and the extent of environment area visited by the agent (*i.e.* the area seen AS), both expressed in m^2.

Captioning Module. For evaluating the performance of the captioner on the COCO dataset, we consider the standard image captioning metrics BLEU-4 [105], METEOR [12], ROUGE [85], CIDEr [145], and SPICE [6].

Episode Description Score. Different from standard captioning settings, where the ground truth captions are available for the images, in our setting such information is not available. However, the considered 3D environments datasets come with annotations of the objects in the scene, which can be exploited for performance evaluation. In particular, based on the objects in the scene, we use the soft-coverage score (Cov) and the diversity score (Div) as presented in Chapter 4, to evaluate the ability of the agent to mention all the relevant objects in the scene and to produce interesting, non-repetitive descriptions, respectively. The first is computed by considering the intersection score between the set of nouns in the produced caption and the set of categories of the relevant objects in the scene. Note that in this work, by 'relevant object' we mean those whose area covers at least 10% of the total image area, and thus, can be more useful to identify a scene. The latter score is defined as the intersection-over-union between the sets

of nouns mentioned in two consecutively generated captions.

Additionally, we measure the agent's overall loquacity (Loq) as the number of times it is activated by the speaker policy, normalized by the episode length. In other words, the loquacity can be seen as the inverse of the average number of navigation steps between two consecutive captions. Moreover, we resort to the recently-proposed CLIP score (CLIP-S) [56], in its unpaired definition.

Finally, to evaluate an overall system on each episode of the proposed task, we define an ad hoc score to measure the concept coverage of the generated descriptions, which is an important aspect of the task. The proposed episode description score (ED-S) reflects the ability of the robot to produce sufficient descriptions in strategic moments so that the maximum amount of information collected in the environment is covered. The rationale is that it should capture the ability of the agent to mention all the relevant landmarks (objects and rooms) when needed, without unnecessary repetitions. This makes the description more useful and interesting. The score is defined as:

$$\text{ED-S} = \overline{\text{CLIP-S}} \cdot \text{IoU}(\mathbf{N}, \mathbf{O}) \cdot \text{\%AS}, \tag{5.9}$$

where $\overline{\text{CLIP-S}}$ is the mean of the CLIP scores of all the captions produced during the episode. Moreover, $\mathbf{N}$ is the list of nouns in all the captions produced during the episode, and $\mathbf{O}$ is the list of objects in the environment. The intersection-over-union operator IoU is implemented via the Jonker-Volgenant linear assignment algorithm [66]. Finally, %AS is the percentage of the total environment area visited by the agent. At the dataset level, the ED-S is given by the average of the scores obtained in the dataset episodes.

5.3 Experimental Results

5.3.1 Navigation Results

First, we compare the different exploration approaches alone on the MP3D and Gibson datasets. The results of this analysis are reported in Table 5.1. The best agent in terms of the area seen (AS) is the impact-based method using density

Model	Gibson Val			MP3D Test		
	IoU	Acc	AS	IoU	Acc	AS
Curiosity [118]	0.528	66.19	102.59	0.368	130.34	186.67
Coverage [32]	0.608	73.69	102.66	0.417	146.16	195.03
Anticipation [115]	0.706	81.13	102.22	0.494	157.02	177.14
Impact (Grid) [18]	**0.738**	**82.91**	104.16	**0.519**	164.26	185.13
Impact (DME) [18]	0.694	79.47	**105.03**	0.496	**167.58**	**205.02**

Table 5.1: Navigation results on Gibson Val and MP3D Test. Impact (DME) model achieves the best results in terms of area seen.

	Train Ims	BLEU-4	METEOR	ROUGE	CIDEr	SPICE
Region-based[tiny]	112k	37.7	28.3	57.6	124.8	21.9
CLIP-based[tiny]	112k	40.6	30.0	59.9	139.4	23.9
CLIP-based[small]	112k	40.9	30.4	60.1	141.5	24.5
CLIP-based[base]	112k	41.4	30.2	60.2	142.0	24.0
CLIP-based[base]	35.7M	**42.9**	**31.4**	**61.5**	**149.6**	**25.0**

Table 5.2: Captioning results on the COCO test set. CLIP-based[base] achieves the best results on all the episode description metrics.

model estimation. In particular, this approach is able to efficiently explore both Gibson and MP3D datasets, giving its best in large environments. In fact, the small $0.87m^2$ margin over the second best approach on Gibson becomes $9.99m^2$ in the larger MP3D environments. Moreover, this method is still competitive in terms of IoU, being also the best in terms of Acc on the MP3D test split. In light of these results, we use the impact-based navigator with DME as the navigator of the overall approach.

5.3.2 CAPTIONING RESULTS

Then, we evaluate the performance of the captioner alone on the COCO dataset. The results of this analysis are reported in Table 5.2. It can be observed that the CLIP-based variants are the best-performing ones, with a noticeable advantage over the region-based captioner. This confirms the representative power of CLIP features. The Base variant has also been trained on additional image-caption pairs

	Loq	COCO Only				COCO + Web-Collected			
		Cov	Div	CLIP-S	ED-S	Cov	Div	CLIP-S	ED-S
Always	100.00	0.864	0.352	0.670	0.119	0.862	0.348	0.692	0.120
Depth									
$D \geq 1.0$	83.26	0.868	0.335	0.670	0.140	0.865	0.338	0.690	0.140
$D \geq 1.5$	55.24	0.871	0.323	0.664	0.203	0.868	0.330	0.683	0.204
$D \geq 2.0$	27.38	0.793	0.293	0.629	0.250	0.780	0.304	0.650	0.257
Object									
$O \geq 1$	41.73	0.793	0.314	0.663	0.222	0.784	0.332	0.682	0.225
$O \geq 2$	21.55	0.703	0.289	0.645	0.219	0.697	0.307	0.664	0.220
$O \geq 3$	7.58	0.416	0.232	0.549	0.107	0.410	0.260	0.561	0.105
Activation									
$A \geq 4.5$	87.79	0.866	0.340	0.672	0.134	0.864	0.343	0.691	0.134
$A \geq 5.0$	51.13	0.828	0.349	0.674	0.223	0.827	0.348	0.691	0.220
$A \geq 5.5$	2.20	0.133	0.153	0.455	0.038	0.140	0.153	0.464	0.040

Table 5.3: Episode description results on Gibson tiny validation set.

from web-collected sources, which further increases its performance. It is worth mentioning that these results are in line with those of state-of-the-art captioners (*e.g.* [83, 157]). In light of these results, we use the CLIP-based Base variant as the captioner of the overall approach.

5.3.3 Episode Description Results

Finally, we compare variants of the overall approach using different speaking policies with different threshold values, and use as reference a dummy policy according to which the captioning module is always activated. The results are reported in Tables 5.3 and 5.4. It can be noticed that the captioner trained on web-collected sources performs better that the variant trained on COCO only in terms of all metrics, suggesting its superior generalization capabilities and thus, suitability to be employed in an embodied setting. However, to evaluate on the overall task, the proposed ED-S score is more informative than the other metrics, which can nonetheless be used in combination with the ED-S to gain additional insights on the agents' behaviour. In fact, the values of all metrics but the ED-S are compa-

	Loq	COCO Only				COCO + Web-Collected			
		Cov	Div	CLIP-S	ED-S	Cov	Div	CLIP-S	ED-S
Always	100.00	0.768	0.363	0.648	0.172	0.771	0.348	0.687	0.179
Depth									
D $\geq$ 1.0	89.05	0.765	0.352	0.648	0.180	0.767	0.341	0.687	0.187
D $\geq$ 2.0	45.06	0.751	0.317	0.637	0.155	0.750	0.311	0.668	0.160
D $\geq$ 3.0	15.98	0.317	0.161	0.338	0.030	0.317	0.151	0.360	0.031
Object									
O $\geq$ 1	75.82	0.754	0.340	0.635	0.190	0.756	0.333	0.670	0.196
O $\geq$ 3	46.57	0.700	0.310	0.605	0.168	0.701	0.310	0.634	0.172
O $\geq$ 5	19.90	0.616	0.255	0.533	0.106	0.614	0.254	0.553	0.107
Activation									
A $\geq$ 4.5	82.05	0.765	0.348	0.641	0.106	0.767	0.337	0.676	0.107
A $\geq$ 5.0	46.28	0.754	0.350	0.636	0.153	0.757	0.341	0.667	0.158
A $\geq$ 5.5	1.28	0.325	0.118	0.347	0.015	0.328	0.116	0.362	0.016

Table 5.4: Episode description results on MP3D test set.

rable in both datasets, while the ED-S is on average higher on Gibson: this is due to the fact that Gibson has on average smaller and less cluttered spaces, which can be more easily fully explored (higher values of the Cov on Gibson confirm this intuition). This trend is further confirmed by the fact that on the Gibson dataset, the speaking policy must ensure the loquacity (Loq) being in a specific range (roughly between 20 and 80) to obtain the best ED-S scores, while on the wider spaces of MP3D, speaking policies ensuring a higher Loq lead to better performance. Qualitative examples of the output of our approach on selected observations are reported in Fig. 5.4.

5.3.4 Real-World Deployment

Using exploration agents trained on the photo-realistic environments of the Habitat simulator and general-purpose captioners allows the deployment of our approach to the real world. In this respect, we use a LoCobot platform [88]. For the deployment, the captioner is left untouched, whilst we modify the camera parameters of the navigator, such as camera height, field of view, and depth sen-
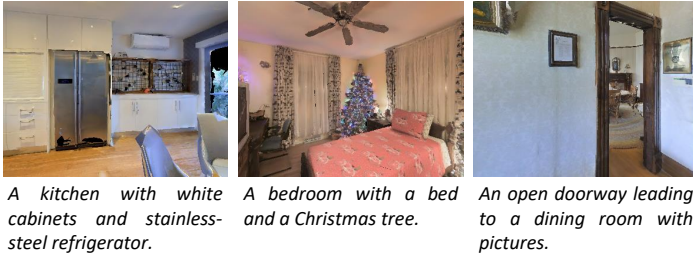
A kitchen with white cabinets and stainless-steel refrigerator.

A bedroom with a bed and a Christmas tree.

An open doorway leading to a dining room with pictures.

Figure 5.4: Sample observations and corresponding captions generated by our model.

5

sor range to match the real-world setting. Furthermore, the deployed agent is trained by adding noise models fitted to mimic the LoCobot camera noise over the observations retrieved from the simulator. As the last step, we apply the correction presented by [21] to correct noisy real-world depth observations. We test the agent exploration and description in a real-world apartment.

5

6

Spot the Difference:
Embodied Agents in Changing Environments

THE work described in previous chapters follows the same paradigm frequently used in Embodied AI, the one of starting every embodied navigation episode without any previously acquired knowledge of the environment. However, now imagine you have just bought a personal robot, and you ask it to bring you a cup of tea. It will start roaming around the house while looking for the cup. It probably will not come back until some minutes, as it is new to the environment. After the robot knows your house, instead, you expect it to perform navigation tasks much faster, exploiting its previous knowledge of the environment while adapting to possible changes in objects, people, and furniture positioning. In fact, usually in the literature on Embodied AI, the agent is initialized in a completely unknown environment. We believe that this choice is not supported by real-world experience, where information about the environment can be stored and reused for future tasks. Nevertheless, as agents are likely

---

This Chapter is related to the publication "F. Landi *et al.*, Spot the Difference: A Novel Task for Embodied Agents in Changing Environments, ICPR 2022" [81]. See the list of Publications on page 133 for more details.
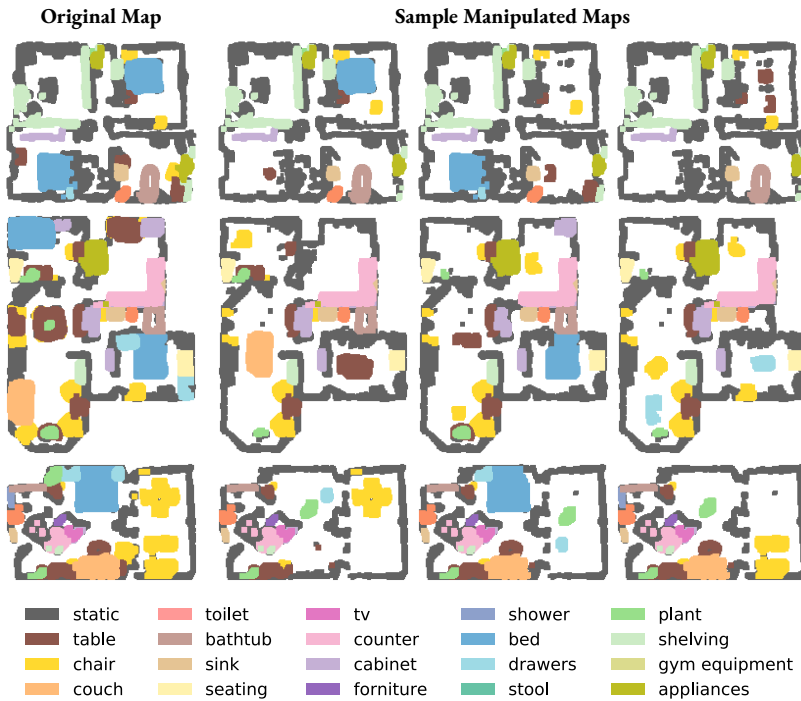
6

Figure 6.1: Generation of alternative states of an environment: original and sample manipulated semantic maps.

to stay in the same place for long periods, such information may be outdated and inconsistent with the actual layout of the environment. Therefore, the agent also needs to discover those differences during navigation.

In this work, we introduce a new task for Embodied AI, which we name *Spot the Difference*. In the proposed setting, the agent must identify all the differences between an outdated map of the environment and its current state, a challenge that combines visual exploration using monocular images and embodied navigation with spatial reasoning. To succeed in this task, the agent needs to develop efficient exploration policies to focus on likely changed areas while exploiting priors about objects of the environment. We believe that this task could be useful to train agents that will need to deal with changing environments.

Recent work on Embodied AI [38, 91, 118, 68, 94], considers building a rep-

resentation of the state of the environment to increase the performance on both exploration and down-stream tasks. Unfortunately, if the environment changes over time, the agent needs to rebuild a full representation from scratch and cannot count on an efficient policy to update its internal representation of the environment. In the following, we simulate the natural evolution of an environment and design a specific policy to navigate in changing environments seamlessly.

Due to the high cost of 3D acquisitions from the real world, existing datasets of photo-realistic 3D spaces [30, 149] do not contain different layouts for the same environment. In this work, we create a reproducible setup to generate alternative layouts for an environment. We semi-automatically build a dataset of 2D semantics occupancy maps in which the objects can be removed and rearranged while the area and the position of architectural elements do not change (Fig. 6.1). In the proposed setting, the agent is deployed in an interactive 3D environment and provided with a map from our produced dataset, which represents the information retained while performing tasks in a past state of the environment.

To train agents that can deal with changing environments efficiently, we develop a novel reward function and an approach for navigation aiming at finding relevant differences between the previous layout of the environment and the current one. Our method is based on Active Neural SLAM paradigm proposed in [32] and [115]. Differently from previous proposals, it can read and update the given map to identify relevant differences in the environment in the form of their projections on the map. Experimental results show that our approach performs better than existing state-of-the-art architectures for exploration in our newly-proposed task. We also compare with different baselines and evaluate our agent in terms of the percentage of area seen in the environment, percentage of discovered differences, and metric curves at varying exploration time budgets. The new dataset, together with our code and pretrained models, is released publicly[*].

---

[*]https://github.com/aimagelab/spot-the-difference

Dataset Split	Semantic Classes	Scans	Generated semantic occupancy maps	Episodes
MP3D Train	42	58	2070	$\approx 4.5\text{M}$
MP3D Val	42	9	160	320
MP3D Test	42	14	260	610
Gibson Val	20	5	130	450

Table 6.1: Number of manipulated maps generated per dataset split.

6.1 SPOT THE DIFFERENCE TASK

At the beginning of an episode, the agent is spawned in a 3D environment and is given a prebuilt occupancy map M, representing its spatial knowledge of the environment, *i.e.* a previous state of the environment that is now obsolete:

$$M = (m_{ij}) \in [0,1], \quad 0 \le i,j < W, \tag{6.1}$$

where m_{ij} represents the probability of finding an obstacle at coordinates (i,j). The task entails exploring the current environment to recognize all the differences with respect to the state in which M was computed, in the form of free and occupied space. To accomplish the task, the agent should build a correct occupancy map of the current environment starting from M, recognizing and focusing on parts that are likely to change (*e.g.*, the middle of wide rooms rather than tight corridors).

For every episode of *Spot the Difference*, the agent is given a time budget of T timesteps. At time $t = 0$, the agent holds the starting map representation M. At each timestep t, the map is updated depending on the current observation to obtain M_t. Whenever the agent discovers a new object or a new portion of free space, the internal representation of the map changes accordingly. The goal is to gather as much information as possible about changes in the environment by the end of the episode. To measure the agent performance, we compare the final map M_T produced by the agent with the ground-truth occupancy map M^*. In this sense, the paradigm we adopt is the one of knowledge reuse starting from partial knowledge.

6.2 DATASET CREATION

In this section, we describe the newly-proposed dataset that we create to enable research on *Spot the Difference*.

6.2.1 SEMANTIC OCCUPANCY MAP

Given a 3D environment, we place the agent in a free navigable location with heading $\theta = 0°$ (facing eastward). We assume that the input consists of a depth image and a semantic image and that the camera intrinsics K are known. To build the semantic occupancy map (SOM) of an environment, we project each semantic pixel of the acquired scene into a 2-dimensional top-down map: given a pixel with image coordinates (i, j) and depth value $d_{i,j}$, we first recover its coordinates (x, y, z) with respect to the agent position. Then, we compute the corresponding (u, v) pixel in map through an orthographic projection, using the information about the agent position and heading:

$$
\begin{bmatrix} x \\ y \\ z \end{bmatrix} = d_{i,j} K^{-1} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} u \\ v \\ 0 \\ 1 \end{bmatrix} = P_v \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.2}
$$

We perform the same operation after rotating the agent by $\Delta_\theta = 30°$ until we perform a span from $0°$ to $180°$. To cover the whole scene, we repeat this procedure placing the agent at a distance of $0.5m$ from the previous capture point, following the axis directions. The agent elevation is instead kept fixed. During this step, we average the results of subsequent observations of overlapping portions of space.

After the acquisition, we obtain a SOM with C channels, where each pixel corresponds to a $5cm \times 5cm$ portion of space in the 3D environment. For each channel $c \in \{0, ..., C\}$, the map values represent the probability that the corresponding portion of space is occupied by an object of semantic class c.

6.2.2 Multiple SOMs for Each Environment

The SOMs obtained in the previous step can be seen as one possible layout for the corresponding 3D environments. In order to create a dataset with different states (*i.e.* different layouts) of the same environment, instead of manipulating the real-world 3D scenes (changing the furniture position, removing chairs, *etc.*), we propose to modify the SOM to create a set of plausible and different layouts for the environment.

First, we isolate the objects belonging to each semantic category by using an algorithm for connected component labeling [52, 24, 4]. Then, we sample a subset of objects to be deleted from the map and a subset of objects to be re-positioned in a different free location on the map. During sampling, we consider categories that have a high probability of being displaced or removed in the real world and ignore non-movable semantic categories such as *fireplaces*, *columns*, and *stairs*. After this step, we obtain a new SOM representing a possible alternative state for the environment, which could be very different from the one in which the 3D acquisition was taken. Sample manipulated maps can be found in Fig. 6.1 and Fig 6.2.

6.2.3 Dataset Details

To generate alternative SOMs, we start from the Matterport 3D (MP3D) dataset of spaces [30], which comprises 90 different building scans, and is enriched with dense semantic annotations. We consider each floor in the building and compute the SOM for that floor. For each map, we create 10 alternative versions of that same environment. In this step, we discard the floors that have few semantic objects (*e.g.*, empty rooftops) or that are not fully navigable by the agent. As a result, we retain 249 floors belonging to 81 different buildings, thus generating a total of 2490 different semantic occupancy maps for these floors. Finally, we split the dataset into train, validation, and test subsets.

As an additional testbed, we also build a set of out-of-domain maps (13 floors from 5 spaces) taken from the Gibson dataset [149], enriched with semantic annotations from [11] and manipulated as done for the MP3D dataset. For each

SOM, multiple episodes are generated by selecting different starting points. More information about our dataset can be found in Table 6.1.

Semantic Classes Division. The generation of semantic maps for each floor of each scene produces $2001 \times 2001 \times 43$ maps for the MP3D dataset and $961 \times 961 \times 21$ maps for the Gibson dataset. The last channel of every map registers the explorable space, so it is ignored for the creation of the dataset and is concatenated, as it is, to the manipulated map obtained at the end of the semi-automatic dataset creation process.

We divide the semantic channels of the maps depending on the possible actions that can be performed on the connected components in that channel. We identify four types of classes: *No Operation*, *Removal*, *Displacement*, and *Overlap Removal*. A list of semantic categories with their classification is reported at the end of this Chapter in Table 6.4 for the MP3D dataset and in Table 6.5 for the Gibson dataset. *No Operation* classes are left untouched, and correspond to non-movable objects, such as *wall*, *stairs*, and *columns*; the connected component of the *Removal* classes can be removed; those in the *Displacement* classes can be either removed or relocated in other free spaces in the environment; and *Overlap Removal* components are removed if connected components removed or displaced in other channels overlap with them, *e.g.*, if a *sofa* is removed, every instance of *cushion* overlapping with that *sofa* will be removed as well because it is supposed to be on top of it.

Fig. 6.1 and Fig 6.2, we report some examples of manipulated semantic maps with relative difference maps obtained by applying our semi-automatic procedure.

Episode Creation. For the creation of the episodes of our dataset, we use the starting positions of the exploration dataset for MP3D, and of the Point Goal navigation dataset for Gibson Tiny. After the episodes located on floors with few semantic objects or that are not fully navigable by the agent are discarded, we associate one of the alternative versions of the ground-truth semantic map with each episode. We use the same scene partitioning as it is adopted by the existing datasets for embodied exploration and Point Goal navigation on Matterport 3D

6

and Gibson Tiny [30, 149]. For the validation and test splits of the MP3D dataset and the validation split of the Gibson dataset, we create new episodes with random sampled starting positions so that the number of episodes on every floor is at least 10 and fix the number of episodes per floor to a multiple of 10. We report a detailed list of scans, selected floors, and the number of episodes per scan in Tables 6.6, 6.7, 6.8, and 6.9 at the end of this Chapter.

6.3 Proposed Method

Our model for embodied navigation in changing environments comprises three major components: a mapper module, a pose estimator, and a navigation policy (which, in turn, consists of a global policy, a planner, and a local policy). The implementation details of the modules of the architecture are described in Section 3.1. An overview of the proposed architecture is shown in Fig. 6.3 and described in the following section. Although the data we provide is enriched with semantic labels, our agent does not make use of such information directly. This is in line with current architectures for embodied exploration that we choose as competitors.

6.3.1 Mapper

The mapper module takes as inputs an RGB observation φ_t^{rgb} and the corresponding depth image φ_t^d, representing the first-person view of the agent at timestep t, and outputs the agent-centric occupancy map m_t of a $V \times V$ region in front of the camera. Differently from the work described in Chapter 3, each pixel in m_t corresponds to a $25mm \times 25mm$ portion of space and consists of two channels containing the probability of that cell being occupied and explored, respectively. The computed agent-centric occupancy map m_t is then registered in the global occupancy map M_{t-1} coming from the previous timestep to obtain M_t using the pose of the agent (x_t, y_t, θ_t).

Figure 6.2: Generation of alternative states of an environment: original and sample manipulated semantic maps with relative difference maps.
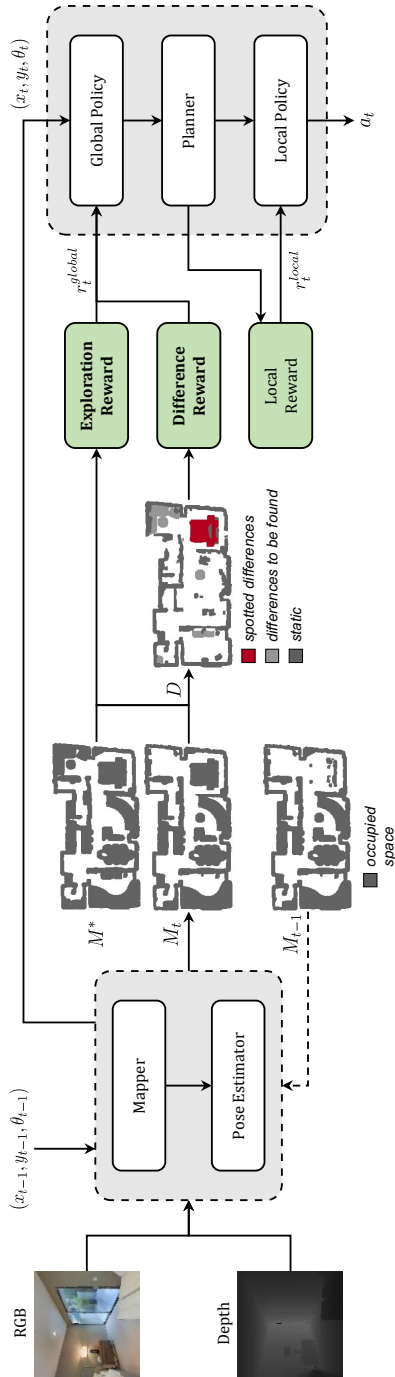
6

Figure 6.3: Overview of the proposed approach for navigation in changing environments.

6.3.2 Pose Estimator

The agent can move across the environment using three actions: *go forward 0.25m*, *turn left 10°*, *turn right 10°*. Since each action may produce a different outcome because of physical interactions with the environment (*e.g.*, bumping into a wall) or noise in the actuation system, the pose estimator is used to estimate the real displacement made at every timestep.

We estimate the agent displacement $(\Delta x_t, \Delta y_t, \Delta \theta_t)$ at timestep t by using two consecutive RGB and depth observations, as well as the agent-centric occupancy maps (m_{t-1}, m_t) computed by the mapper at $t-1$ and t. The actual agent position (x_t, y_t, θ_t) is computed iteratively as:

$$(x_t, y_t, \theta_t) = (x_{t-1}, y_{t-1}, \theta_{t-1}) + (\Delta x_t, \Delta y_t, \Delta \theta_t). \tag{6.3}$$

We assume that the agent starting position is the triple $(x_0, y_0, \theta_0) = (0, 0, 0)$.

6.3.3 Navigation Policy

The sampling of atomic actions for the exploration relies on a three-component hierarchical policy. The first component is the global policy, which samples a long-term global goal on the map. Details on the navigation policy can be found in Section 3.1.3. The global policy outputs a probability distribution over discretized locations of the global map. We sample the global goal from this distribution and then transform it in (x, y) global coordinates. The second component is a planner module, which employs the A* algorithm to decode a local goal on the map. The local goal is an intermediate point, within *0.25m* from the agent, along the trajectory towards the global goal. The last element of our navigation module is the local policy, which decodes the series of atomic actions taking the agent towards the local goal. In particular, the local policy is an RNN decoding the atomic action a_t to execute at every timestep. The reward r_t^{local} given to the local policy is proportional to the reduction in the Euclidean distance d between the agent position and the current local goal.

Following the hierarchical structure, a global goal is sampled every η timesteps.

A new local goal is computed if a new global goal is sampled, if the previous local goal is reached, or if the local goal location is known to be not traversable.

6.3.4 Exploiting Past Knowledge for Efficient Navigation

The global policy is trained using a two-term reward. The first term encourages exhaustive exploration and is proportional either to the increase of coverage [38] or to the increase of anticipated map accuracy as in [115]. Intuitively, the agent strives to maximize the portion of the seen area and thus maximizes the knowledge gathered during exploration. Moreover, since we consider a setting where a significant amount of knowledge is already available to the agent, we add a reward term to guide the agent towards meaningful points of the map. These correspond to the coordinates where major changes are likely to happen.

Given the occupancy map of the agent M_t, the true occupancy map for the same environment M^*, and a time budget of T timesteps for exploration, we aim to minimize the following, for $0 < t \leq T$:

$$D = \sum \mathbb{1}[M_t \neq M^*] \tag{6.4}$$

In other words, we want to maximize the number of pixels in the online reconstructed map M_t that the agent correctly shifts from free to occupied (and vice-versa) during exploration. This leads to the reward term for difference discovery:

$$r_{\text{diff}} = \sum \mathbb{1}[M_t = M^*] - \sum \mathbb{1}[M_{t-1} = M^*]. \tag{6.5}$$

The proposed reward term is designed to encourage navigation towards areas in the map that are more likely to contain meaningful differences (*e.g.*, rooms containing more objects that can be displaced or removed from the scene). Additionally, an agent trained with this reward will tend to avoid difficult spots that are likely to produce a mismatch in terms of the predicted occupancy maps. This is because errors in the mapping phase would result in a negative reward.

To train our model, we combine a reward promoting exploration and a more specific reward on found differences to exploit semantic clues in the environ-

ment:

$$r_t^{global} = \beta_1 r_{\exp} + \beta_2 r_{\text{diff}} \tag{6.6}$$

where $r_{\exp}$ is the reward term encouraging task-agnostic exploration (such as coverage-based or anticipation-based rewards, as described in the next section), and β_1 and β_2 are two coefficients weighing the importance of the two elements.

6.4 EXPERIMENTAL SETUP

6.4.1 EVALUATION PROTOCOL

To evaluate the performance in *Spot the Difference*, we consider three main classes of metrics. First, we consider the percentage of navigable area in the environment seen by the agent during the episode (%AS). Then, we evaluate the percentage of elements that have been correctly detected as changed in the occupancy map (Acc) and the pixel-wise intersection-over-union for the *changed* occupancy map elements (IoU). Besides, we evaluate the task as a two-class problem and compute the IoU score for objects that were added in place of free space (IoU$_+$) and for objects that were deleted during the map creation (IoU$_-$). In addition, to evaluate the performance independently from the exploration capability, we propose to compute the metrics only on the portion of space that the agent actually visited (mAcc, mIoU, mIoU$_+$, and mIoU$_-$).

6.4.2 IMPLEMENTATION DETAILS

We conduct our experiment using Habitat [126], a popular platform for Embodied AI in photo-realistic indoor environments [149, 30]. The agent observations are 128×128 RGB-D images from the environment. The learning algorithm adopted for training is PPO [129]. The learning rate is 10^{-3} for the mapper and 2.5×10^{-4} for the other modules. Every model is trained for $\approx 6.5M$ frames using Adam optimizer [72]. A global goal is sampled every $\eta = 25$ timesteps. The local and global policies are updated, respectively, every η and $20 \times \eta$ timesteps, and the mapper is updated every $4 \times \eta$ timesteps. The size of the local map is $V = 101$, while the global map size is set to $W = 2001$ for MP3D and to

$W = 961$ for Gibson. The reward coefficients $\{\beta_1, \beta_2\}$ are set to $\{1, 10^{-2}\}$ and $\{1, 10^{-1}\}$ when the exploration reward is based on coverage and anticipation reward, respectively. The length of each episode is fixed to $T = 1000$ timesteps.

6.4.3 Competitors and Baselines

We consider the competitors that do not require additional modules to compute the reward signal. The competitors and the variants of the proposed method are evaluated on two different setups: one where the agent position is predicted by the agent (as in Eq. 6.3), and one where it has access to oracle coordinates:

Difference Reward (DR): an exploration policy that maximizes the correctly predicted changes between M and M^*. This corresponds to setting $\beta_1 = 0$ and $\beta_2 = 1$ in Eq. 6.6.

Coverage Reward (CR): an agent that explores the environment with an exploration policy that maximizes the covered area and builds the occupancy map as it goes, as in [115].

Anticipation Reward (AR): an agent that explores the environment with an exploration policy that maximizes the covered area and the correctly anticipated values in the occupancy map built as it goes, from [115].

Occupancy Anticipation (OccAnt): we also compare with the agent presented by Ramakrishnan *et al*. [115] using the available pretrained models, referenced to as *OccAnt*. Note that *OccAnt* was trained on the Gibson dataset for the standard exploration task and without any prior map. Thus, it is not directly comparable with the other methods considered. We include it to gain insights into the performance of an off-the-shelf agent on our task.

Our proposed approach consists of an agent trained with the combination of the difference reward with the coverage reward (*CR+DR*) or with the anticipation reward (*AR+DR*).

	%AS	Acc	IoU$_+$	IoU$_-$	IoU	mAcc	mIoU$_+$	mIoU$_-$	mIoU
	Estimated Localization								
OccAnt	52.1	26.2	13.4	6.1	11.5	51.1	19.1	8.3	15.8
DR	49.4	29.3	15.3	8.7	13.9	59.7	23.1	11.9	20.2
AR	43.8	30.6	19.7	12.9	18.8	72.5	36.8	18.4	32.7
CR	**53.2**	33.1	18.1	9.6	16.1	65.2	26.4	12.7	22.6
AR+DR	51.4	34.5	20.9	12.0	19.3	71.5	33.9	16.2	30.0
CR+DR	52.3	**37.8**	**24.2**	**14.8**	**22.7**	**76.2**	**39.1**	**19.8**	**34.8**
	Oracle Localization								
OccAnt	49.0	35.6	26.5	16.1	24.8	77.8	49.2	23.6	43.2
DR	48.6	37.4	27.2	18.4	26.5	80.1	49.8	27.4	45.8
AR	43.6	32.5	23.2	17.5	23.0	78.7	47.5	26.7	44.5
CR	**52.8**	39.2	29.6	18.8	28.0	78.5	51.0	26.6	45.7
AR+DR	51.4	37.8	27.3	18.0	26.2	79.3	48.9	25.8	44.4
CR+DR	51.8	**40.3**	**29.2**	**19.2**	**28.1**	**82.1**	**50.4**	**26.9**	**46.2**

Table 6.2: Experimental results on MP3D test set. The agent incorporating the proposed reward term for discovered differences outperforms the competitors on the main metrics for the novel Spot the Difference task. Our model CR+DR achieves the best results on all the metrics except for %AS.

6.5   Experimental Results

Results on MP3D dataset. As a first testbed, we evaluate the different agents on the MP3D *Spot the Difference* test set. We report the results for this experiment in Table 6.2.

We observe that the agent combining a reward based on coverage and our reward based on the differences in the environment (*CR+DR*) performs best on all the pixel-based metrics and places second in terms of percentage of seen area. It is worth noting that, even if the results in terms of the area seen are not as high as the ones obtained by the *CR* agent, the addition of our Difference Reward helps the agent to focus on more relevant parts, and thus, it can discover more substantial differences. Additionally, predictions are more accurate and more precise, as indicated by the 4.7% and 6.6% improvements in terms of Acc and IoU with respect to the *CR* competitor. Instead, a reward based on differences alone is not
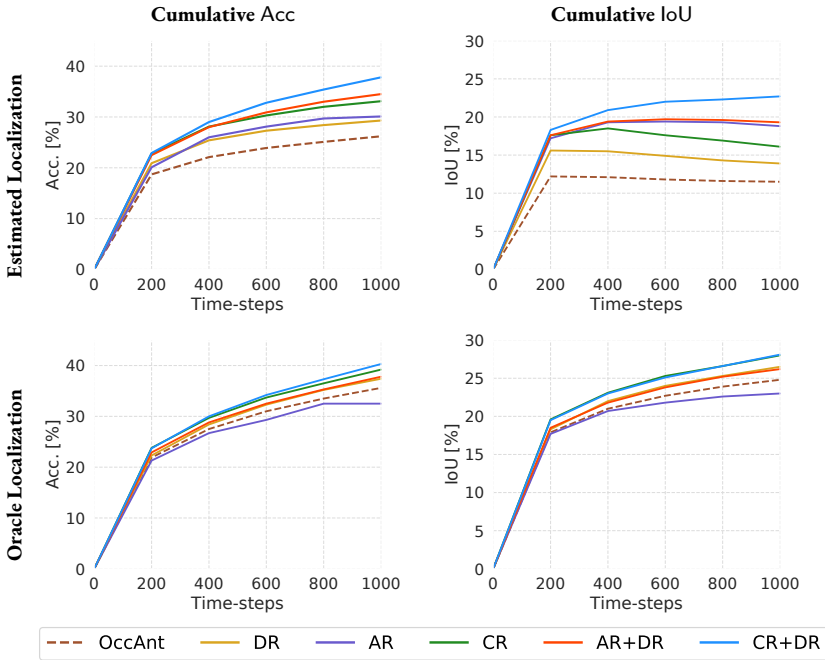
Figure 6.4: Value of accuracy and IoU for the different models at varying timesteps on the MP3D test set.

sufficient to promote good exploration. In fact, although the *DR* agent outperforms the *CR* and *AR* agents on some metrics, our reward alone does not provide as much improvement as when combined with rewards encouraging exploration (as for *CR+DR* and *AR+DR*).

Even in the oracle localization setup, the *CR+DR* agent achieves the best results. Interestingly, the gap with the *CR* agent decreases to 1.1% and 0.1% in terms of Acc and IoU, respectively. This is because our *CR+DR* agent learns to sample trajectories that can be performed more efficiently and without accumulating a high positioning error. For this reason, the performance boost given by the oracle localization is lower. For both setups, our *CR+DR* agent outperforms the state-of-the-art *OccAnt* agent for exploration on all the metrics.

Finally, in Fig. 6.4, we plot different values of Acc and IoU over different timesteps during the episodes. This way, we can evaluate the whole exploration

	%AS	Acc	IoU$_+$	IoU$_-$	IoU	mAcc	mIoU$_+$	mIoU$_-$	mIoU
	Estimated Localization								
OccAnt	86.2	49.8	11.9	7.2	10.4	58.0	12.3	7.5	10.8
DR	**86.2**	53.2	13.2	8.5	11.7	63.7	13.9	8.8	12.3
AR	75.3	51.5	21.4	16.6	20.4	72.7	25.8	17.3	23.3
CR	85.9	57.6	16.7	11.9	15.4	71.3	18.6	12.3	16.7
AR+DR	83.4	58.7	20.0	14.9	19.0	75.8	23.0	15.6	21.1
CR+DR	82.1	**60.1**	**24.0**	**19.0**	**23.1**	**78.5**	**27.8**	**19.9**	**25.9**
	Oracle Localization								
OccAnt	81.6	60.1	**32.1**	21.2	29.2	78.7	**39.6**	22.2	**34.1**
DR	**86.1**	65.2	30.1	24.1	29.9	81.1	36.0	25.2	33.8
AR	74.1	53.8	27.9	21.9	27.2	77.0	35.4	23.5	32.7
CR	84.0	62.2	30.6	22.1	28.8	79.5	36.1	23.3	32.8
AR+DR	83.2	63.2	29.6	23.8	29.1	**81.6**	35.8	25.1	33.7
CR+DR	82.6	63.8	30.3	**24.1**	29.5	**81.6**	36.1	**25.5**	34.0

Table 6.3: Experimental results on Gibson validation set. Our model CR+DR achieves the best results in the setting of estimated localization while being competitive when using oracle localization.

trend, and not only its final point. We can observe that the proposed models incorporating the difference reward outperform the competitors. In particular, the *CR+DR* agent scores first by a significant margin. The performance gap can be noticed even in the first half of the episode and tends to grow with the number of steps.

RESULTS ON GIBSON DATASET. The environments from the Gibson dataset [149] are generally smaller than those in MP3D, and thus, they can be explored more easily and exhaustively. We report the results for this experiment in Table 6.3. Also in this experiment, the *CR+DR* agent performs best on all the metrics but the percentage of the area seen. Although *CR+DR* explores 3.8% of the environment less than the *CR* agent, it still overcomes the competitor by 2.5% and 7.7% in terms of Acc and IoU. The *AR+DR* agent is the second-best in terms of Acc. The *OccAnt* agent, instead, is competitive in terms of area seen but achieves low Acc and IoU metrics. In the setting of exploration with oracle
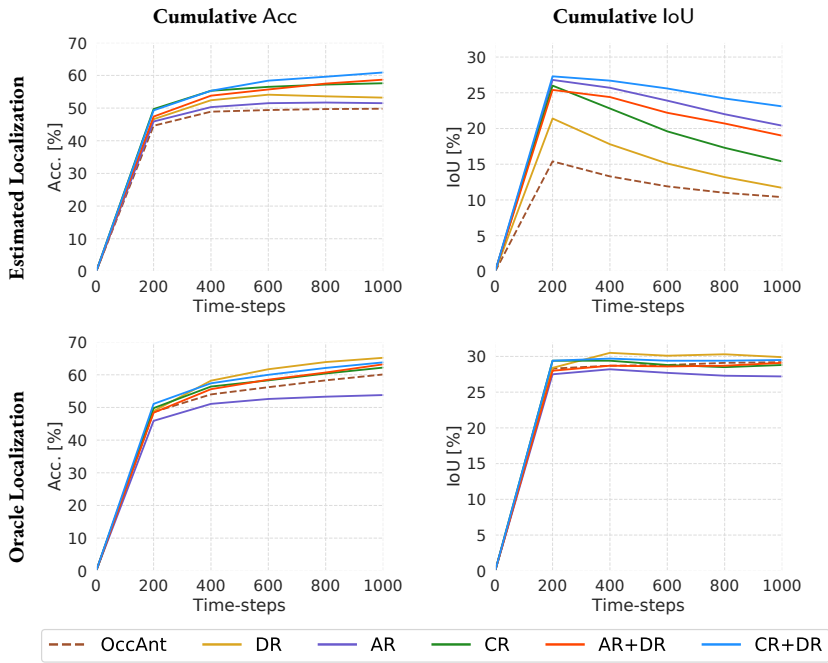
6

Figure 6.5: Value of accuracy and IoU for the different models at varying timesteps on the Gibson validation set.
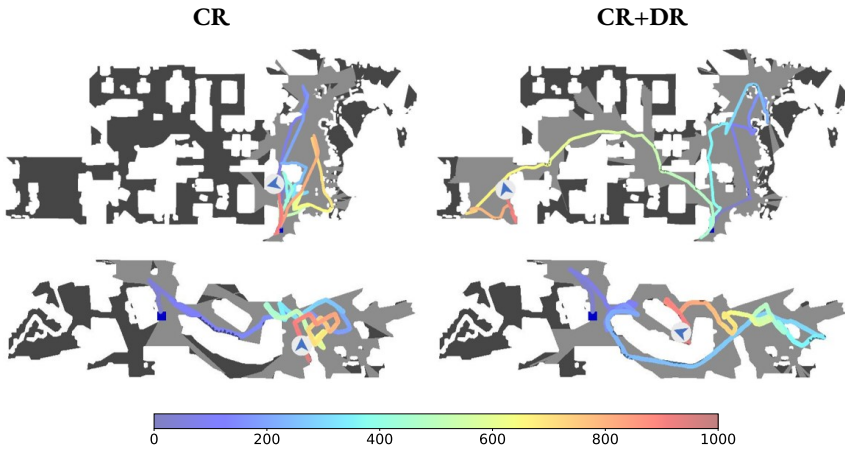

Figure 6.6: Exploration trajectories of the CR and CR+DR agents on sample MP3D test episodes.

localization, the agent using only the proposed difference reward (*DR*) performs the best on almost all the metrics. We can conclude that, for small environments, and given an optimal localization system, our reward alone is sufficient to surpass the competitors on *Spot the Difference*.

In Fig. 6.5, we report the plots of different values of Acc and IoU for the MP3D and Gibson validation sets, respectively. In these plots, we show how Acc and IoU vary at different timesteps during the episodes for the various methods.

QUALITATIVE RESULTS. In Fig. 6.7, we report some qualitative results. Starting from the left-most column, we present the starting map given to the agent as the episode begins, the results achieved by the *CR* agent, those of the proposed *CR+DR* agent, and the ground-truth map. The differences that the agents have correctly identified during the episode are highlighted in red. As it can be seen, the *CR+DR* agent can identify more differences than the *CR* counterpart, even in small environments (top row). As the size of the environments grows (bottom three rows), the performance gap increases and the *CR+DR* agent outperforms its competitor. Moreover, we report sample exploration trajectories for both the CR and the CR+DR agents with estimated localization in Fig. 6.6. These confirm the competitive exploration capabilities of our proposed agent.

6.6 FUTURE DIRECTIONS

Our method exploits outdated information about the current environment to improve the exploration capabilities of the agent. However, the focus of this work is on pure occupation, ignoring semantic information. For future work, we expect to include semantic reasoning in the agent's pipeline, assuming that additional information could boost the performance. With the proposed dataset, we enable a series of possible embodied tasks that imply dynamic environments and incorporate available past knowledge.

6

Starting Map **CR** **CR+DR** **Ground-truth Map**

unchanged spotted differences differences to be found

Figure 6.7: Qualitative results comparing the performances of the CR and CR+DR agents for different episodes.

MP3D		
Index	Category	Action
0	Void	No Operation
1	Wall	No Operation
2	Floor	No Operation
3	Chair	Displacement
4	Door	No Operation
5	Table	Displacement
6	Picture	No Operation
7	Cabinet	Removal
8	Cushion	Overlap Removal
9	Window	No Operation
10	Sofa	Displacement
11	Bed	Displacement
12	Curtain	No Operation
13	Chest of Drawers	Displacement
14	Plant	Displacement
15	Sink	Empty
16	Stairs	No Operation
17	Ceiling	No Operation
18	Toilet	Removal
19	Stool	Displacement
20	Towel	Overlap Removal
21	Mirror	No Operation
22	TV Monitor	Removal
23	Shower	Removal
24	Column	No Operation
25	Bathtub	Removal
26	Counter	Removal
27	Fireplace	No Operation
28	Lighting	No Operation
29	Beam	No Operation
30	Railing	No Operation
31	Shelving	Removal
32	Blinds	No Operation
33	Gym Equipment	Displacement
34	Seating	Removal
35	Board Panel	No Operation
36	Furniture	Displacement
37	Appliances	Removal
38	Clothes	Overlap Removal
39	Objects	Overlap Removal
40	Misc	Overlap Removal
41	Unlabeled	No Operation

Table 6.4: MP3D semantic categories per channel index.

Gibson		
Index	Category	Action
0	Chair	Displacement
1	Couch	Displacement
2	Potted Plant	Removal
3	Bed	Displacement
4	Toilet	Removal
5	TV	Removal
6	Dining Table	Displacement
7	Oven	Removal
8	Sink	Removal
9	Refrigerator	Removal
10	Book	Overlap Removal
11	Clock	Removal
12	Vase	Removal
13	Cup	Overlap Removal
14	Bottle	Overlap Removal
15	Bench	Removal
16	Appliances	Removal
17	Objects	Overlap Removal
18	Misc	Overlap Removal
19	Void	No Operation

Table 6.5: Gibson semantic categories per channel index.

Gibson Validation		
Scan	Floors	# Episodes
Wiconisco	1,2	90
Corozal	0,2,4	90
Collierville	0,1,2	80
Markleeville	0,1	90
Darden	0,1,2	100
Total: 5	13	450

Table 6.6: Gibson validation scans and floors, with relative number of episodes for *Spot the Difference*.

MP3D Train		
Scan	**Floors**	**# Episodes**
HxpKQynjfin	0	81967
gTV8FGcVJC9	0,1,2,3,4,6,10,11	77186
29hnd4uzFmX	0,1,2,3	81967
5LpN3gDmAk7	0,1,2,3	81885
SN83YJsR3w2	0,1,2,3,7,8,10,12	81438
VzqfbhrpDEA	0,1,3,6	81641
D7N2EKCX4Sj	0,1,2,3,5,6	81830
5q7pvUzZiYa	0,1,2,3,4	81967
ac26ZMwG7aT	0,1	81967
r47D5H71a5s	0,1	81965
Pm6F8kyY3z2	0	81967
8WUmhLawc2A	0,1,2	81967
82sE5b5pLXE	0,1,2	80682
mJXqzFtmKg4	0,1,2	81967
i5noydFURQK	0,1	81120
V2XKFyX4ASd	0,1,2,3,4,5,7	81129
759xd9YjKW5	0,1,2,3	81913
r1Q1Z4BcV1o	0	81812
S9hNv5qa7GM	0,1	81967
1LXtFkjw3qL	0,1,2,3,4,5,6	81967
PuKPg4mmafe	0	81940
EDJbREhghzL	0,1,3	64755
ur6Fq6Qu1A	0,1	81967
B6ByNegPMKs	0	81951
b8cTxDM8gDG	0,1,2,8,11	73307
17DRP5sb8fy	0	81967
YmJkqBEsHnH	0	80780
ULsKaCPVFJR	0,1,2	81967
XcA2TqTSSAj	0,2,3,5,6,8,9,11,12	60679
sKLMLpTHeUy	0,1,2,4	79736
ZMojNkEp431	0,1,2	81967
e9zR4mvMWw7	0,1,2	80193
JeFG25nYj2p	0,1	81967
uNb9QFRL6hY	1,4,5,6	59613
p5wJjkQkbXX	0,1,2,3	81967
Vvot9Ly1tCj	0,3	78115
E9uDoFAP3SH	0,1,5,6	81914
qoiz87JEwZ2	0,1,2,3	81967
VFuaQ6m2Qom	0,1,2,4,5,6	81758
VLzqgDo317F	0,1,2	81396
kEZ7cmS4wCh	0,1,2,3,7	69135
7y3sRwLe3Va	0,1,2,5	81386
VVfe2KiqLaN	0,1,2	81967
2n8kARJN3HM	0,1,2,4	81076
PX4nDJXEHrG	0,1,2,3,4,5	79151
Uxmj2M2itWa	0,1,3,4	49942
pRbA3pwrgk9	0,2,3,7,9,11	53295
cV4RVeZvu5T	0,1,2,3	81038
sT4fr6TAbpF	0	81625
GdvgFV5R1Z5	0	81967
JF19kD82Mey	0,1,2	81927
JmbYfDe2QKZ	0,1	81489
s8pcmisQ38h	0,1,2	80428
1pXnuDYAj8r	0,1,2,5	81901
jh4fc5c5qoQ	0,1,2	81967
vyrNrziPKCB	0,1,3,4,7	81388
aayBHfsNo7d	0,1,2	81693
rPc6DW4iMge	0,1,3,4	80296
Total: 58	207	4581881

Table 6.7: MP3D train scans and floors.

MP3D Validation		
Scan	**Floors**	**# Episodes**
2azQ1b91cZZ	0,1	40
8194nk5LbLH	0	40
EU6Fwq7SyZv	0	30
QUCTc6BB5sX	1	20
TbHJrupSAjP	0,1,2	30
Z6MFQCViBuw	0	40
oLBMNvg9in8	0,1,2,3	50
x8F5xyUWy9e	0,1	30
zsNo4HB9uLZ	0	40
Total: 9	16	320

Table 6.8: MP3D validation scans and floors, with relative number of episodes for *Spot the Difference*.

MP3D Test		
Scan	**Floors**	**# Episodes**
2t7WUuJeko7	0	50
5ZKStnWn8Zo	0,1	50
RPmz2sHmrrY	0	50
UwV83HsGsw3	0,1,2,3	50
WYY7iVyf5p8	0,2	30
YFuZgdQ5vWj	1	10
YVUC4YcDtcY	0	50
fzynW3qQPVF	0,1	50
jtcxE69GiFV	0,1	40
pa4otMbVnkk	0,1	50
q9vSo1VnCiC	0	50
rqfALeAoiTq	0,2	20
wc2JMjhGNzB	0,1	50
yqstnuAEVhm	0,1,2	60
Total: 14	26	610

Table 6.9: MP3D test scans and floors, with relative number of episodes for *Spot the Difference*.

7

Out of the Box:
Embodied Navigation in the Real World

FollowING the previous chapters where the contributions are mainly related to the work in simulation devising novel methods or tasks for Embodied AI. In this chapter and in the following one, our efforts are targeting real-world aspects of Embodied AI. Embodied AI has recently attracted a lot of attention from the vision and learning communities. Nevertheless, the majority of current research focuses on the creation of rich and complex architectures that are trained in simulation, using large amounts of data. Thanks to powerful simulating platforms [44, 126, 149], the Embodied AI community could achieve nearly perfect results on tasks such as Point Goal navigation (PointNav) [147]. However, current research is still in the first mile of the race for the creation of intelligent and autonomous agents. Naturally, the next milestones involve bridging the gap between simulated platforms (in which the training takes place) and the real world [67]. In this work, we aim to design a robot that can navigate in

---

This Chapter is related to the publication "R. Bigazzi *et al.*, Out of the Box: Embodied Navigation in the Real World, CAIP 2021" [21]. See the list of Publications on page 133 for more details.

97

unknown, real-world environments [29].

We ask ourselves a simple research question: *can the agent transfer the skills acquired in simulation to a more realistic setting?* To answer this question, we devise a new experimental setup in which models learned in simulation are deployed on a LoCoBot [88]. Previous work on Sim2Real adaptability from the Habitat simulator [126] has focused on a setting where the real-world environment was matched with a corresponding simulated environment to test the Sim2Real metric gap. To that end, Kadian *et al*. [67] carry on a 3D acquisition of the environment specifically built for robotic experiments. Here, we assume a setting in which the final user cannot count on the technology/expertise required to make a 3D scan. This experimental setup is more challenging for the agent, as it cannot count on semantic priors on the environment acquired in simulation. Moreover, while [67] employs large boxes as obstacles, our testing scene contains real-life objects with complicated shapes such as desks, office chairs, and doors.

Our agent builds on a recent model proposed by Ramakrishnan *et al*. [115] for the PointNav task. As a first step, we research the optimal setup to train the agent in simulation. We find out that default options (tailored for simulated tasks) are not optimal for real-world deployment: for instance, the simulated agents often exploit imperfections in the simulator physics to slide along the walls. As a consequence, deployed agents tend to get stuck when trying to replicate the same sliding dynamic. By enforcing a more strict interaction with the environment, it is possible to avoid such shortcomings in the locomotor policy. Secondly, we employ the software library PyRobot [97] to create a transparent interface with the LoCoBot: thanks to PyRobot, the code used in simulation can be seamlessly deployed on the real-world agent by changing only a few lines of code. Finally, we test the navigation capabilities of the trained model on a real scene: we create a set of navigation episodes in which goals are defined using relative coordinates. While previous tests were mainly made in robot-friendly scenarios (often consisting of a single room), we test our model, which we call LoCoNav, in a realistic type of environment: obstacles are represented by common office furniture such as desks, chairs, cupboards; the floor is uneven as there are gaps between floor tiles that make actuation noisy and very position-dependent, and there are mul-

tiple rooms that must be accessed through doorways (Fig.7.1). Thanks to our experiments, we show that models trained in simulation can adapt to real unseen environments. By making our code and models publicly available, we hope to motivate further research on Sim2Real adaptability and deployment in the real world of agents trained on the Habitat simulator. Our code and models are available publicly*.

7.1 Real-World Navigation with Habitat

In this section, we describe our out-of-the-box navigation robot. First, we describe the baseline architecture and its training procedure that takes place in the Habitat simulator [126]. Then we present our LoCoNav agent, which builds upon the baseline and implements various modules to enable real-world navigation.

7.1.1 Baseline Architecture

We draw inspiration from the occupancy anticipation agent [115] to design our baseline architecture. The model consists of three main parts: a mapper, a pose estimator, and a hierarchical policy, which we describe in the following. More details on their implementation is contained in Section 3.1.

Mapper. The mapper is responsible for producing an occupancy map of the environment, which is then employed by the agent as an auxiliary representation during navigation. Following the work presented in Chapters 3, 5, and 6, we use two different types of maps at each timestep t: the local map m_t that depicts the portion of the environment immediately in front of the agent, and the global map M_t that captures the area of the environment already visited by the agent. The global map of the environment M_t is blank at $t = 0$ and it is built in an incremental way. Each map has two channels, identifying the free/occupied and the explored/unexplored space, respectively; each pixel contains the state of a 5cm

---

*https://github.com/aimagelab/LoCoNav

$\times$ 5cm area. The mapper module takes as input the RGB and depth observations $(\varphi_t^{rgb}, \varphi_t^d)$ at time t and produces the agent-centric local map $m_t \in [0, 1]^{2 \times V \times V}$.

As described in previous chapters, at each timestep m_t is registered to the global map $M_t \in [0, 1]^{2 \times W \times W}$, with $W > V$, using the agent's position and heading in the environment (x_t, y_t, θ_t).

POSE ESTIMATOR. While the agent navigates towards the goal, the interactions with the environment are subject to noise and errors, so that, for instance, the action *go forward* 25*cm* might not result in a real displacement of 25cm. That could happen for a variety of reasons: bumping into an obstacle, slipping on the terrain, or simple actuation noise. The pose estimator is responsible for avoiding such positioning mistakes and keeps track of the agent pose in the environment at each timestep t. This module computes the relative displacement $(\Delta x_t, \Delta y_t, \Delta \theta_t)$ caused by the action selected by the agent at time t. It takes as input the RGB-D observations $(\varphi rgbr_t, \varphi_t^d)$ and $(\varphi_{t-1}^{rgb}, \varphi_{t-1}^d)$ retrieved at time t and $t-1$, and the egocentric maps m_t and m_{t-1}, and outputs the displacement $(\Delta x_t, \Delta y_t, \Delta \theta_t)$ The estimated pose of the agent at time t is given by:

$$(x_t, y_t, \theta_t) = (x_{t-1}, y_{t-1}, \theta_{t-1}) + (\Delta x_t, \Delta y_t, \Delta \theta_t). \tag{7.1}$$

NAVIGATION POLICY. The baseline navigation policy is defined by a hierarchical design. The highest-level component of our policy is the global policy. The global policy selects a long-term goal on the global map, which we call global goal g_t. A new global goal is sampled every η timesteps during training and is set to the navigation goal during deployment and test. The middle-level component of our hierarchical policy is the planner. After the global goal is set, an A* planner decodes the next local goal within 0.25m from the agent and on the trajectory toward the global goal. A new local goal is sampled if at least one of the following three conditions verifies: a new global goal is sampled by the global policy, the previous local goal is reached, or the local goal is known to be in an occupied area. Finally, the local policy performs the low-level navigation and decodes the series of actions to perform. The actions available to the agents are *go forwards* 25*cm* and *turn* 15°. The local policy samples an atomic action a_t at each timestep t.
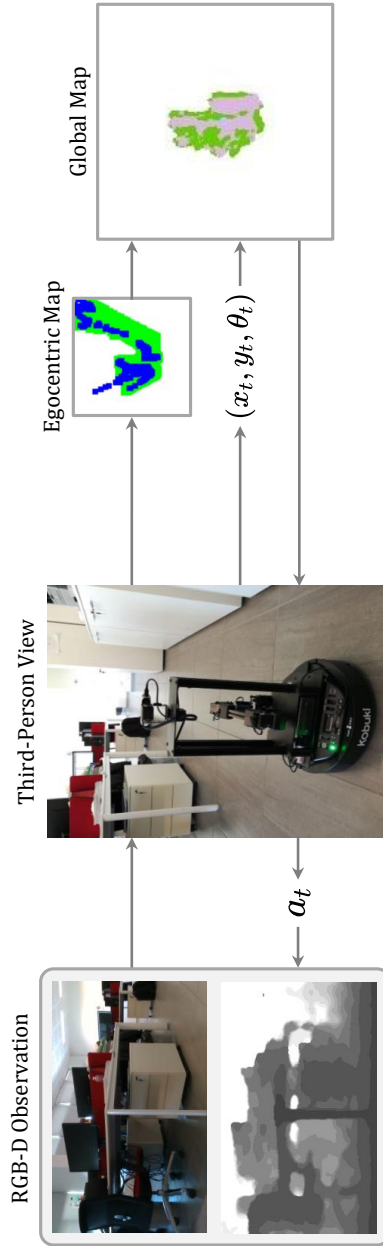
7

Figure 7.1: We deploy a state-of-art navigation architecture on a LoCoBot and test it in a realistic, office-like environment. Our model exploits egocentric and global occupancy maps to plan a route towards the goal.

7.1.2 TRAINING IN SIMULATION

The baseline architecture described in the previous lines is trained in simulation using Habitat [126] and 3D scans from the Gibson dataset of spaces [149]. The mapper is trained with a binary cross-entropy loss using the ground-truth occupancy maps of the environment, obtained as described in [115]. The navigation policy is trained using reinforcement learning. We choose PPO [129] as training algorithm. The global policy receives a reward signal equal to the increase in terms of anticipated map accuracy [115]:

$$r_t^{global} = \text{Acc}(M_t, M^*) - \text{Acc}(M_{t-1}, M^*), \tag{7.2}$$

where M_t and M_{t-1} represent the global occupancy maps computed at time t and $t-1$ respectively, and $^* \in [0,1]^{2 \times W \times W}$ is the ground-truth global map. The map accuracy is defined as:

$$\text{Acc}(M, M^*) = \sum_{i=1}^{W^2} \sum_{j=1}^{2} \mathbb{1}[M_{ij} = M_{ij}^*], \tag{7.3}$$

where $\mathbb{1}[\cdot]$ is an indicator function that returns one if the condition $[\cdot]$ is true and zero otherwise. The local policy is trained using a reward that encourages the decrease in the euclidean distance between the agent and the local goal while penalizing collisions with obstacles:

$$r_t^{local} = d_t - d_{t-1} - \alpha \cdot bump_t, \tag{7.4}$$

where d_t and d_{t-1} are the euclidean distances to the local goal at times t and $t-1$, $bump_t \in \{0,1\}$ identifies a collision at time t and α regulates the contributions of the collision penalty. The training procedure described in this section exploits the experience collected throughout 6.5 million exploration frames.

	Height	RGB FoV	Depth FoV	Depth Range	Obst. Height Thr.
Default for Simulation	1.25	**H**: 90, **V**: 90	**H**: 90, **V**: 90	[0.0, 10.0]	[0.2, 1.5]
LoCoNav	0.60	**H**: 70, **V**: 90	**H**: 57, **V**: 86	[0.0, 5.00]	[0.3, 0.6]

Table 7.1: List of hyperparameters changes for Sim2Real transfer.

7.1.3 LoCoNav: Adapting for Real World

The baseline architecture described above is trained in simulation and achieves state-of-art results on embodied exploration and navigation [115]. The reality, however, poses some major challenges that need to be addressed to achieve good real-world performances. For instance, uneven ground might give rise to errors and noise in the actuation phase. To overcome this and other discrepancies between simulated and real environments, we design LoCoNav: an agent that leverages the availability of powerful simulating platforms during training but is tailored for real-world use. In this section, we describe the main characteristics of the LoCoNav design. We deploy our architecture on a LoCoBot [88] and use PyRobot [97] for seamless code integration.

Prevent your Agent from Learning Tricks. All simulations are imperfect. One of the main objectives when training an agent for real-world use in simulation is to prevent it from learning simulator-specific tricks instead of the basic navigation skills. During training, we observed that the agent tends to hit the obstacles instead of avoiding them. This behavior is given by the fact that the simulator allows the agent to slide towards its direction even if it is in contact with an obstacle as if there were no friction at all. Unfortunately, this ideal situation does not fit the real world, as the agent needs to actively rotate and head towards a free direction every time it bumps into an obstacle. To replicate the realistic *sticky* behavior of surfaces, we check the $bump_t$ flag before every step. If a collision is detected, we prevent the agent from moving forward. As a result, our final agent is more cautious about any form of collision.

Sensor and Actuation Noise. Another important discrepancy between simulation and real world is the difference in the sensor and actuation systems. Luckily, the Habitat simulator allows for great customization of input-output

dynamics, thus being very convenient for our goal. In order to train a model that is more resilient to the camera noise, we apply a Gaussian Noise Model on the RGB observations and a Redwood Noise Model [39] on the depth observations. Unfortunately, the LoCoBot RealSense camera still presents various artifacts and regions with missing depth values. For that reason, we need to restore the observation retrieved from the depth camera before using it in our architecture. To that end, we apply the hole-filling algorithm described in [140], followed by the application of a median filter.

Regarding the actuation noise, we find out that the use of the incremental pose estimator (employed in the occupancy anticipation model and described in our baseline architecture) is not optimal, especially when combined with the actuation noise typical of real-world scenarios. Luckily, we can count on more precise and reliable information coming from the LoCoBot actuation system. By checking the actual rotation of each wheel at every timestep, the robot can update its position step by step. We adapt the odometry sensor of the LoCoBot platform to be compliant with our architecture. To that end, the pose returned by the sensor is converted by resetting it with respect to its state at the beginning of the episode. We name $\hat{\omega}_0 = (\hat{x}_0, \hat{y}_0, \hat{\theta}_0)$ the coordinate triplet given by the odometry sensor at $t = 0$. We then define:

$$\mathbf{A} = \begin{pmatrix} \mathbf{R}_0 & \mathbf{t}_0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos \hat{\theta}_0 & -\sin \hat{\theta}_0 & \hat{x}_0 \\ \sin \hat{\theta}_0 & \cos \hat{\theta}_0 & \hat{y}_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.5}$$

Let us define $\tilde{\omega}_t$ as the augmented position vector $(\hat{x}_t, \hat{y}_t, 1)$ containing the agent position at each step t. We compute the relative position of the robot as:

$$\bar{\omega}_t = \mathbf{A}^{-1}\tilde{\omega}_t, \qquad \theta_t = \hat{\theta}_t - \hat{\theta}_0 \tag{7.6}$$

where $\bar{\omega}_t = (x_t, y_t, 1)$ and θ_t relatively contain the position and the orientation of the agent after the conversion to episode coordinates. In fact, the relative position and heading are given by $\omega_t = (x_t, y_t, \theta_t)$.

Note that, for $t = 0$, $\omega_0 = (x_0, y_0, \theta_0) = (0, 0, 0)$.

Hyperparameters. Finally, we noticed that typical hyperparameters employed in simulation do not match the real robot characteristics. For instance, the camera height is set to 1.25m in previous works, but the RealSense camera on the LoCoBot is placed only 0.6m from the floor. During the adaptation to the real-world robot, we change some hyperparameters to align the observation characteristics of the simulated and the real world and to match real robot constraints. These parameters are listed in Table 7.1.

7.2 Experiments

7.2.1 Testing Setup

We run multiple episodes in the real environment, in which the agent needs to navigate from a starting point A to a destination B. The goal is specified by using relative coordinates (in meters) with respect to the agent's starting position and heading. Although the agent knows the position of its destination, it has no prior knowledge of the surrounding environment. Because of this, it cannot immediately plan a direct route to the goal and must check for obstacles and walls before stepping ahead. After each run, we reset the agent memory so that it cannot retain any information from previous episodes. We design five different navigation episodes that take place in three different office rooms and the corridor connecting them (Fig. 7.3). For each episode, we run different trials with different configurations: obstacles are added/moved, or people are sitting/standing in the room. In total, we run 50 different experiments, resulting in more than 10 hours of real-world testing.

7.2.2 Evaluation Protocol

An episode is considered successful if the agent sends a specific *stop* signal within $0.2m$ from the goal. This threshold corresponds to the radius of the robot base. For every navigation episode, we also track the number of steps and the time required to reach the goal. Since the absolute number of steps is not comparable among different episodes, we ask human users to control the LoCoBot and com-

Path	Length [m]	Time [s]	Steps
A	3.80	124	23
B	6.75	239	45
C	5.95	223	43
D	6.55	217	42
E	4.20	227	33

Table 7.2: Path-specific information, as obtained with human supervision.

Figure 7.2: Layout of the navigation episodes.

plete each navigation path via a remote interface (we report human performance in Table 7.2). We then normalize these measures using this information so that results close to 1.0 indicate human-like performances. We provide absolute and normalized length and time for each episode, as well as the popular SPL metric (Success rate weighted by inverse Path Length). We employ a slightly modified version of the SPL, in which the normalization is made basing on the number of steps and not on the effective path length to penalize purposeless rotations. Additionally, we set a boolean flag for each episode that signals whether the robot has bumped into an obstacle, and we report the average bump rate (BR). We also report the hard failure rate (HFR) as the fraction of episodes terminated if the agent gets stuck and cannot proceed, or if the episode length exceeds the limit of 300 steps.

Path	SR↑	SPL↑	HFR↓	BR↓	Abs. Steps	Norm. Steps↑	Abs. Time	Norm. Time↑
A	1.0	0.718	0.0	0.30	32.70±1.73	0.717±0.033	176.11±10.39	0.718±0.031
B	0.8	0.711	0.10	0.22	51.67±1.72	0.880±0.027	273.70±8.24	0.879±0.030
C	0.5	0.205	0.10	0.78	123.44±10.66	0.374±0.034	631.15±50.09	0.372±0.036
D	0.5	0.318	0.10	0.89	65.67±3.90	0.645±0.037	344.00±20.08	0.657±0.038
E	0.2	0.060	0.40	1.00	135.17±29.97	0.290±0.049	722.76±162.01	0.38±0.066
Mean	0.6	0.402	0.14	0.60	-	0.608±0.036	-	0.617±0.034

Table 7.3: Navigation results. Numbers after ± denote the standard error of the mean.

7.2.3 REAL-WORLD NAVIGATION

In this experiment, we test our robot on five different realistic navigation paths (Fig. 7.2). We report the numerical results for these experiments in Table 7.3, and we plot the main metrics in Fig. 7.3 to allow for a better visualization of navigation results across different episodes. When a path is contained in a single room (A), the agent achieves optimal results, as it always stops within the success threshold from the goal. The number of steps is slightly higher than the minimum required by the episode (33 instead of 23), but this overhead is necessary as the agent must rotate and "look around" to build a decent map of the surrounding before planning a route to the goal. Paths that involve going outside the room and navigating different spaces (B, C, D, E) are fairly complicated, but the agent can generally terminate the episode without hard failures. When the shortest path to the goal leads to a wall or a dead-end, the agent needs to find an alternative way to circumvent this obstacle (*e.g.* a door). This leads to a higher episode length because the robot must dedicate some time to general exploration of the surroundings. Finally, we find out that the most challenging scenario for our LoCoNav is when reaching the goal requires to get out of a room and then enter a door immediately after, on the same side of the corridor (as in E). Since the robot sticks to the shortest path, the low parallax prevents it from identifying the second door correctly. Even in these cases, a bit of general exploration helps to solve the problem.

Figure 7.3: Comparison of the main navigation metrics on different episodes.

7.2.4 DISCUSSION

Overall, our experimental setup provides a challenging test-bed for real-world robots. We find out that failures are due to two main issues. First, when the agent must navigate to a different room, it has no access to a map representing the general layout of the environment. This prevents the robot from computing a general plan to reach the long-term goal and forces it to explore the environment before proceeding. If a map was given to the agent, this problem would have been greatly alleviated. A second problem arises when the goal is close in terms (x, y) coordinates but is physically placed in an adjacent room. To solve this problem, one could decompose the navigation between rooms in a multi-goal problem where neighboring nodes are closer. In this way, it is possible to reduce a complex navigation episode in simpler sub-episodes (like A or B), in which our agent has proved to be successful.

8

Embodied Navigation at the Art Gallery

I n this last chapter, we contribute to the literature on Embodied AI by intro-
ducing a novel dataset for embodied navigation. In recent years, Embodied
AI has benefited from the introduction of rich datasets of 3D spaces and
new tasks, ranging from exploration to Point Goal or Image Goal Navigation
[30, 149]. Such availability of 3D data allows the training and deployment of
modular embodied agents, also thanks to powerful simulation platforms [126].
Despite the high number of available spaces, though, the topology and nature
of the different scenes have low variance. Indeed, many environments represent
apartments, offices, or houses. In this work, we take a different path and collect
and introduce the 3D space of an art gallery.

Current agents for embodied exploration feature a modular approach [32,
115]. While the agents are trained for embodied exploration using deep rein-
forcement learning, this hierarchical paradigm allows for great adaptability on
downstream tasks. Hence, models trained to explore the Gibson dataset can
solve Point Goal navigation with satisfactory accuracy under the appropriate hy-

---

This Chapter is related to the publication "R. Bigazzi *et al*., Embodied Navigation at the Art
Gallery, ICIAP 2021" [19]. See the list of Publications on page 133 for more details.

potheses. Furthermore, accurate and realistic simulating platforms such as Habitat [126] facilitate the deployment in the real world of the trained agents [67]. While agent architectures and simulating platforms are possible sources of improvement, there is a third important direction of research that regards the availability of 3D scenes to train and test the different agents. Indeed, the nature of the different environments influences the variety of tasks that the agent can learn and perform.

In this work, we contribute to this third direction by collecting and presenting a previously unseen type of 3D space, *i.e.* a museum. This new environment for embodied exploration and navigation, named Art Gallery 3D (AG3D), presents unique features when compared to flats and offices. First, the dimension of the rooms drastically increases, and the same goes for the size of the building itself. In our 3D model, some rooms are as big as 20×15 meters, while the floor hosting the art gallery spans a total of 2000 square meters. However, dimensions are not the only difference with current available 3D spaces. As a second factor, the presented gallery is incredibly rich in visual features, offering multiple paintings, sculptures, and rare objects of historical and artistic interest. Every item represents a unique point of interest, and this is in contrast to traditional scenes where all elements have approximately the same visual relevance. Finally, the museum has sparse occupancy information. Many agents count on depth information to plan short-term displacements. However, when placed in the middle of an open empty hall, the depth information is less informative. In our challenging 3D scene, the agent must learn to combine RGB and depth information and not be overconfident in immediately available knowledge on the occupancy map. All these challenges make our newly-proposed 3D space a valuable asset for current and future research.

Together with the 3D model of the museum, we present a dataset for embodied exploration and navigation. For the navigation task, we annotate the position of most of the points of interest in the museum. Examples include numerous paintings, sculptures, and other relevant objects. Finally, we present an experimental analysis including the performance of existing architectures on this novel benchmark and a discussion of potential future research directions made possible

Figure 8.1: On the left: a view of the 3D model of the acquired environment. On the right: images captured during the acquisition of the scene.

by the presence of the collected 3D space.

8.1 ART GALLERY 3D DATASET

Existing datasets for indoor navigation comprise 3D acquisitions of different typologies of buildings, ranging from private houses, that cover the majority of the scenes, to offices and shops. Nevertheless, the focus of these datasets is on private spaces and there is low variance in terms of dimension and contained objects. In fact, to the best of our knowledge, among the publicly available datasets, no acquired indoor environment is composed of large rooms with a low occupied/free space ratio as in a museum. To overcome this deficiency in current literature we release a new indoor dataset for exploration and navigation captured inside a museum environment, called AG3D*.

ACQUISITION. To build the 3D model of the art gallery, we employ a Matterport camera† and related software. This technology is the same employed to collect Matterport3D and HM3D datasets of spaces [30, 117] and is particularly suitable to capture indoor photo-realistic environments. We place the camera in the physical environment and capture a 360° RGB-D image of the surrounding. Then, we repeat the same process after moving the camera approximately

---

*The dataset has been collected at the Galleria Estense museum of Modena and can be found at `https://github.com/aimagelab/ag3d`.

†`https://matterport.com/it/cameras/pro2-3D-camera`

111

1.5 meters away. Using consecutive panoramic acquisitions, the software is able to compute the 3D geometry of the space using depth information and the correspondences between the same keypoints in different acquisitions. To capture the entire museum, we make 232 different scans. Thanks to the high number of acquisitions, we are able to reproduce fine geometric and visual details of the original space (see Fig. 8.1). The resulting 3D model consists of more than 1430 m^2 of navigable space.

DATASET DETAILS. The proposed dataset allows two different tasks: exploration and navigation. Episodes for the exploration task include starting position and orientation of the agent which are sampled uniformly over the entire navigable space. The navigation dataset, instead, extends traditional Point Goal navigation where episodes are defined with a starting pose and a goal coordinate, including an additional final orientation vector. Conceptually, we can consider this setting as the link between Point Goal navigation and ImageGoal navigation since the goal is to rotate the agent towards a precise objective/scene, specifying the goal using coordinates instead of an image. We name this new setting Point Goal++ navigation (PointNav++). To create the navigation dataset we annotate 147 points of interest mostly consisting of paintings and statues. The annotated goal position is around 1 meter in front of the artwork and the goal orientation vector is directed to its center. For each point of interest, we define three episodes with different difficulties based on the geodesic distance between start and goal positions: easy ($<$ 15m), medium ($>$ 15m), and difficult ($>$ 30m). In particular, thanks to the dimension of the acquired environment, each difficult episode has a geodesic distance larger than the longest path of Matterport 3D and Gibson datasets. A comparison of the geodesic distance distribution of the episodes of various available Point Goal navigation datasets is presented in Fig. 8.2. The introduction of AG3D enables the evaluation of agents on long navigation episodes which were previously not possible and highlights the inaccuracy of components of the architecture that accumulate error over time. The exploration task dataset contains 500k, 100, 1000 episodes respectively for training, validation, and test, while the PointNav++ dataset includes 411 annotated navigation episodes.
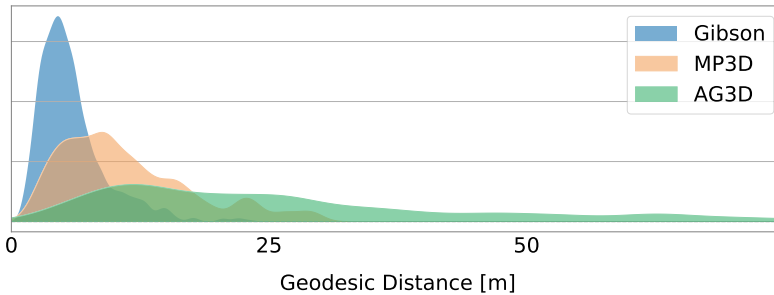
Figure 8.2: Comparison of the distribution of the geodesic distances from starting position to goal position of the episode for different datasets.

8.2 Proposed Method

We provide an experimental analysis comparing recently proposed approaches on the devised environment, both for exploration and PointNav++ tasks. The evaluated methods are consistent with recent literature on Embodied AI [32, 115] and adopt an architecture shown in Fig. 8.3, which is composed of a neural mapper, a pose estimator, and a hierarchical navigation policy. The mapper generates a representation of the environment while the agent moves, the pose estimator is in charge of locating the agent in the environment, and the policy is responsible for the movement capabilities of the agent. The core difference between the evaluated approaches resides in the navigation policy, as described in the following. For further details, see Section 3.1.

8.2.1 Mapper

The mapper module incrementally builds an occupancy grid map of the environment in parallel with the navigation task. At each timestep, the RGB-D observations $\left(s_t^{rgb}, s_t^d\right)$ coming from the visual sensors are processed to extract a $V \times V \times 2$ agent-centric map m_t where the channels indicate, respectively, the occupancy and exploration state of the currently observed region, and each pixel of the map describes the state of an area of 5×5 cm. The mapper is not limited to predicting the occupancy map of the visible space but infers also occluded and not visible regions of the local map. More details on the architecture of the mapper

can be found in Section 3.1.1. The global level map of the environment M_t has a dimensionality of $W \times W \times 2$, where $W > V$, and is built using local maps m_t step-by-step. At each timestep the pose of the agent ω_t is used to apply a roto-translation to the local map, then, the transformed local map is finally registered to the global map M_t with a moving average.

8.2.2 POSE ESTIMATOR

In order to create a coherent representation of the environment during navigation, a precise and robust pose estimation needs to be achieved. To address problems like noise in the sensors and collisions with obstacles, we adopt a pose estimator and avoid the direct use of sensor readings. The pose estimator computes the pose of the agent $\omega_t = (x_t, y_t, \theta_t)$ where (x_t, y_t) and θ_t are its position and orientation in the internal representation of the environment.

Specifically, the output of this module is the displacement $\Delta\omega_t$ caused by the agent's last action. The input of the pose estimator is the difference between consecutive readings of the pose sensor $(\widetilde{\omega}_{t-1}, \widetilde{\omega}_t)$, but such measure could be noisy and needs to be adjusted. To do so, we use consecutive local maps (m_{t-1}, m_t) coming from the mapper as feedback. At each timestep $\Delta\omega_t$ is used to compute the pose of the robot ω_t:

$$\omega_t = \omega_{t-1} + \Delta\omega_t, \tag{8.1}$$

where we assume $\omega_0 = (0, 0, 0)$ without loss of generality and ω_0 corresponds to the center of the map M_t with the agent facing north.
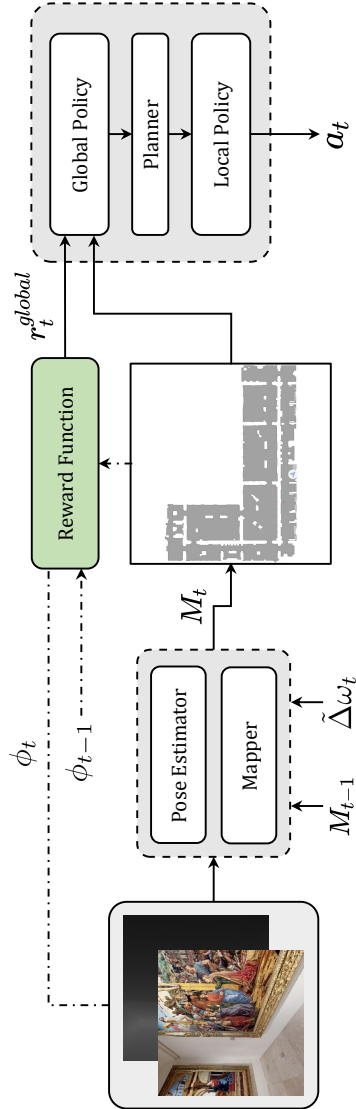
8

Figure 8.3: Overall architecture of the models employed for exploration and navigation on AG3D .

8.2.3 NAVIGATION POLICY

The navigation policy is the module that determines the movement of the agent in the environment. Its hierarchical design is required in order to allow the agent to uncouple high-level navigation concepts, such as navigating through different rooms, and low-level concepts, like obstacle avoidance. The navigation policy is defined by a three-component module consisting of a global farsighted policy, a deterministic planner, and a local policy for atomic action inference. The architecture of such modules is presented in Section 3.1.3.

The global policy is the high-level component of the navigation policy and is responsible for extracting a long-term goal on the global map g_t. The global policy is trained with reinforcement learning using PPO [129] to maximize different rewards used in literature. In the experiments, we employ and compare different reward methods, namely Coverage, Anticipation, and Curiosity. The Coverage reward [32, 118] maximizes the information gathered at each timestep, expressed in terms of the area seen. The Anticipation reward [115] is defined by comparing the predicted local occupancy map with the ground truth considering also occluded areas. The Curiosity reward [108] encourages the agent towards areas that maximize the prediction error of a model trained to predict future states, thus improving the learning of the dynamics of the environment.

Given the global goal on the map, the planner has the task of computing a short-term goal on the map that the agent should reach. We employ an A* algorithm on the global map M_t to plan a path from the current position of the agent to the global goal and a local goal l_t is computed on the obtained trajectory within a distance D from the agent.

The local policy is the module that allows the movement of the agent in the environment and its objective is to reach the local goal l_t determined by the planner. The input of the local policy, formed by the relative displacement from the position of the agent to the local goal l_t and the current RGB observation φ_t^{rgb}, is processed to compute an atomic action a_t. The available actions are *move ahead 0.25m*, *turn left 10°*, *turn right 10°*, with the addition of a *stop* action when performing the navigation task. During training with reinforcement learning, the

Figure 8.4: An episode of PointNav++ in AG3D where consecutive frames have a distance of 10 timesteps approximately. The red frame indicates the stop action in the traditional PointNav task. The green frame corresponds to the stop action in PointNav++.

8

reward of the local policy r_t^{local} encourages the agent to reduce the distance from the local goal.

Following the hierarchical design, the global goal is sampled every η timesteps, while the local goal is reset if a new global goal is sampled, if the previous local goal is found to be in an occupied area, or if the previous local goal has been reached.

8.3   Experimental Setup

We perform some experiments on the proposed dataset comparing various models trained with different global rewards on another dataset, with models trained from scratch or finetuned on AG3D on exploration and PointNav++ to evaluate the performance gap between these approaches and highlight the difference between the characteristics of AG3D compared to other datasets. An example of an episode of Point Goal++ navigation of Art Gallery 3D is shown in Fig. 8.4.

8.3.1   Evaluation Protocol

The baselines are trained with Coverage, Anticipation, and Curiosity rewards on the Gibson dataset for $\approx$ 5M frames corresponding to 12 GPU days on a single NVIDIA V100. The best-performing approach among the baseline is also both

trained from scratch and finetuned, but since high-quality textures and memory occupancy of AG3D do not allow training with the same number of environments in parallel as Gibson, we trained the model from scratch on AG3D with the same GPU time for $\approx$ 2.8M frames, while the finetuned model is trained for $\approx$ 1M additional frames.

For the exploration task, we evaluate the following metrics: intersection-over-union (IoU) between the map built at the end of the episode and the ground-truth map. Acc measures the correctly reconstructed map in m^2. AS indicates the area seen by the agent during exploration (in m^2). FIoU, OIoU, FAS, and OAS measure, respectively, IoU and area seen for free and occupied portions of the environment. TE and AE are the translation and angular error between estimated and ground-truth pose measured respectively in meters and degrees. Point Goal++ navigation is evaluated considering these metrics: distance to the goal (D2G) and orientation error (OE) are the mean geodesic distance to the goal and the mean orientation error at the end of the episode. The orientation error is computed considering the vector between the center of the artwork and the position of the agent as ground truth. Success rate (SR) is the percentage of episodes terminated successfully. In PointNav++ the agent needs to be within 0.2 meters of the goal and with an orientation error lower than 10 degrees. PointNav success rate (PNSR) and angular success rate (ASR) consider only one component of SR; respectively D2G and OE. SPL and SoftSPL (sSPL) are success rates weighted on the length of the trajectory of the agent.

8.3.2 IMPLEMENTATION DETAILS

The experiments are performed by extracting 128×128 RGB-D observations from the acquired 3D model using the Habitat simulator. The maximum length of the exploration episodes during training is set to $T = 500$. Regarding the mapping process, we set $V = 101$ and $W = 2881$ for the local and global map dimensionalities. The action space grid $H \times H$ of the global policy is 240×240. The maximum distance of the local goal l_t from the position of the agent is $D = 0.5$m for exploration and $D = 0.25$m for PointNav++.

Model	Training	IoU↑	FloU↑	OIoU↑	Acc↑	AS↑	FAS↑	OAS↑	TE↓	AE↓
Noise-Free										
Anticipation [115]	Gibson	0.163	0.170	0.157	294.4	290.6	258.3	32.3	0.0	0.0
Curiosity [108]	Gibson	0.175	0.184	0.166	317.9	317.5	281.7	35.8	0.0	0.0
Coverage [32, 118]	Gibson	0.214	0.237	0.191	403.1	384.3	341.1	43.2	0.0	0.0
Coverage [32, 118]	AG3D	0.219	0.239	0.200	400.6	354.6	316.6	38.0	0.0	0.0
Coverage [32, 118]	Gibson+AG3D	**0.296**	**0.313**	**0.278**	**531.8**	**470.2**	**418.1**	**52.1**	0.0	0.0
Noisy										
Anticipation [115]	Gibson	0.144	0.157	**0.131**	269.6	281.7	249.9	31.9	**0.48**	**2.95**
Curiosity [108]	Gibson	0.119	0.151	0.086	251.1	307.3	272.8	34.5	2.62	15.99
Coverage [32, 118]	Gibson	0.148	0.203	0.093	327.0	380.7	337.9	42.8	2.98	12.54
Coverage [32, 118]	AG3D	0.144	0.200	0.088	320.0	356.4	317.3	39.2	2.65	13.35
Coverage [32, 118]	Gibson+AG3D	**0.191**	**0.266**	0.116	**427.3**	**461.9**	**413.2**	**48.7**	2.68	10.16

Table 8.1: Exploration results over the 100 episodes of the AG3D validation split in noise-free and noisy conditions. Using both Gibson and AG3D during the training guarantees significantly higher performance.

8.4   EXPERIMENTAL RESULTS

8.4.1   EXPLORATION RESULTS

As a first experiment, in Table 8.1 we compare the considered models on the exploration task on the AG3D validation set. Each exploration episode has a length of $T = 1000$ timesteps during which the agent has to disclose the initially unknown environment. Among the baselines trained only on the Gibson dataset, the Coverage-based model achieves the best results in terms of IoU and area seen in both noise-free and noisy settings. The model trained with Coverage from scratch obtains competitive results even using fewer training frames (2.8M vs. 5M), showing the importance of adapting the models to AG3D. This conclusion is supported by the fact that the model trained on Gibson and finetuned for 1M frames on AG3D achieves the best results on noise-free and noisy settings, with a significant margin on the second-best model. In both settings, the performance gap in terms of area seen (85.9m^2 and 81.2m^2) and IoU (0.082 and 0.043) between the best models trained only on Gibson dataset and using AG3D denotes the need of adapting the weight of the models to the different visual characteristics and occupation of AG3D.

Model	Training	SPL↑	sSPL↑	SR↑	PNSR↑	ASR↑	Steps↓	D2G↓	OE↓
Noise-Free									
Anticipation [115]	Gibson	0.697	0.780	0.803	0.873	0.808	364.3	4.131	12.2
Curiosity [108]	Gibson	0.625	0.706	0.732	0.803	0.732	416.4	7.934	17.0
Coverage [32, 118]	Gibson	0.760	0.838	0.876	0.954	0.883	314.6	0.700	5.2
Coverage [32, 118]	AG3D	**0.805**	**0.875**	**0.898**	**0.973**	**0.908**	**270.3**	**0.268**	**4.8**
Coverage [32, 118]	Gibson+AG3D	0.793	0.873	0.883	0.964	0.891	273.1	0.323	5.0
Noisy									
Anticipation [115]	Gibson	0.211	0.788	0.224	0.255	0.387	338.6	3.152	32.2
Curiosity [108]	Gibson	0.225	0.655	0.243	0.275	0.341	446.3	9.746	38.7
Coverage [32, 118]	Gibson	0.228	0.783	0.243	0.260	0.392	348.6	3.165	34.1
Coverage [32, 118]	AG3D	0.235	0.832	0.248	0.273	0.445	306.2	2.420	28.8
Coverage [32, 118]	Gibson+AG3D	**0.373**	**0.853**	**0.399**	**0.443**	**0.543**	**283.8**	**1.430**	**19.8**

Table 8.2: PointNav++ results on the AG3D navigation episodes under noise-free and noisy settings. AG3D is fundamental to reach the best results.

8.4.2 NAVIGATION RESULTS

Moving on to the navigation task, models trained on exploration substitute the global goal with a fixed goal specified by the navigation episode. Experimental results on PointNav++, shown in Table 8.2, present a similar trend as on the exploration task. In fact, the Coverage model has the best results in terms of SPL and success rate related metrics among the models trained only on Gibson. Moving to the Coverage models trained on AG3D, in the noise-free setting, the model trained from scratch achieves the best results even in comparison to the finetuned counterpart which is trained with more than double the total observations (2.8M vs 6M). This behavior can be explained by the performance of its mapper which is trained for more frames using visual observation from AG3D (2.8M vs 1M) and extracts a more detailed map sacrificing robustness and generalization. Accordingly, in the noisy setting, the higher robustness of the Coverage-based model finetuned on AG3D regains the first place with a noteworthy margin on the other models, while the Coverage model trained from scratch goes down to the second position in terms of SPL and SR. As in the case of the exploration task, the performance gap between models trained on Gibson and using AG3D (0.045 and 0.145 for SPL in noise-free and noisy settings) stresses the importance of adapting the parameters to the features extracted from AG3D. Moreover, it is worth noting that the gap of the best model from noise-free to noisy navigation (0.432

for SPL) is a consequence of the length of the navigation episodes of AG3D, and the difficulty of performing precise lengthy trajectories in the presence of noise. This is an interesting aspect that the AG3D dataset offers for exploration in future works.

8

8

9

Conclusions

This dissertation contributes to the research in Embodied AI, aiming to foster future research on the topic and hoping to help researchers willing to work in this complex field. This final chapter starts with a section that summarizes the work included in this thesis, reviewing the contributions introduced in each chapter. In the following section, we include a discussion about the problems and lacks afflicting the current research in this field. After that, the possible future work and directions of research are described. To conclude this thesis, we present some final remarks and the activities carried out during the Ph.D. program.

9.1 Contributions of the Thesis

9.1.1 Exploration with Intrinsic Motivation

In Chapter 3, we presented an impact-driven approach for robotic exploration in indoor environments. Differently from previous research that considered a setting with procedurally-generated environments with a finite number of possible states, we tackle a problem where the number of possible states is non-numerable.

To deal with this scenario, we exploit a deep neural density model to compute a running pseudo-count of past states and use it to regularize the impact-based reward signal. The resulting intrinsic reward allows to efficiently train an agent for exploration even in absence of an extrinsic reward, including also remarkable results on the downstream task of navigation. The proposed agent stands out from the recent literature on embodied exploration in photo-realistic environments.

9.1.2 EXPLORATION AND RECOUNTING

Chapter 4 presents a new setting for Embodied AI that is composed of two tasks: exploration and captioning. The architecture of eX^2 uses intrinsic rewards applied to exploration in a photo-realistic environment and a speaker module that generates captions. The captioner produces sentences according to a speaker policy that could be based on three metrics. The experiments show that eX^2 is able to generalize to unseen environments in terms of exploration, while the speaker policy functions to filter the number of timesteps where the caption is actually generated.

9.1.3 EFFICIENT EXPLORATION AND SMART SCENE DESCRIPTION

In this work proposed in Chapter 5, we have improved the architecture presented in the previous chapter in both the exploration and the speaker module. Alongside these improvements, we devise a novel metric for the task that enables the evaluation and comparison of different baselines. This approach is a viable solution to gain insights into the perception and navigation capabilities of embodied agents. Moreover, the generalization capabilities of the modules adopted allow real-world deployment without major redesigns.

9.1.4 EMBODIED AGENTS IN CHANGING ENVIRONMENTS

Chapter 6 proposed *Spot the Difference*: a new task for navigation agents in changing environments. In this setting, the agent has to find all variations that occurred in the environment with respect to an outdated occupancy map. Since current datasets of 3D spaces do not account for such variety, we collected a new

dataset with different layouts for the same environment. We tested two exploration agents on this task and proposed a novel reward term to encourage the discovery of meaningful information during exploration. The proposed agent outperforms the competitors and can identify changes in the environment more efficiently.

9.1.5 NAVIGATION IN THE REAL WORLD

In Chapter 7 we have presented LoCoNav, an out-of-the-box architecture for embodied navigation in the real world. After a phase of training using simulation, the implementation of the trained model on a real robotic platform needs to be performed. We present a series of techniques specifically designed for real-world deployment. Experiments are conducted in reality on challenging navigation paths and in a realistic office-like environment demonstrating the validity of our approach.

9.1.6 NAVIGATION AT THE ART GALLERY

In this work contained in Chapter 8, we introduced AG3D, a photo-realistic 3D dataset designed for embodied exploration and navigation tasks. The dataset has been collected in an art gallery, which features larger and more uncluttered spaces compared to most of the environments available in commonly used benchmark datasets. For the PointNav task, we propose a variant that is more suitable to the type of environment in the AG3D dataset. The variant entails not only reaching the specified coordinates, as in standard PointNav but also assuming a specified orientation. We also present an experimental comparison of exploration approaches on the devised dataset, which can serve as baselines for future research in museum-like environments.

9.2 KEY FUTURE DIRECTIONS

After the description of the contributions presented in the thesis, this section describes what is still missing and what we think could be important deficiencies

in the research on Embodied AI.

One of the aspects that has been only slightly explored in the current literature on Embodied AI is the capacity of the robots to reason about semantic concepts in the environment. Great effort has been poured into agents' ability to perform high-level navigation reaching objects and coordinates, or following textual instructions. These tasks are usually done by training models with massive use of data, instructions, and training time, focusing only on reaching the final goal. However, if we take Object Goal navigation as a case study, the relations between different types of objects or commonsense knowledge has been hardly exploited to pursue the task. If the agent is requested to find a chair, there is a high probability that it will be found next to a table. Instead, the goal object is a bathtub, commonsense suggests finding a bathroom first. Unfortunately, these concepts and relations are usually not acquired or annotated in current datasets, or in case the dataset contains this type of annotations they are usually very noisy or not curated. Our expectations for the next years of research in Embodied AI is to focus more on the reasoning of the agents, instead of working exclusively looking for higher numbers on the available datasets.

Another major drawback is that in current research on Embodied AI, the agent's interactions are limited. Usually, most tasks consist of just an input at the beginning of the episode and the agent does not interact with the surrounding environment for the entire duration of the episode. We think that before being able to reach the seamless interaction between robots and humans, ideally the episode should not be reset and the agents should be able to interact more with surrounding humans and other entities and be able to condition their behavior using these interactions.

Additionally, after having considered some of the work that has been published in the last few years [68, 161, 107], we would like to address the fact that taking inspiration from some traditional robotic approaches, also learning-based AI research could flourish and produce improvements over previous literature. Even if this aspect could seem simple, we think that traditional robotic knowledge should be linked tighter with the research on this topic.

The last concern we would like to address is the actual realism of simulating

platforms. After having discussed photo-realistic environments in Chapter 8, we believe that a major problem is the presence of artifacts, reconstruction errors, and unrealistic physics behaviors in such simulators. All these issues degrade the performances of agents trained in simulation when deployed in real-world settings. Improving the simulators in these aspects could return a significant boost to embodied agents' performance in the real world.

9.3 FUTURE WORK

Regarding the possible future directions of research, we are interested in tackling the issues presented in the previous section. A first step towards real semantic navigation could be giving the agent the ability to understand the semantic meaning of the surrounding environment, *i.e.* being able to understand that a certain room is a kitchen, a bathroom, or another room, studying its visual appearance and the objects contained in it. Moreover, observing the outstanding results achieved in the research on conversational agents with large language models (LLMs), an increasing interest in exploiting such models for robotic tasks has risen. For example, recent work has started leveraging LLMs for manipulation [63, 84, 3] and navigation tasks [61, 133]. However, further research is still required to fully integrate LLMs capabilities on board of an embodied agent to consistently upgrade its reasoning. We aim to be part of such research by developing new approaches that combine aspects of both robotic and conversational agents. In fact, the advances brought by this dissertation can be further evolved by integrating the semantics contained in large language models, because while the presented work is focusing on the observation level, LLMs could be adopted as a general high-level controller.

In parallel, we also think that extending the duration of episodes and moving towards multi-goal navigation could be another way to move towards fluid robot-human interaction. In such a task, the robot should be able to request more information on the object to look for and be corrected when it goes the wrong way. The final goal of this change is the development of lifelong or never-resetting agents, while in the current setting, agents are reset every episode and the capacity

to backtrack from errors is not explored.

9.4   FINAL REMARKS

The efforts presented in this thesis have been published in international journals and conferences. For example, the work on intrinsic motivation presented in Chapter 3 has been published in IEEE Robotics and Automation Letters and has been presented at IEEE International Conference on Robotics and Automation 2022, while the paper related to Chapter 5 on efficient exploration and scene description has been accepted at IEEE International Conference on Robotics and Automation 2023. We aim to follow the track started with the research presented in the previous chapters and hope that Embodied AI researchers will find it useful for their future work.

9

9.5   PH.D. ACTIVITIES

This final section presents a list of the main activities carried out by the candidate during the Ph.D. program in Information and Communication Technologies.

9.5.1   EXCHANGE PERIODS

1 June - 31 August 2022: Visiting Student Researcher at Stanford University in the research laboratory "Autonomous Systems Lab (ASL)" supervised by Professor Marco Pavone.

9.5.2   TEACHING ACTIVITIES

2020 - present: Teaching Assistant for "Computer Architectures and Lab. (Calcolatori Elettronici e Lab.)" course held by Prof. Rita Cucchiara;

2019 - 2020: Project Tutor for "Computer Vision and Cognitive Systems" course held by Prof. Rita Cucchiara;

2019 - 2022: Presenter for "Into The Future" university orientation program;

2020 - 2021: Project Tutor for "Neural Network Computing, AI and Machine Learning for Automotive" course held by Prof. Rita Cucchiara;

2021: Lecturer for "AI for Automotive" course (topic: SLAM - Simultaneous Localization and Mapping);

2021 - 2022: Project Tutor for "AI for Automotive" course held by Prof. Rita Cucchiara;

9 October 2021: Presenter for "In the mind of robots, how AI teaches robots to navigate" laboratory;

4 November 2022: Presenter for "Humanities & Intelligence" laboratory.

9.5.3 CONFERENCE ATTENDANCES

10 - 15 January 2021 IAPR 25th International Conference on Pattern Recognition (ICPR), 2020, *Milan, Italy (Remote)*;

27 September - 1 October 2021: International Conference on Computer Analysis of Images and Patterns (CAIP), 2021, *Nicosia, Cyprus (Remote)*;

23 - 27 May 2022 IEEE International Conference on Robotics and Automation (ICRA), 2022, *Philadelphia, US*.

23 - 27 May 2022 21st International Conference on Image Analysis and Processing (ICIAP), 2022, *Lecce, Italy (Remote)*.

21 - 25 August 2022 IAPR 26th International Conference on Pattern Recognition (ICPR), 2022, *Montreal, Canada (Remote)*.

9.5.4 SEMINARS AND WORKSHOPS

September 2020: Attendance at "About Time" seminar, speaker: Prof. Arnold Smeulders;

November 2020: Attendance at "Deep Scene Perception without Labeled Data" seminar, speaker: Prof. Luigi Di Stefano;

January - February 2021: Lecturer for "Deep Learning, Artificial Intelligence and Neurolinguistic Processing (IBM, SAS)" training course at IFOA Modena;

March - April 2021: Lecturer for "Data Analysis and Data Visualization" training course at IFOA Bologna;

June 2021: Attendance at "Research in videogames: use of deep learning for saliency estimation and cheating prevention" seminar, speaker: Dr. Iuri Frosio;

October 2021: Attendance at "Brain Inspired Computing Workshop: from Neuroscience to Artificial Intelligence";

October 2021: Attendance at "Safe, Interaction-Aware Decision Making and Control for Robot Autonomy" seminar, speaker: Prof. Marco Pavone;

November 2021: Lecturer for "Python and Machine Learning" training course at Prometeia.

9.5.5 Schools

19 - 23 July 2021: Attendance and completion of the "4th Advanced Course on Data Science and Machine Learning - ACDL 2021" summer school.

9.5.6 Technical Program Committees

Conferences.

ACM International Conference on Multimedia (ACM Multimedia), 2020, *Seattle, US*;

ACM International Conference on Multimedia (ACM Multimedia), 2021, *Chengdu, China*;

IAPR International Conference on Pattern Recognition (ICPR), 2022, *Montreal, Canada*;

IEEE International Conference on Robotics and Automation (ICRA), 2022, *London, UK*.

Journals.

ACM Transactions on Multimedia Computing Communications and Applications (TOMM);

IEEE Transactions on Geoscience and Remote Sensing (TGRS);

IAPR Pattern Recognition Letters (PRL);

IEEE Robotics and Automation Letters (RA-L).

9

Bibliography

[1] Achiam, J. and Sastry, S. (2017). Surprise-based intrinsic motivation for deep reinforcement learning. In *NeurIPS Workshops*.

[2] Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[3] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., et al. (2022). Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning*.

[4] Allegretti, S., Bolelli, F., and Grana, C. (2019). Optimized block-based algorithms to label connected components on gpus. *IEEE Transactions on Parallel and Distributed Systems*.

[5] Anderson, P., Chang, A., Chaplot, D. S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al. (2018a). On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

[6] Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). SPICE: Semantic Propositional Image Caption Evaluation. In *Proceedings of the European Conference on Computer Vision*.

[7] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018b). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[8] Anderson, P., Shrivastava, A., Truong, J., Majumdar, A., Parikh, D., Batra, D., and Lee, S. (2021). Sim-to-Real Transfer for Vision-and-Language Navigation. In *Proceedings of the Conference on Robot Learning*.

[9] Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018c). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[10] Anjomshoae, S., Najjar, A., Calvaresi, D., and Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

[11] Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., and Savarese, S. (2019). 3D Scene Graph: A structure for unified semantics, 3D space, and camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[12] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics Workshops*.

[13] Barraco, M., Stefanini, M., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2022). CaMEL: Mean Teacher Learning for Image Captioning. In *Proceedings of the International Conference on Pattern Recognition*.

[14] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. (2016). DeepMind Lab. *arXiv preprint arXiv:1612.03801*.

[15] Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*.

[16] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*.

[17] Bigazzi, R., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2023). Embodied Agents for Efficient Exploration and Smart Scene Description. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[18] Bigazzi, R., Landi, F., Cascianelli, S., Baraldi, L., Cornia, M., and Cucchiara, R. (2022a). Focus on Impact: Indoor Exploration with Intrinsic Motivation. *IEEE Robotics and Automation Letters*.

[19] Bigazzi, R., Landi, F., Cascianelli, S., Cornia, M., Baraldi, L., and Cucchiara, R. (2022b). Embodied navigation at the art gallery. In *Proceedings of the International Conference on Image Analysis and Processing*.

[20] Bigazzi, R., Landi, F., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2020). Explore and Explain: Self-supervised Navigation and Recounting. In *Proceedings of the International Conference on Pattern Recognition*.

[21] Bigazzi, R., Landi, F., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2021). Out of the Box: Embodied Navigation in the Real World. In *Proceedings of the International Conference on Computer Analysis of Images and Patterns*.

[22] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding Horizon "Next–Best–View" Planner for 3D Exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[23] Biswas, J. (2019). The Quest For" Always-On" Autonomous Mobile Robots. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[24] Bolelli, F., Allegretti, S., Baraldi, L., and Grana, C. (2019). Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling. *IEEE Transactions on Image Processing*.

[25] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

[26] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2019). Large-scale study of curiosity-driven learning. In *Proceedings of the International Conference on Learning Representations*.

[27] Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation. In *Proceedings of the International Conference on Learning Representations*.

[28] Cartillier, V., Ren, Z., Jain, N., Lee, S., Essa, I., and Batra, D. (2020). Semantic MapNet: Building Allocentric Semantic Maps and Representations from Egocentric Views. *arXiv preprint arXiv:2010.01191*.

[29] Cascianelli, S., Costante, G., Bellocchio, E., Valigi, P., Fravolini, M. L., and Ciarfuglia, T. A. (2016). A robust semi-semantic approach for visual localization in urban environment. In *Proceedings of the IEEE International Smart Cities Conference*.

[30] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3D: Learning from RGB-D Data in Indoor Environments. In *Proceedings of the International Conference on 3D Vision*.

[31] Changpinyo, S., Sharma, P., Ding, N., and Soricut, R. (2021). Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[32] Chaplot, D. S., Gandhi, D., Gupta, S., Gupta, A., and Salakhutdinov, R. (2019a). Learning To Explore Using Active Neural SLAM. In *Proceedings of the International Conference on Learning Representations*.

[33] Chaplot, D. S., Gandhi, D. P., Gupta, A., and Salakhutdinov, R. R. (2020). Object Goal Navigation using Goal-Oriented Semantic Exploration. In *Advances in Neural Information Processing Systems*.

[34] Chaplot, D. S., Lee, L., Salakhutdinov, R., Parikh, D., and Batra, D. (2019b). Embodied Multimodal Multitask Learning. *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[35] Chen, S., Guhur, P.-L., Schmid, C., and Laptev, I. (2021). History aware multimodal transformer for vision-and-language navigation. In *Advances in Neural Information Processing Systems*.

[36] Chen, S., Guhur, P.-L., Tapaswi, M., Schmid, C., and Laptev, I. (2022a). Learning from unlabeled 3d environments for vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision*.

[37] Chen, S., Guhur, P.-L., Tapaswi, M., Schmid, C., and Laptev, I. (2022b). Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[38] Chen, T., Gupta, S., and Gupta, A. (2019). Learning Exploration Policies for Navigation. In *Proceedings of the International Conference on Learning Representations*.

[39] Choi, S., Zhou, Q.-Y., and Koltun, V. (2015). Robust reconstruction of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[40] Cornia, M., Baraldi, L., and Cucchiara, R. (2020a). SMArT: Training Shallow Memory-aware Transformers for Robotic Explainability. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[41] Cornia, M., Baraldi, L., Fiameni, G., and Cucchiara, R. (2022). Universal Captioner: Inducing Content-Style Separation in Vision-and-Language Model Training. *arXiv preprint arXiv:2111.12727*.

[42] Cornia, M., Stefanini, M., Baraldi, L., and Cucchiara, R. (2020b). Meshed-Memory Transformer for Image Captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[43] Da Silva, F. L., Taylor, M. E., and Costa, A. H. R. (2018). Autonomously Reusing Knowledge in Multiagent Reinforcement Learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[44] Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al. (2020). RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[45] Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Salvador, J., Ehsani, K., Han, W., Kolve, E., Farhadi, A., Kembhavi, A., et al. (2022). Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*.

[46] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *Proceedings of the International Conference on Learning Representations*.

[47] Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[48] Georgakis, G., Bucher, B., Arapin, A., Schmeckpeper, K., Matni, N., and Daniilidis, K. (2022). Uncertainty-driven planner for exploration and navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[49] Gervet, T., Chintala, S., Batra, D., Malik, J., and Chaplot, D. S. (2022). Navigating to objects in the real world. *arXiv preprint arXiv:2212.00922*.

[50] González-Banos, H. H. and Latombe, J.-C. (2002). Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research*.

[51] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*.

[52] Grana, C., Borghesani, D., and Cucchiara, R. (2010). Optimized block-based connected components labeling with decision trees. *IEEE Transactions on Image Processing*.

[53] Guhur, P.-L., Tapaswi, M., Chen, S., Laptev, I., and Schmid, C. (2021). Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[54] Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017). Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[55] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[56] Hessel, J., Holtzman, A., Forbes, M., Bras, R. L., and Choi, Y. (2021). CLIPScore: A Reference-free Evaluation Metric for Image Captioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[57] Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the International Conference on Machine Learning*.

[58] Holz, D., Basilico, N., Amigoni, F., and Behnke, S. (2010). Evaluating the Efficiency of Frontier-based Exploration Strategies. In *ISR and ROBOTIK*.

[59] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. In *Proceedings of the International Conference on Learning Representations*.

[60] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*.

[61] Huang, C., Mees, O., Zeng, A., and Burgard, W. (2023). Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[62] Huang, L., Wang, W., Chen, J., and Wei, X.-Y. (2019). Attention on Attention for Image Captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[63] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the Conference on Robot Learning*.

[64] Irshad, M. Z., Ma, C.-Y., and Kira, Z. (2021). Hierarchical Cross-Modal Agent for Robotics Vision-and-Language Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[65] Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., and Baldridge, J. (2019). Stay on the path: Instruction fidelity in vision-and-language navigation. *arXiv preprint arXiv:1905.12255*.

[66] Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*.

[67] Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. (2020). Sim2Real Predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*.

[68] Karkus, P., Cai, S., and Hsu, D. (2021). Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[69] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[70] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[71] Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). ViZDoom: A doom-based ai research platform for visual reinforcement learning. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*.

[72] Kingma, D. and Ba, J. (2015). Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[73] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

[74] Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. In *CEC*.

[75] Krantz, J., Wijmans, E., Majumdar, A., Batra, D., and Lee, S. (2020). Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *Proceedings of the European Conference on Computer Vision*.

[76] Ku, A., Anderson, P., Patel, R., Ie, E., and Baldridge, J. (2020). Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[77] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*.

[78] Landi, F., Baraldi, L., Cornia, M., Corsini, M., and Cucchiara, R. (2021a). Multimodal Attention Networks for Low-Level Vision-and-Language Navigation. *Computer Vision and Image Understanding*.

[79] Landi, F., Baraldi, L., Cornia, M., and Cucchiara, R. (2021b). Working Memory Connections for LSTM. *Neural Networks*.

[80] Landi, F., Baraldi, L., Corsini, M., and Cucchiara, R. (2019). Embodied vision-and-language navigation with dynamic convolutional filters. *Proceedings of the British Machine Vision Conference*.

[81] Landi, F., Bigazzi, R., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2022). Spot the difference: A novel task for embodied agents in changing environments. In *Proceedings of the International Conference on Pattern Recognition*.

[82] Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., et al. (2021). igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the Conference on Robot Learning*.

[83] Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., et al. (2020). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Proceedings of the European Conference on Computer Vision*.

[84] Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. (2023). Code as policies: Language model programs for embodied control. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[85] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics Workshops*.

[86] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision*.

[87] Liu, W., Chen, S., Guo, L., Zhu, X., and Liu, J. (2021). CPTR: Full Transformer Network for Image Captioning. *arXiv preprint arXiv:2101.10804*.

[88] Locobot (2020). LoCoBot: An Open Source Low Cost Robot. https://locobot-website.netlify.com.

[89] Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[90] Lu, J., Yang, J., Batra, D., and Parikh, D. (2018). Neural Baby Talk. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[91] Luperto, M., Antonazzi, M., Amigoni, F., and Borghese, N. A. (2020). Robot exploration of indoor environments using incomplete and inaccurate prior knowledge. *Robotics and Autonomous Systems*.

[92] Mac, T. T., Copot, C., Tran, D. T., and De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*.

[93] Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*.

[94] Mayo, B., Hazan, T., and Tal, A. (2021). Visual navigation with spatial attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[95] Mohamed, S. and Rezende, D. J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*.

[96] Morad, S. D., Mecca, R., Poudel, R. P., Liwicki, S., and Cipolla, R. (2021). Embodied visual navigation with automatic curriculum learning in real environments. *IEEE Robotics and Automation Letters*.

[97] Murali, A., Chen, T., Alwala, K. V., Gandhi, D., Pinto, L., Gupta, S., and Gupta, A. (2019). PyRobot: An Open-source Robotics Framework for Research and Benchmarking. *arXiv preprint arXiv:1906.08236*.

[98] Nardi, L. and Stachniss, C. (2020). Long-term robot navigation in indoor environments estimating patterns in traversability changes. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[99] Niroui, F., Zhang, K., Kashino, Z., and Nejat, G. (2019). Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robotics and Automation Letters*.

[100] Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*.

[101] Ordonez, V., Kulkarni, G., and Berg, T. (2011). Im2Text: Describing Images Using 1 Million Captioned Photographs. In *Advances in Neural Information Processing Systems*.

[102] Oßwald, S., Bennewitz, M., Burgard, W., and Stachniss, C. (2016). Speeding-Up Robot Exploration by Exploiting Background Information. *IEEE Robotics and Automation Letters*.

[103] Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the International Conference on Machine Learning*.

[104] Oudeyer, P.-Y. and Kaplan, F. (2009). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*.

[105] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[106] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In *Proceedings of the International Conference on Machine Learning*.

[107] Partsey, R., Wijmans, E., Yokoyama, N., Dobosevych, O., Batra, D., and Maksymets, O. (2022). Is mapping necessary for realistic pointgoal navigation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[108] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the International Conference on Machine Learning*.

[109] Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement. In *Proceedings of the International Conference on Machine Learning*.

[110] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[111] Poppi, S., Bigazzi, R., Rawal, N., Cornia, M., Cascianelli, S., Baraldi, L., and Cucchiara, R. (2023). Towards Explainable Embodied Navigation and Recounting. *Under Review*.

[112] Qi, Y., Wu, Q., Anderson, P., Wang, X., Wang, W. Y., Shen, C., and Hengel, A. v. d. (2020). Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[113] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning*.

[114] Raileanu, R. and Rocktäschel, T. (2021). RIDE: Rewarding impact-driven exploration for procedurally-generated environments. In *Proceedings of the International Conference on Learning Representations*.

[115] Ramakrishnan, S. K., Al-Halah, Z., and Grauman, K. (2020). Occupancy Anticipation for Efficient Exploration and Navigation. In *Proceedings of the European Conference on Computer Vision*.

[116] Ramakrishnan, S. K., Chaplot, D. S., Al-Halah, Z., Malik, J., and Grauman, K. (2022). PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[117] Ramakrishnan, S. K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J. M., Undersander, E., Galuba, W., Westbury, A., Chang, A. X., Savva, M., Zhao, Y., and Batra, D. (2021a). Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Advances in Neural Information Processing Systems*.

[118] Ramakrishnan, S. K., Jayaraman, D., and Grauman, K. (2021b). An Exploration of Embodied Visual Exploration. *International Journal of Computer Vision*.

[119] Rawal, N., Bigazzi, R., Baraldi, L., and Cucchiara, R. (2023). AIGeN: An Adversarial Approach for Instruction Generation in Vision-and-Language Navigation. *Under Review*.

[120] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[121] Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[122] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*.

[123] Rosano, M., Furnari, A., Gulino, L., and Farinella, G. M. (2020). On Embodied Visual Navigation in Real Environments Through Habitat. In *Proceedings of the International Conference on Pattern Recognition*.

[124] Saputra, M. R. U., Markham, A., and Trigoni, N. (2018). Visual SLAM and structure from motion in dynamic environments: A survey. *ACM Computing Surveys*.

[125] Savinov, N., Dosovitskiy, A., and Koltun, V. (2018). Semi-parametric topological memory for navigation. In *Proceedings of the International Conference on Learning Representations*.

[126] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. (2019). Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[127] Sax, A., Emi, B., Zamir, A. R., Guibas, L., Savarese, S., and Malik, J. (2019). Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. In *Proceedings of the Conference on Robot Learning*.

[128] Schmidhuber, J. (2010). Formal Theory of Creativity, Fun, and Intrinsic Motivation. *IEEE Trans. on Autonomous Mental Development*.

[129] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

[130] Selin, M., Tiger, M., Duberg, D., Heintz, F., and Jensfelt, P. (2019). Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments. *IEEE Robotics and Automation Letters*.

[131] Sharma, P., Ding, N., Goodman, S., and Soricut, R. (2018). Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[132] Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[133] Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

[134] Sridharan, M. and Mota, T. (2020). Commonsense Reasoning to Guide Deep Learning for Scene Understanding. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[135] Srinivasan, K., Raman, K., Chen, J., Bendersky, M., and Najork, M. (2021). WIT: Wikipedia-based Image Text Dataset for Multimodal Multilingual Machine Learning. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[136] Stachniss, C. (2009). *Robotic Mapping and Exploration*. Springer.

[137] Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. (2019). The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv preprint arXiv:1906.05797*.

[138] Sun, Y., Gomez, F., and Schmidhuber, J. (2011). Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *AGI*.

[139] Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. (2017). #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

[140] Telea, A. (2004). An image inpainting technique based on the fast marching method. *J. of Graphics Tools*.

[141] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). YFCC100M: The new data in multimedia research. *Communications of the ACM*.

[142] Truong, J., Chernova, S., and Batra, D. (2021). Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*.

[143] Truong, J., Rudolph, M., Yokoyama, N., Chernova, S., Batra, D., and Rai, A. (2022). Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation. In *Proceedings of the Conference on Robot Learning*.

[144] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

[145] Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). CIDEr: Consensus-based Image Description Evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[146] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2017). Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[147] Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., and Batra, D. (2019). DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. In *Proceedings of the International Conference on Learning Representations*.

[148] Xia, F., Shen, W. B., Li, C., Kasimbeg, P., Tchapmi, M. E., Toshev, A., Martín-Martín, R., and Savarese, S. (2020). Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*.

[149] Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. (2018). Gibson Env: Real-world perception for embodied agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[150] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*.

[151] Yadav, K., Ramrakhya, R., Ramakrishnan, S. K., Gervet, T., Turner, J., Gokaslan, A., Maestre, N., Chang, A. X., Batra, D., Savva, M., et al. (2022). Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*.

[152] Yamauchi, B. (1997). A Frontier-Based Approach for Autonomous Exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*.

[153] Ye, J., Batra, D., Das, A., and Wijmans, E. (2021a). Auxiliary tasks and exploration enable ObjectNav. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[154] Ye, J., Batra, D., Wijmans, E., and Das, A. (2021b). Auxiliary Tasks Speed Up Learning Point Goal Navigation. In *Proceedings of the Conference on Robot Learning*.

[155] You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. (2019). Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *Proceedings of the International Conference on Learning Representations*.

[156] Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[157] Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., Choi, Y., and Gao, J. (2021a). VinVL: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[158] Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. (2020a). Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*.

[159] Zhang, X., Sun, X., Luo, Y., Ji, J., Zhou, Y., Wu, Y., Huang, F., and Ji, R. (2021b). RSTNet: Captioning with Adaptive Attention on Visual and Non-Visual Words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[160] Zhang, Y., Tan, H., and Bansal, M. (2020b). Diagnosing the Environment Bias in Vision-and-Language Navigation. *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[161] Zhao, X., Agrawal, H., Batra, D., and Schwing, A. G. (2021). The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[162] Zhu, D., Li, T., Ho, D., Wang, C., and Meng, M. Q.-H. (2018). Deep Reinforcement Learning Supervised Autonomous Exploration in Office Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[163] Zhu, F., Liang, X., Zhu, Y., Yu, Q., Chang, X., and Liang, X. (2021). Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[164] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Glossary

D2G distance to the goal. 34, 35, 118, 120

Deep Learning a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. 2, 37

Div diversity. 53, 54, 68, 71, 72

DME Density Model Estimation. 26–35, 61, 70

DR difference reward. 88–94

DRL Deep Reinforcement Learning. 8, 17, 18, 28

ED-S episode description score. 58, 69, 71, 72

Embodied AI research field that strives for the creation of embodied agents which learn, through interaction and exploration, to solve challenging tasks within their environments. 2–4, 7, 8, 10–12, 37, 38, 41, 75, 76, 87, 97, 109, 113, 123, 124, 126, 128

eX^2 *Explore and Explain* method. 39, 40, 50, 51, 53–55, 57, 124

FAS area seen of the free space. 29–31, 118, 119

FIoU intersection-over-union of the free space. 29–31, 118, 119

FoV field of view. 72, 103

GPU specialized electronic circuit designed to manipulate and alter memory to accelerate the creation of images. 3, 12, 117, 118

HFR hard failure rate. 106, 107

Image Captioning a task at the intersection between Computer Vision and NLP whose goal is to generate a natural language description of a given image. 7, 14, 38, 39, 46

Acknowledgments

I would like to express my gratitude to everyone who played an instrumental role in the realization of my Ph.D. program and everything that surrounded it.

Beginning with my personal life, I feel extremely fortunate for the support of my family, especially my mother and girlfriend, who played a fundamental role in the achievements documented in this dissertation. As I said, without their presence, none of this would have been possible, and I am deeply grateful. I would also like to thank the rest of my family, including my uncles, aunts, and cousins, who have always been interested in my work and provided invaluable encouragement. My heartfelt thanks also go to all of my friends, who provided a much-needed outlet when things weren't going well and were the first to celebrate my accomplishments.

In the academic sphere, I would like to thank my advisor, Prof. Rita Cucchiara, for serving as a reference and mentor in my work and for providing me with this opportunity. I would also like to extend my thanks to the Post-Docs, Lorenzo, Silvia, and Marcella, for their support, guidance, and knowledge-sharing. Federico, a colleague and friend, deserves a special mention for his support, guidance, and collaboration ("I will visit you in Amsterdam, wait for me").

I would also like to extend my gratitude to all my colleagues at AImageLab, my second home for the past three years, who have been constant companions of laughter, coffee, and a lot of food. Thank you for your suggestions, ideas, memes, and the shared coffee breaks and lunch hours. I am thankful to everyone who joined the lab with me or later, and everyone who left, for making my time there memorable.

I am also grateful to my friends and colleagues at Stanford, even though my time there was brief. I would like to thank Prof. Marco Pavone for accepting me into his lab and for allowing me to visit Stanford. My heartfelt thanks to Shreyas, who was not only my guide but also a good friend. I will always cherish our conversations about work and hobbies. I wish him the best of luck with his new job in Atlanta. I would also like to thank my Stanford buddies, Nicholas and John, for their companionship and the stimulating conversations we had over lunch at Treehouse, at the gym, and over coffee. Finally, I want to thank everyone else at ASL, your warmth and hospitality made my stay comfortable and enjoyable, and I am grateful for the warm farewell you gave me.