PhD Program in Science and Technology for Electronic and Telecommunication Engineering

curriculum

Computational Vision, Recognition and Machine Learning

Spatial Reasoning with Graph Neural Networks

Francesco Giuliari

University of Genova
Department of Electrical, Electronics and Telecommunication Engineering and Naval Architecture (DITEN)
Istituto Italiano di Tecnologia (IIT)
Pattern Analysis and Computer Vision (PAVIS)

Dr. Alessio Del Bue	Supervisor
Dr. Yiming Wang	Advisor
Dr. Maurizio Valle	PhD Course Coordinator

This dissertation is submitted for the degree of
Doctor of Philosophy (XXXVI cycle)
XXXVI Cycle
February 2024

I would like to dedicate this thesis to my parents, who always supported me.

Declaration

Acknowledgements

I would like to acknowledge all the brilliant people I've met during these three years of my PhD.

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Alessio Del Bue, and my advisor Dr. Yiming Wang, for their guidance and mentorship in all these years. Their insights and ideas have been invaluable in all of the projects we did together.

I want to thank all the people I've collaborated with during these years, your help has been key in finalizing many projects. In particular, Prof. Marco Cristani, who I have always seen as a Role Model; and Dr. Christian Wolf, from whom I learned a lot in my six months period of internship.

Next, I want to thank all my fellow students in the PAVIS lab: Gianluca Scarpellini, Davide Talon, Julio Carrazco, Javed Ahmad, Sanket Thankur, Giancarlo Paoletti, Saber Mohammadi, and Dario Serez. You have been amazing people and I am glad to have been able to share these three years with you. I have learned a lot from you guys and I hope to keep working with you in the future.

Finally, I want to thank my family for supporting me during all the time, and for always being there when I needed them.

Abstract

Spatial reasoning, a crucial aspect of human cognition, is essential for interpreting and interacting within a three-dimensional environment. This ability is equally crucial in automated systems, particularly those functioning in dynamic, real-world scenarios where spatial interaction is a fundamental requirement.

This thesis focuses on integrating spatial reasoning into computational systems, starting with the explicit incorporation of spatial reasoning through manually encoded rules for robot navigation. Here, the emphasis is on maintaining complete control over the system's actions, ensuring predictability and transparency in its behavior.

Learning-based solutions are then analyzed for cases where the complexity of scenarios makes manual encoding of rules impractical. Specifically, the use of Graph Neural Networks (GNNs) is motivated by their proficiency in handling data with varying components and types, showcasing their effectiveness in tasks like object localization and object reassembly. We demonstrate that, by using GNNs combined with Attention mechanisms and a graph representation of the 3D environment, we are able to achieve remarkable capabilities in representing complex spatial relationships, resulting in state-of-the-art performance in Object Localization in Partial 3D Scenes.

Finally, we enhance the spatial reasoning capabilities of GNNs by integrating them with Diffusion Probabilistic Models (DPMs). This approach uses DPMs for iterative solution refinement from random noise, moving beyond single-step predictions.

In this thesis, we show that Graph Neural Networks are an extremely versatile and effective solution for solving spatial problems that can be represented using graphs. They are able to solve complex tasks, such as Object Localization in Partial Scenes and Object Reassembly, outperforming other non-GNN-based approaches.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Rationale

Spatial Reasoning, also called *Spatial Intelligence*, refers to our ability to perceive the visual-spatial world accurately and performs mental transformation upon the perceived space (Byrne and Johnson-Laird, 1989; Gardner, 2011).

This skill enables us to understand objects within the context of the 3D world, where their location is as critical as their visual appearance. Spatial reasoning is a fundamental skill that we employ naturally in most, if not all, of our everyday tasks (Ishikawa and Newcombe, 2021). For instance, while walking, we instinctively understand the optimal trajectory to avoid obstacles and navigate around other people. In searching for an object, we can guess a plausible location based on our surroundings. Given a jigsaw puzzle, we manage to move pieces around so that nearby pieces form a coherent part of the picture, eventually creating a complete image. The development of this skill is not innate but rather acquired in our early years of life.

From a young age, we learn to deal with the spatial nature of our world, a fundamental and unchanging aspect of our reality. We develop spatial reasoning skills through play, such as using toys that require fitting specific shapes into corresponding holes and building objects with interlocking pieces like LEGO (Nagy-Kondor, 2017). This early interaction with spatial elements lays the groundwork for more complex spatial understanding and problem-solving in later life.

Similarly, for computational systems interfacing with our world, the integration of some form of spatial reasoning is crucial (Sisbot et al., 2007). Just as we humans learn to navigate and interact with a three-dimensional world, computational systems, especially those deployed in dynamic, real-world scenarios, must possess the capability to understand and interpret spatial relationships (Holzmann, 2009; Lawton et al., 1990; Polkowski and Osmialowski, 2010). This is vital in a range of applications, from autonomous vehicles navigating busy streets to robotic systems performing tasks in cluttered and unpredictable environments. In these contexts, the ability to accurately perceive spatial relationships and adapt to new spatial configurations is key to the effectiveness and safety of the system (Alves et al., 2018).

There are two primary approaches to integrating spatial reasoning into automated systems (Karpas and Magazzeni, 2020): Programming-based spatial reasoning and learning-based spatial reasoning. Programming-based Reasoning (in which we include also the category defined as “Model-based" in the cited work) relies on explicitly coded rules and models to interpret spatial information (Duchoň et al., 2014; Mobasheri, 2017). These models use principles of geometry and spatial relationships to define decision rules that the system has to follow to perform spatial reasoning. The significant advantage of this approach is the high level of control it offers over the system’s behavior. This control is particularly important in scenarios where a fully explainable system is needed, such as in robotic applications where understanding the rationale behind each decision is crucial. However, while programming-based systems offer predictability and transparency, they are not effective in handling the complexity and variability inherent in real-world environments. Their rigid structure can be a limitation when dealing with novel situations or when the environment changes unpredictably, as demonstrated by the loss of the Mars Polar Lander, which crashed on the surface of Mars in 1999 due to an unconventional environment condition not being properly handled (Albee et al., 2000).

On the other hand, Learning-based Spatial Reasoning leverages the capabilities of machine learning, especially deep learning, to enable systems to deduce spatial relationships from data (Chaplot et al., 2020a; Du et al., 2020). This approach typically involves training models on extensive datasets that encompass a wide array of examples in diverse scenarios. Through exposure to many examples, these systems are able to learn from the data directly without the need to rely on any hard-coded rules. This method is particularly advantageous in complex scenarios where encoding every possible rule for interaction with the environment is impractical or impossible. For instance, in Object Localization, where localization can be achieved by accounting for the location of the surrounding objects, the sheer number of combinations of possible objects and their placement makes it impossible to account for every potential situation. Learning-based solutions can adapt to such complex environments by learning directly from a vast amount of examples of objects arrangement in a multitude of possible rooms. These solutions excel in tasks requiring high flexibility and adaptability, as they are able to adjust to new information or changes in the environment.

In this thesis, we analyze how to integrate spatial reasoning into computational systems for a variety of tasks: such as robot navigation, object localization in 3D space, and object reassembly. We show that each of these tasks has nuances and requirements that motivate either the use of hand-crafted rules or the use of a learning-based approach. Furthermore, we showcase that Graph Neural Networks are a well-motivated choice for Spatial Reasoning tasks, due to their ability to handle dynamic data and their adaptability for a wide range of tasks.

1.2 Works Summary

In this section, we provide a high-level summary of the works that compose the thesis, highlighting the motivation that guided our choices in these research directions.

1.2.1 Robot Navigation with Programming-based Spatial Reasoning

For robot navigation, one of the many tasks to assess the capability of an agent to move and act in a novel environment is searching for an object in an unseen environment. This task, dubbed Active Visual Search (AVS), or also Object-goal Navigation, leverages several basic human skills, including self-localization, visual search, and object detection, along with the necessary motor skills for reaching the object of interest, especially in highly cluttered and dynamic environments. In this case, spatial reasoning is needed to understand both how to navigate the environment avoiding obstacles, and also to define areas to explore in order to search for the target object. The task can be solved with both methods that rely on programming-based spatial reasoning or learning-based spatial reasoning, as both have their own pros and cons. In this thesis, we solve the task with hand-crafted spatial reasoning combined with an online planner called POMCP (Silver and Veness, 2010). We model the 3D environment as a top-down map in 2D, forming an $n \times n$ grid, where each cell describes a small part of the scene. We use this 2D map to keep track of which part of the environment has been seen during the exploration and which parts still have to be observed. We also use it to record which part of the environment is navigable, meaning that there are no obstacles and the robot can move in that location. To guide the search to the target object, we initialize a uniform distribution in the boundary between the explored and unexplored area and update it using a Bayesian update every time we observe a new part of the scene. We model the exploration problem as a Partially Observable Markov Decision Process (POMDP) and use POMCP combined with our 2D map to create a policy that prioritizes the exploration into frontier zones, maximizing the chance to move toward unexplored areas and minimizing wasted movements. Solving the problem this way allows for an effective solution that is fully explainable, robust in both simulation and the real world, and does not require any training data.

However, while effective, our solution does not use prior knowledge about the target object's typical location relative to its environment. For example, it does not differentiate whether a target object like a television is more likely to be found in a living room rather than a kitchen. This is because manually encoding these rules is very hard. For instance, the TV can be found in the living room, but it can also be in the kitchen, the dining room, and even the bedroom, at least for the "European" style of houses, but that is not the case for other cultures. It would also depend on the size of the house, as the size of the house would indicate a measure of wealth, and that would also influence whether the house would have one or more TVs and in which rooms, and so on. These nuances make it exceedingly difficult to encapsulate all potential spatial rules manually. This challenge has led the research community to gravitate toward learning-based approaches, particularly those involving Deep Learning (Chaplot et al., 2020a). These methods, trained on datasets of spatial scenarios, can learn all these complex rules directly from the data without any manual intervention.

1.2.2 Object Localization with Graph Neural Networks

Building on this perspective, we hypothesize that it would be beneficial for AVS agents to be able to estimate the location of the target object and use it as a goal for navigation. Given the complexity of integrating this directly into a Reinforcement Learning (RL) agent, we decouple the problem and treat the object location estimation from a purely perceptual perspective without accounting for the agent navigation. Specifically, we aim to predict the location of a target object within the unseen part of a room based on partial 3D scans.

We tackle this problem by employing Graph Neural Networks (GNNs) (Kipf and Welling, 2017a). We train the network to analyze the arrangement of objects in the part of the room we can see and use that information to predict where the target object might be located. The idea to solve this is that we can use priors regarding the arrangement of the objects to extrapolate the locations of an unknown one. For example, in many cases, the TV is located opposite the couch or a sofa; the coffee table is directly in front of the sofa; the nightstand is beside the bed.

Starting from a partial scan of a 3D scene, we first identify all the objects and their location. We then formulate the set of objects as a graph, where each node identifies one particular object. We connect all the nodes to each other, forming a complete graph, where each edge in this graph contains, as information, the Euclidean distance between a couple of objects. To locate the unseen object, we add a new node in the graph describing the object and use the GNN to predict the values on the edges between the target node and all other objects. The distances describe a sphere in 3D space around each visible object. We consider the point of intersection as the predicted location for the target object. This method involving the use of a Scene Graph as a high-level description of the environment combined with our use of Graph Neural Networks is able to reach state-of-the-art results in the task of object localization in partial scenes, outperforming previous learning and non-learning-based approaches.

1.2.3 Reassembly Tasks with Graph-based Diffusion Models

Encouraged by the impressive performance of Graph Neural Networks (GNNs) in the Object Localization Task, we extend our investigation to more intricate spatial tasks. One such task is Reassembly, where the objective is to determine a rigid transformation for each piece from a set of individual parts to construct a complete entity, such as a 2D image or a 3D object.

This Reassembly task shares similarities with object localization, which allows us to adapt the approach used previously. However, it also presents unique challenges necessitating modifications to our method. In object localization, the goal is to predict the position of a single object, given the locations of other objects in the environment. Reassembly, in contrast, involves unknown locations for all pieces, with the aim of arranging them to form a singular coherent structure. This absence of initial spatial information from our input data adds complexity to the task. We are required to rely heavily on the visual characteristics of the individual pieces for spatial reasoning, unlike in object localization where spatial cues from known object locations are available. This shift in reliance from spatial to visual cues requires a different approach in our use of GNNs, emphasizing the need for sophisticated visual processing capabilities in the network.

To address this challenge, we propose a graph-based diffusion model. This model progressively adjusts the position and orientation of each piece, starting from an initial state of pure noise, to assemble a coherent structure. Utilizing Graph Neural

Networks (GNNs) enables us to understand the relative position and orientation of each piece in relation to the others. Additionally, we incorporate Diffusion Probabilistic Models (DPMs) (Ho et al., 2020), which facilitate a gradual refinement towards the correct poses. This iterative refinement process enhances the accuracy of localization, as it allows for the correction of errors at each step of the refinement. The combination of these techniques - GNNs for contextual positioning and Diffusion Probabilistic Models for iterative adjustment - proves effective in the complex task of assembling disparate pieces into a singular, cohesive entity.

1.3 Contributions

To summarize, the main contributions of this thesis are the following:

- We conduct an in-depth study on different ways to treat spatial data for a multitude of settings, such as robot navigation, object localization, and object reassembly. We illustrate how for each of these tasks, the spatial data has to be treated differently and provide appropriate solutions for each of them.
- We devise a novel programming-based spatial reasoning formulation for robotic navigation in unknown environments using Partially Observable Markov Decision Processes. This allows for the integration of spatial reasoning into analytical planners with the possibility for adjustments and tweaking of the spatial navigation policy.
- We propose two novel Graph-based deep learning approaches for spatial reasoning that learn connections between objects' locations directly from data. We showcase the effectiveness of this formulation for the tasks of object localization and reassembly.
- We demonstrate how to combine Diffusion Probabilistic Models with Graph Neural Networks to iteratively refine the location of multiple object fragments simultaneously. We detail how the combination of these two approaches achieves state-of-the-art results in Reassembly tasks thanks to their effective spatial reasoning capabilities.

1.4 Publications

The work presented in this thesis is based on the following publications:

- **Giuliari, F.**, Castellini, A., Berra, R., Del Bue, A., Farinelli, A., Cristani, M., Setti, F. & Wang, Y. (2021, September). POMP++: Pomcp-based Active Visual Search in unknown indoor environments. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1523-1530). IEEE.
- **Giuliari, F.**, Skenderi, G., Cristani, M., Wang, Y., & Del Bue, A. (2022). Spatial commonsense graph for object localization in partial scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 19518-19527).
- **Giuliari, F.**, Scarpellini, G., Fiorini, S., Mererio, P., & Del Bue, A.,(2023). DiffAssemble: A Unified Graph-Diffusion Model for 2D and 3D Reassembly. Currently under Review

The following are other works that I have been involved in during the PhD program that are not discussed in this thesis:

- **Giuliari, F.**, Skenderi, G., Cristani, M., Del Bue, A., & Wang, Y. (2023). Leveraging commonsense for object localization in partial scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Franco, L., Placidi, L., **Giuliari, F.**, Hasan, I., Cristani, M., & Galasso, F. (2023). Under the hood of transformer networks for trajectory forecasting. Pattern Recognition, 138, 109372.
- Taioli, F., **Giuliari, F.**, Wang, Y., Berra, R., Castellini, A., Del Bue, A., Farinelli, A., Cristani, M., & Setti, F. (2023). Unsupervised Active Visual Search with Monte Carlo planning under Uncertain Detections. Currently under review.
- **Giuliari, F.**, Scarpellini, G., James, S., Wang, Y., & Del Bue, A. (2023). Positional Diffusion: Ordering Unordered Sets with Diffusion Probabilistic Models. Currently under review.

1.5 Chapter Description

The structure of this thesis is outlined as follows: Chapter 2 serves as the starting point, where we delve into the related works that provide a foundational understanding of spatial reasoning and the approaches we use in the following chapters. In Chapter 3, our exploration into the Active Visual Search task, as applied to robot navigation, is detailed. Following this, Chapter 4 discusses our approach to Object Localization in partial 3D scenes, emphasizing the application of Graph Neural Networks. Chapter 5 is dedicated to demonstrating our methodology using Graph-based Diffusion Models for tackling reassembly tasks in both 2D and 3D contexts. The thesis concludes with Chapter 6, where we present our final thoughts and insights on employing Graph Neural Networks for Spatial Reasoning tasks.

Chapter 2

Related Works

In this chapter, we introduce the literature related to spatial reasoning for computational systems, focusing on the three tasks we identified: Robot Navigation, Object Localization, and Object Reassembly. In Section 2.1, we report the details regarding the two categories of approaches: Programming-based and Learning-based approaches for the three tasks. Section 2.2, reports the relevant works regarding Graph Neural Networks, and their usage for spatial tasks. Finally, Section 2.3, introduces Diffusion Probabilistic Models.

Specific related works for each particular task are described in their chapters.

2.1 Approaches to Spatial Reasoning

2.1.1 Programming-based Spatial Reasoning

Programming-based spatial reasoning involves manually encoding a series of decision rules that guide how an autonomous system interfaces with the physical world. For robot navigation, the decision describes how to move towards the goal given the knowledge of the environment. Early works like the bug algorithm (Lumelsky and Stepanov, 1987), have very simple rules to deal with physical obstacles, like for example following a wall to avoid the collision, and then proceeding to the goal when the wall is not blocking the line-of-sight to the target. Its variants, Bug1 and Bug2, offer different strategies for navigating around obstacles. Despite its simplicity, the algorithm has been instrumental in various applications, although it faces limitations in complex environments due to its inefficiency and lack of adaptability. Other works, like A* (Hart et al., 1968) and D* (Stentz, 1994), allow for more complex planning, by following a set of heuristics to find the shortest path on a grid-like map of the environment. These methods are effective and are still relevant today for navigation and are used for real-world tasks, albeit with particular modifications that account for sensor noise and online mapping of the environment (Duchoň et al., 2014).

More complex solutions are Model-based approaches (Lauri et al., 2022), here instead of directly defining the decision rules for the behavior, we define a model of the environment and the agent, and define how actions taken by the agent affect the environment. We categorize this type of approach alongside the Programming-based approaches as they still require a manual definition of the model. The approaches are typically formulated as a Partially Observable Markov Decision Process (POMDP) and have been proposed for a variety of navigation tasks such as SLAM (Lauri and Ritala, 2016; Martinez-Cantin et al., 2009), search and tracking (Zhang et al., 2013) and for many non-navigation tasks such as grasping (Zhou et al., 2017) and 6DOF object pose estimation (Eidenberger and Scharinger, 2010).

In Chapter 3, we present a POMDP-based approach for Robot Navigation in Unseen Environments, where the goal is to navigate to a target object that is placed in an unknown location. We show how with our novel POMDP-based formulation, we can solve the problem efficiently while maintaining full transparency on the agent's behavior, outperforming previous state-of-the-art methods.

While Programming-based approaches are effective and allow for a fully explainable system(Lauri et al., 2022), they require a manual definition of the model for both the agent and the 3D physical space it acts on, which is hard to define if not impossible for more complex scenarios. For this reason, many researchers are shifting their work from programming or model-based approaches to learning-based approaches, as they allow learning to solve the problem from examples of solutions, without manually defining the underlying environment model.

2.1.2 Learning-based Spatial reasoning

Learning-based Spatial Reasoning Methods employ machine learning algorithms, particularly deep learning with Neural Networks (NN), to learn interactions in the physical world by examining a large number of examples of such interactions. Neural Networks have been effectively employed to solve a multitude of spatial-visual problems.

Early cases show how learning the spatial relationships between 2D object detections can improve accuracy by leveraging priors between object co-occurrence in a single image (Hu et al., 2018; Liu et al., 2018). Subsequent works have further enhanced performances in 2D detection tasks by analyzing not only co-occurrences but also the spatial relations in co-occurring objects, such as food being *above* a plate (He et al., 2021).

Neural networks excel not only in inferring spatial relations between objects on the image plane but also in translating the 2D visual information into physical 3D representations. They are highly effective in creating bird's eye view semantic maps from front-facing cameras, a critical component in accurately mapping environments for indoor robot navigation (Chaplot et al., 2020a) and autonomous driving applications (Can et al., 2022).

Additionally, they are suited for purely spatial problems where a high level of spatial reasoning is required. For instance, in room rearrangement (Wei et al., 2023), networks arrange furniture pieces into plausible layouts. In molecular generation (Hoogeboom et al., 2022), they position and connect atoms to form molecules with specific properties.

Dealing with 2D and 3D data requires different types of neural networks, each suited for specific tasks:

Convolutional Neural Networks (CNNs) These networks are the standard for dealing with images(Krizhevsky et al., 2012) or dense 3D voxel representations(Wu et al., 2015). CNNs utilize convolutional filters to process data in a grid-like topology. They excel in capturing hierarchical patterns in images, making them ideal for tasks such as object recognition, image classification, and image segmentation. They are also extremely relevant in spatial problems with dense data, as they can be used to generate distribution over dense space in the form of heat maps.

Point-cloud Neural Networks These networks specialize in processing 3D data represented as point clouds, which are sets of data points in space. Unlike CNNs, point-cloud networks must handle the irregularity and unordered nature of point clouds. Pioneering architectures like PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) introduced ways to directly process point clouds, enabling applications in 3D object classification, part segmentation, and scene understanding.

Graph-Neural Networks (GNNs) GNNs (Kipf and Welling, 2017a) are designed to handle data structured as graphs, making them well-suited for relational reasoning and complex networked data. They are particularly effective in tasks where relationships and interactions between elements are crucial, such as social network analysis, protein-protein interaction networks, and chemical molecule structure prediction.

In Chapters 4 and 5, we use Graphs Neural Networks to solve Object Localization and Object Reassembly. We choose to use these networks for these problems as they are perfectly suited to our representation of the problem: in Object Localization, we have to predict the position of one object given the arrangement of the other objects in the room, while in Object Reassembly we have to predict the Translation and rotation of a set of pieces to compose a single coherent structure. Graphs are a perfect choice to represent these data: each node identifies one particular object or piece, and the edges connect all objects and pieces between each other. We then use node and edge features to encode the objects and pieces' visual information and their spatial relations with each other.

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) (Kipf and Welling, 2017a) have become extremely relevant in solving spatial reasoning problems, thanks to their ability to model complex, graph-structured data.

These Networks use Message Passing as the primary process to exchange information between neighboring nodes. This allows them to contextualize the node information within the more global information present in the graph, which is the key to understanding spatial context.

This process starts with nodes holding their own features, representing entities in a spatial scenario, such as objects of a room or fragments of an object. During the message-passing phase, every node collects features from its neighboring nodes. The method of collection is not just a straightforward aggregation; it involves a more complex process where the neighboring nodes' features are combined with the node's features. This combination can include various operations like concatenation and summation, and is where most GNN models differ, as the type of aggregation is very task-specific.

The Message Passing can be repeated multiple times, each time increasing by one the distance that the information travels across the graph. With one iteration, the node information is passed to its neighbors. With two iterations, the node information is then spread to the neighbor's neighbors, and so on.

Incorporating attention mechanisms (Vaswani et al., 2017) further increases the capabilities of GNNs. With attention, the importance of information from different neighbors can be weighted variably. In technical terms, attention scores are computed based on the features of both the nodes and their connecting edges. These scores determine how much focus should be placed on each neighbor's information during the aggregation process (Brody et al., 2022; Shi et al., 2021). This selective focus allows the network to adaptively prioritize certain parts of the graph, which is especially crucial in complex spatial scenarios where some interactions or relationships are more critical than others.

In spatial reasoning tasks, this ability of GNNs to process and interpret graph-structured data becomes particularly valuable. They can model the intricate spatial relationships and dependencies inherent in tasks such as object localization, where understanding the layout and relationship between different objects is essential, or in object reassembly, where the spatial arrangement of fragments influences the object's final composition.

Overall, the message-passing and attention mechanisms of GNNs provide a powerful framework for tackling a wide range of spatial problems. By effectively capturing the complexity and nuances of spatial relationships, GNNs offer a robust approach to understanding and reasoning about space in various contexts.

2.2.1 Graph Neural Network for Spatial Reasoning

Scene graphs are a graph-based representation to define an image or a 3D scene via the elements it is composed of, and the spatial relations between them. Scene graphs were initially used to describe images of scenes based on the elements they contained and how they were connected. The work of (Johnson et al., 2015) showed that for certain applications, e.g. Image Retrieval, the abstraction of higher-level image concepts was improving the results compared to using the standard pixel space. Since then, scene graphs have been successfully used in many other tasks such as image captioning (Gu et al., 2019a; Xu et al., 2019; Yang et al., 2019) and visual question answering (Lee et al., 2019; Shi et al., 2019).

Recently, the use of scene graphs has also been extended to the 3D domain, providing an efficient solution for 3D scene description. The 3D scene graph can vary from a simple representation of a scene and its content, in which the objects are nodes, and the spatial relationships between objects are the graph's edges (Gay et al., 2018; Wald et al., 2020; Wu et al., 2021); to a more complex hierarchical structure that describes the scene at different levels: from the image level with description about the scene from only a certain point of view, moving up to a higher level description of objects, rooms and finally buildings (Armeni et al., 2019). The work of (Zhou et al., 2019b) uses a scene graph to augment 3D indoor scenes with new objects matching their surroundings using a message-passing approach. A relatively similar task is indoor scene synthesis (Wang et al., 2019a), in which the goal is to generate a new scene layout using a relation graph encoding objects as nodes and spatial/semantic relationships between objects as edges. A graph convolutional generative model synthesizes novel relation graphs and thus new layouts. In (Dhamo et al., 2021a) and (Luo et al., 2020) the authors use a 3D scene graph to describe the object arrangement, they then modify the scene graph and generate a new scene.

In Chapter 4, we use a Graph-based approach for the task of Object Localization in Partial Scenes. We use an attention-based GNN combined with a heterogeneous graph description of the environment that encodes the visible object locations as well as “commonsense" knowledge that describes how these objects are used and where they are typically located. We showcase how GNNs are an excellent solution, able to correctly perform spatial reasoning to locate an unseen object, given our heterogeneous representation of the scene objects.

Following this, we then test the ability of GNNs to deal with a similar but more complex problem: recovering the position and orientation of a set of multiple pieces to form a coherent singular object, for both 2D and 3D scenarios. For 2D the task is to arrange and rotate square patches to form an image, while for 3D the task is to place and orientate fragments of a broken 3D object to recompose it. While our GNN-based approach works fine by itself, we show we can improve accuracy, by gradually recovering the individual pieces' position and orientation with Diffusion Probabilistic Models, instead of predicting it in a single step.

2.3 Diffusion Probabilistic Models

Diffusion Probabilistic Models (DPM) (Ho et al., 2020) have recently emerged as a powerful tool in the field of deep learning, particularly for generating high-quality samples in diverse domains such as image and audio synthesis (Dhariwal and Nichol, 2021; Song et al., 2020). These models are based on the concept of a diffusion process, where data is gradually transformed from a complex distribution into a simpler, known distribution, typically Gaussian.

At the core of diffusion models is a Markov chain of latent variables. In this chain, each step incrementally adds a small amount of Gaussian noise to the data, progressively transforming it into random noise over a fixed number of steps. This forward diffusion process effectively maps the data distribution into a Gaussian distribution.

The innovative aspect of these models lies in their ability to reverse this diffusion process. This reversal is achieved by training a neural network to perform the opposite of the diffusion steps, essentially denoising the data at each stage. The network learns to predict the original data from its noised version, with the training involving the minimization of a specific form of variational lower bound on the data likelihood.

In spatial reasoning, diffusion models open up new avenues for generating and manipulating spatial data. They can be applied to synthesize new room layouts, create 3D object models, or predict future states in dynamic environments. Their capacity to model complex distributions is particularly advantageous for tasks where spatial relationships are intricate and challenging to capture with traditional generative models.

The flexibility and general applicability of diffusion probabilistic models position them as a potent tool for a range of spatial reasoning tasks. By mastering the reversal of a process that turns complex spatial data into simple distributions, these models offer a unique and powerful approach to understanding and manipulating complex spatial relationships, providing innovative solutions to challenging problems in the domain of spatial reasoning.

2.3.1 Diffusion Probabilistic Models for Visual-Spatial Reasoning

In the last couple of years, DPMs have seen an exponential increase in adoption for visual tasks. Starting from 2020, with the famous work called "Diffusion Models beats Gan in Image Synthesis" (Dhariwal and Nichol, 2021) it has become the competing standard for Image Generation. Since then, the research done on these models has spread quickly, and these models have been subject to many modifications. They have been adapted to work in latent spaces instead of directly in the pixel space, forming another famous model i.e. "Stable Diffusion" (Rombach et al., 2021), they have been adopted for Video Generation (Ho et al., 2022), for Image Inpainting (Lugmayr et al., 2022) and also for text-generation (Gong et al., 2022).

Aside from being adapted to multiple tasks, these methods have also been altered to have new properties. DDIM (Song et al., 2020), is one of the most important works that change how DPM works. They propose to have a non-Markovian chain, allowing to skip steps during the inference process, which enormously speeds up the generation procedure. Additionally, they also show how the forward and reverse processes can be made deterministic, forming a bijective mapping between the noise distribution and the data distribution. Other works show how the diffusion model can be expressed not only as a noising process in the space of real numbers but also as a noising process in the discrete space (Austin et al., 2021).

In Chapter 5, we show a novel formulation combining DPMs and GNNs and show how the combination of these two models is critical in solving the Reassembly tasks.

Chapter 3

Robot Navigation in Unseen Rooms with Programming-based Spatial Reasoning

Abstract

In this chapter, we focus on the problem of learning online an optimal policy for Active Visual Search (AVS) of objects in unknown indoor environments with a programming-based approach. We propose POMP++, a planning strategy that introduces a novel formulation on top of the classic Partially Observable Monte Carlo Planning (POMCP) framework to allow training-free online policy learning in unknown environments. We present a new belief reinvigoration strategy that enables the use of POMCP with a dynamically growing state space to address the online generation of the floor map. We evaluate our method on two public benchmark datasets, AVD, which is acquired by real robotic platforms and Habitat ObjectNav, which is rendered from real 3D scene scans, achieving the best success rate with an improvement of $>10\%$ over the state-of-the-art methods.

3.1 Introduction

Finding a specific object in an indoor environment is a human activity that is trivial to define but complex to encode into an autonomous agent like a robot(Anderson et al., 2018). This task leverages several basic human skills, including self-localisation, visual search, object detection, along with all the necessary motor skills for reaching the object of interest, especially in highly cluttered and dynamic environments.

In such context, this paper focuses on a specific task, Active Visual Search (AVS), where a robot is asked to navigate in indoor environments to search for a given object, with the performance being evaluated mainly on the success rate and the path efficiency (see Fig. 3.1). To address AVS, a joint solution for both visual perception and motion planning is needed. Recent works mostly tackle this problem by intertwining deep Reinforcement Learning (RL) techniques, *e.g.* deep recurrent Q-network (DRQN)(Hausknecht and Stone, 2015), with visual semantics, by either feeding deep visual embeddings to policy learning networks (Schmid et al., 2019; Ye et al., 2019) or obtaining 3D scene semantics to guide the planning (Chaplot et al., 2020a). These RL-based methods require a large amount of training data, *i.e.* sequences of observa-

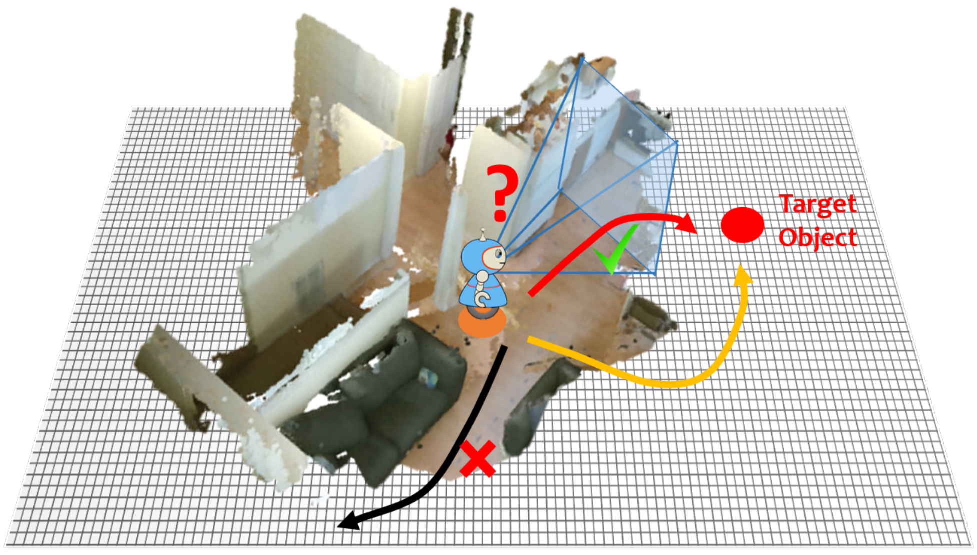

Fig. 3.1 A robot is navigating in an *unknown* environment with the task of visually searching for a target object (highlighted in red), *i.e.* to have the object captured within the field of view of the camera. The robot is driven by a motion policy based on partially observed scene to visually detect the target object located in the unseen part of scene (highlighted in gray area) with the shortest travelled path (highlighted in red), to avoid longer trajectories (in yellow) or missing entirely the target (in black).

tions of various lengths, covering successful and unsuccessful episodes from real and simulated scenarios. Such pre-collected data may not be representative of the deployed (unknown) environments, thus possibly reducing the performance of a learning-based strategy.

In this chapter, we present a method to specifically address AVS in *unknown* environments without a priori knowledge of the area. While online policy learning has been used in previous approaches (Wang et al., 2020b), our method is the first to address this problem in the *unknown* environment setting. Our approach, named POMP++, is developed within the Partially Observable Monte Carlo Planning (POMCP) framework with two main adaptations introduced. We first present a formal definition of the AVS task in unknown environments following the Partially Observable Markov Decision Process (POMDP) formulation, in particular addressing the problem of a growing

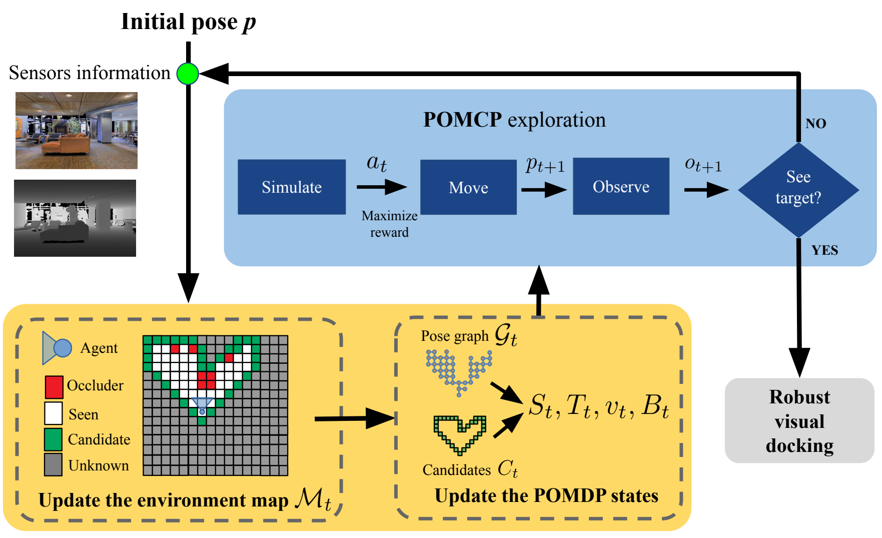

Fig. 3.2 Overall architecture of POMP++. The yellow block represents the partial knowledge model of the environment that gets updated during the exploration, while the blue block represents the exploration strategy to locate the target. The mathematical notation is defined as: pose p_t, action a_t, observation o_t, POMDP state space S_t, POMDP transition model T_t, visibility function v_t, belief B_t.

search space. Secondly, since the state space is dynamically changing in our problem, the original belief reinvigoration strategy of POMCP does not work as it requires a fixed set of states. We therefore propose a novel belief reinvigoration strategy that allows for an efficient path planning as the robot builds up the 2D map of the environment. Once the POMCP module locates the target, the robot approaches it following the shortest path estimated by a local planner.

POMP++ is evaluated on public benchmark datasets including Active Vision Dataset (AVD) (Ammirato et al., 2017) with sequences acquired by real robotic platforms and the Habitat ObjectNav Challenge (Batra et al., 2020) with observations rendered from real 3D scans. Our approach outperforms baselines and state-of-the-art (sota) approaches with a significant improvement of $>10\%$ in terms of the success rate, without performing any costly offline training.

Our main technical contributions can be summarised as :

1. We formalize the AVS problem in unknown environments as a POMDP;
2. We integrate frontier-based exploration with POMCP planning for the AVS task, allowing the state space of the model to dynamically expand as the robot moves;

3. We propose a novel two-step belief reinvigoration strategy to address the dynamic state-space within the POMCP framework.

3.2 Related works

We cover closely related works addressing the AVS task and the development and application of POMCP to exploration or vision domains.

3.2.1 Active Visual Search

Earlier works addressing the AVS task often exploit target-specific inferences, such as object co-occurrences (Aydemir et al., 2013; Garvey, 1976; Kollar and Roy, 2009; Sjöö et al., 2012; Wixson and Ballard, 1994), while recent works exploit Deep Reinforcement Learning techniques (Chaplot et al., 2020a; Schmid et al., 2019; Ye et al., 2019), where visual neural embeddings are often used for policy training. EAT (Schmid et al., 2019) performs feature extraction from the current RGB image, and the image crop of the candidate target generated by a Region Proposal Network (RPN). The features are then fed into the action policy network. EAT is trained with twelve scans from the AVD (Ammirato et al., 2017) dataset. Similarly, GAPLE (Ye et al., 2019) uses deep visual features enriched by 3D information (from depth) for training the policy with a large dataset from 100 synthetic scenes. Recent benchmarking efforts offer larger amount of data for training and foster more data-driven AVS solutions. For example, RoboTHOR (Deitke et al., 2020) provides observations rendered from photo-realistic rooms (Deitke et al., 2020), while Habitat ObjectNav Challenge (Batra et al., 2020) provides renderings from real 3D scans. SemExp (Chaplot et al., 2020a), the current best-performing method on the ObjectNav Challenge, takes RGB-D frames as input and learns a 3D scene semantics representation, which can then be used for selecting a long-term goal for reaching the target object. SemExp proved that the 3D scene semantics provide more efficient guidance compared to the runner-up solution driven by scene exploration.

In general, learning-based strategies leverage large amounts of data to learn a model of the environment together with the motion policy, while online motion planning strategies have the advantage of being general and easy to deploy. A recent POMCP-based solution (Wang et al., 2020b) is able to achieve comparable search performance against Sota data-driven methods without any offline training. However, this approach is only applicable to known environments, i.e. the 2D floor map of the scene is required. Instead, our method POMP++ follows the idea of online policy learning and further proposes novel adaptations in terms of map representation and belief reinvigoration, which address unknown environments.

3.2.2 POMCP

Partially Observable Markov Decision Process (POMDP) is an established framework for sequential planning under uncertainty (Kaelbling et al., 1998). It allows to model dynamical processes in uncertain environments, where the uncertainty is related to actions and observations. While computing exact solutions for large POMDPs is computationally intractable (Papadimitriou and Tsitsiklis, 1987), impressive progress has been made by approximated (Hauskrecht, 2000) and online (Ross et al., 2008) solvers. One of the most used and efficient online solvers for POMDPs is Partially Observable Monte Carlo Planning (POMCP) (Silver and Veness, 2010) which combines a Monte Carlo update of the belief state with a *Monte Carlo Tree Search (MCTS)* based policy (Browne et al., 2012; Coulom, 2006; Kocsis and Szepesvári, 2006), thus enabling the scalability to large state spaces. Most recent extensions of POMCP include applications to multi-agent problems (Amato and Oliehoek, 2015), reward maximisation with cost constraints (Lee et al., 2018) and the introduction of prior knowledge about state space to refine the belief space and increase policy performance (Castellini et al., 2019).

We advance the state of the art by applying POMCP to AVS in unknown environments by proposing mechanisms to extend the state space, the transition and the observation models step by step. To address unknown environments, we exploit the elements of the frontier-based exploration theory (Yamauchi, 1997). Frontier-based exploration has been merged with POMDPs (Lauri and Ritala, 2016), but only for the exploration task. We instead focus on the AVS task and propose a novel POMDP formalisation and belief reinvigoration strategy to benefit effective path planning.

3.3 Proposed Method

We consider the scenario where a robot moves in an *unknown* environment, with the task of searching for a specific object. The robot explores the environment to: *i)* observe the target object, *ii)* localise the object in the floor map, and *iii)* finally approach the object, *i.e.* moving close to the object location. Each search episode is terminated once the robot believes the target object is sufficiently near to itself and visually detectable[1]. We formulate the AVS problem as a POMDP and employ

[1]The exact stop condition depends on the specific application. In the result section we consider standard evaluation protocols used in public benchmarks for AVS evaluation.

POMCP to compute the planning policy online. In the following section, we use the subscript t to represent the elements at the current time and the subscript $t+1$ to represent the updated elements after a new sensor capturing and observation.

Our overall framework is shown in Fig. 3.2. At each time step t the robot locates at a pose $p_t = \{x_t, y_t, \theta_t\}$, where x and y are the coordinates on the 2D floor plane, and θ is the orientation. From that pose it receives a new capturing from a RGB-D sensor for updating the environment map $\mathscr{M}_t$, which encodes all the information about the environment that the robot has gathered until t. The environment map is used to: 1) create a pose graph $\mathscr{G}_t$, that includes all the reachable poses of the robot, 2) extract the candidate locations of the target object $\mathscr{C}_t$, namely the frontier positions bordering the explored and unknown cells, 3) update the internal states of the POMDP, and reinvigorate the belief to make it cover newly discovered candidate locations. Using the planning policy calculated by the POMCP exploration module, the robot reaches a new pose p_{t+1} by taking an action a_t. The process repeats until the target object is detected. Once it locates the object, the robot will approach it following the shortest path between the pose of the robot and the estimated position of the object, on the graph $\mathscr{G}_t$[2].

In section 3.3.1 we present our new POMDP formulation that addresses the problem of exploring a search space that is initially unknown, and grows as the robot discovers new parts of the environment. In section 3.3.2 we detail how we use partial information about the environment to define the search space in POMCP to extract the exploration policy. We also show how the map of the scene can be updated every time when the robot discovers new parts of the environment. Finally, section 3.3.3 focuses on a new belief reinvigoration strategy for updating the belief when new states are discovered. This is a key contribution of our approach since the state space in the standard POMCP is fixed and only the probability over states is updated step by step.

3.3.1 POMDP formulation for Active Visual Search

The POMDP used in our context is a tuple $(S, A, O, T, Z, R, \gamma)$ where

- S is a finite set of partially observable *states* representing combinations of robot poses and target object locations, whose cardinality is unknown as the environment is unknown;
- A is the finite set of robot *actions*;

[2]We apply the same approaching strategy, i.e. robust visual docking, as in (Wang et al., 2020b) that allows path re-planning with continuous observations to be more robust to poor object detection.

- O: $S \times A \to \Pi(Z)$ is the *observation model* (with $\Pi(Z)$ probability distribution over observations in Z);
- T: $S \times A \to \Pi(S)$ is the *state-transition model* (with $\Pi(S)$ probability distribution over states in S);
- Z is a finite set of *observations*, namely, 1 if the searched object has been observed, 0 otherwise;
- R: $S \times A \to \mathbb{R}$ is the *reward function*, which is positive when the object is detected, and a negative when the robot moves in position where the object is not detected;
- $\gamma \in [0,1)$ is a *discount factor*, which is a constant that is multiplied by the reward function at each step which is used to prioritize early rewards instead of later ones.

The goal is to maximise the expected total discounted reward $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$, by choosing the best action a_t in each state s_t. The discount factor γ reduces the impact of future rewards and ensures that the (infinite) sum convergences to a finite value. The partial observability of the state is dealt with by considering at each time step a probability distribution over all the states, called *Belief* $B \in \mathscr{B}$. POMDP *solvers* compute, in an exact or approximate way, a *policy*, *i.e.* a function π: $\mathscr{B} \to A$ that maps beliefs to actions. We use POMCP (Silver and Veness, 2010) to compute the policy online. It is a recent algorithm employing Monte-Carlo Tree Search (MCTS) to solve POMDP. Every time an action has to be selected by the robot, POMCP performs controlled simulations using the known transition and observation models, and from those simulations, it learns the approximate value of each action. Then it selects the action with the highest value. The algorithm uses a particle filter to represent the (approximated) belief and MCTS to drive the simulation process in an optimal way.

3.3.2 Adaptation to unknown environments

In unknown environments, the state space S can change dynamically because all the possible robot poses and object locations are not known *a priori* but instead discovered over time as the robot explores the environment. Here we describe how the main elements in the state space S and, consequently, the transition model T, the observation model O, and the belief B are updated at each time step.

The pose graph $\mathscr{G}$ is composed of nodes representing possible robot poses and edges connecting poses reachable by the robot with a single action. We use $V^{\mathscr{G}}$ and $E^{\mathscr{G}}$ for the set of nodes and edges in $\mathscr{G}$ respectively. The pose graph $\mathscr{G}$ initially contains

Algorithm 1 Map update after each step. We omit the subscript t or $t+1$ for simplicity.

```
procedure MapUpdate(map $\mathscr{M}$, pose $p$)
    retrieve image from RGB-D sensor from pose $p$
    $rec \leftarrow$ scene reconstruction from RGBD Image
    for all cells $c \in \mathscr{M}$ do
        if $c$ in camera frustum then
            if $c$ is unoccupied in $rec$ then
                $c \leftarrow seen$
            else
                $c \leftarrow obstacle$
    for all cells $c \in \mathscr{M}$ do
        if $c$ not camera frustum then
            if $c$ is unoccupied in $rec$ then
                if $c$ is adjacent to $seen$ cell then
                    $c \leftarrow candidate$
                else
                    $c \leftarrow unknown$
    return $\mathscr{M}$
```

only poses in the field of view of the robot, while new nodes and edges are added as the exploration proceeds. The calculation of the reachable poses from the current camera view is solved by scene reconstruction methods with the depth maps (Choi et al., 2015).

We represent the *environment map* $\mathscr{M}$ as a grid that covers the whole environment containing observed and unseen parts of the scene. For ease of computation, we do this by using a very big grid size, but this can be easily adapted to a small grid that grows in size when needed. We assign each cell of the map $\mathscr{M}$ with one of the four possible values: *i) occluder*, the cell in the real environment contains any occluders, e.g. walls, preventing the robot from seeing what is behind the cell; *ii) seen*, a location that could contain the target but has been observed and found empty by the robot; *iii) candidate*, the cell is unknown but adjacent to a *seen* cell, *i.e.* a frontier cell; *iv) unknown*, the cell is unseen and is not a candidate cell. In the beginning, all the cells outside the field of view are initialised as *unknown*. Cell values are then updated step by step as the robot explores the environment (see Algorithm 1).

With the updated environment map $\mathscr{M}_{t+1}$, we can update the pose graph $\mathscr{G}_{t+1}$. For each node $p_i \in \mathscr{G}_t$, if taking action a_t leads the robot to a new pose p_j in the environment which falls into a *seen* cell of the updated map $\mathscr{M}_{t+1}$, then a node for pose p_j is added to $\mathscr{G}_{t+1}$ with an edge (p_i, p_j).

Algorithm 2 POMCP exploration. We omit subscripts t and $t+1$ for simplicity.

```
procedure POMCP-EXP(initial pose $p$)
    $\mathscr{M} \leftarrow$ init all cells to *unknown*
    $V^{\mathscr{G}} \leftarrow \{p\}$
    $E^{\mathscr{G}} \leftarrow \emptyset$
    repeat
        $\mathscr{M} \leftarrow$ MAPUPDATE$(p)$
        $V^{\mathscr{G}} \leftarrow V^{\mathscr{G}} \cup$ {poses related to *seen* cells in $\mathscr{M}$}
        $E^{\mathscr{G}} \leftarrow$ edges between $p_i, p_j \in V^{\mathscr{G}}$
        $C \leftarrow$ {candidate cells in $\mathscr{M}$}
        $S \leftarrow$ states considering updated $V^{\mathscr{G}}$ and $C$
        $T \leftarrow$ transition model from $E^{\mathscr{G}}$
        $O \leftarrow$ visibility function on $\mathscr{M}$
        $B \leftarrow$ BELIEF REINVIGORATION$(S)$
        $a \leftarrow$ POMCP$(S,T,O,p,B)$
        $p \leftarrow$ MOVE$(a)$
        $o \leftarrow$ DETECTOR                          ▷ true if object is detected
    until $o =$ True
```

We update the visibility function at each time step t as $v_t : V_t^{\mathscr{G}} \times C_t \rightarrow \{0,1\}$, where $V_t^{\mathscr{G}}$ is the set of nodes in $\mathscr{G}$ at time t and C_t is the set of *candidate* cells, i.e. candidate object locations at time t. The function takes a robot pose $p \in V_{\mathscr{G}}^t$, and a candidate object location $c \in C_t$, and returns 1 if c is visible from p (*i.e.* no obstacle lying between p and c in the current map $\mathscr{M}_t$), 0 otherwise. Notice that, the aim of function v_t is to address pose-cell visibility for all robot poses in $V_t^{\mathscr{G}}$ and all *candidate* cells.

POMP++ uses the elements described above to compute the optimal policy. Poses in $V_t^{\mathscr{G}}$ along with the *candidate* cells in $\mathscr{M}_t$ are used to build the state set S_t; *edges* in $E_t^{\mathscr{G}}$ are used in the transition model T_t, and the visibility function v_t is used by the observation model O_t in the simulation phase. Each state in S_t represents the target object in a specific candidate cell. Algorithm 2 describes how these states are updated during the exploration. At each step t, POMCP executes β number of simulations to compute the optimal action to perform in the real environment. Each simulation uses a specific state, which assumes the object is in a specific *c*andidate cell. During the simulation the environment is not updated as there is no observation in the real environment. After each step, the observation model O_t predicts if the robot sees the target object from the current pose and a reward is assigned to the action a_t. The *Reward* for a_t is a high positive value if the robot moves in a position where it can detect the object, otherwise, a small negative value is used to penalize longer paths.

Once the optimal action is identified by the simulations, the robot moves in the real environment and observes the object using an object detector. If the object is detected, the POMCP exploration terminates and the robot approaches the object with the robust visual docking strategy as in (Wang et al., 2020b). Otherwise, the environment map $\mathscr{M}_t$ and pose graph $\mathscr{G}_t$ are updated, and POMCP iterates until the maximum number of allowed steps μ is reached.

3.3.3 Belief reinvigoration for dynamically growing state space

POMCP assigns the states to simulations by sampling particles b from Belief B, which is updated over time when the robot makes an action and performs an observation in the real environment.

In standard POMCP, the belief update is performed considering only particles of the previous belief that are sampled by simulations whose first observation is the one observed in the real environment after performing the optimal action. After the belief update, to avoid particle deprivation, new particles are added to the updated belief B_{t+1} by sampling from the previous B_t after executing the chosen action a_t. In such a way, belief B_{t+1} tends to have a much higher probability in states that were already present before the reinvigoration, thus discouraging actions towards newly added states, as can be seen in Figure **??**.

Moreover, when dealing with an unknown environment where the state space changes over time, the particles at time t do not account for the part of the environment that will be observed at time $t+1$. Hence the states in the particle filter, after the traditional belief update, cannot represent correctly the state space at time $t+1$. To address the above-mentioned limitations of the belief update strategy in standard POMCP, we propose a two-step belief reinvigoration strategy. First, we map all the states in B_t to B_{t+1}, and remove the particles in simulations with their simulated observation different from the one made by the robot in the real environment. In this way, states in B_t are not penalized in B_{t+1} if they are not used as initial states by the simulations. Second, we add a new set of particles to B_{t+1} sampled with a uniform distribution over the new candidate cells introduced at time $t+1$. Note that without adding these states to B_{t+1}, simulations in the next steps will assume that the object could never be in one of these newly discovered parts of the environment. The proposed strategy allows to tune the percentage of new particles introduced in the second step with respect to the previously discovered candidate cells. By doing so, we can tune the attention the robot puts on the newly discovered frontier cells.

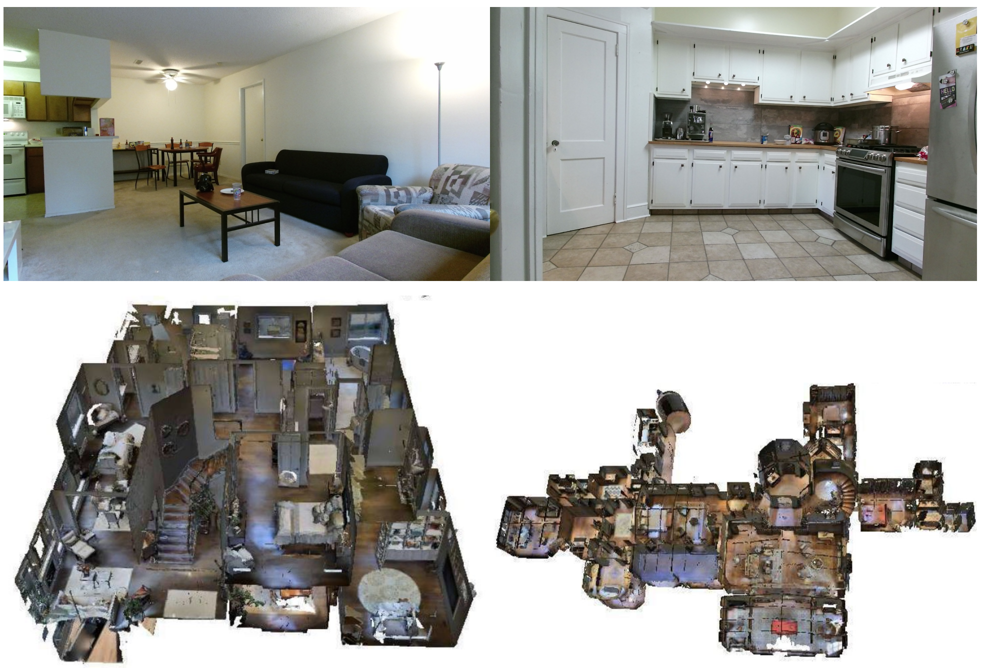

Fig. 3.3 Evaluation dataset for AVS methods. The upper row shows sample images in *AVD*, captured with a real robotic platform in real apartments. The lower row shows sample 3D scans of complex scenes in *Habitat ObjectNav* Challenge.

3.4 Experiments

We evaluate our proposed POMP++ on two public datasets featuring real-world scenarios that have been commonly used for benchmarking methods addressing the AVS task. Active Vision Dataset (AVD) (Ammirato et al., 2017) is captured with an RGB-D sensor equipped on a robotic platform moving within real apartments in a grid manner, while Habitat ObjectNav (Savva et al., 2019) provides a simulator for rendering observations within 3D scans of real large indoor environments provided by Matterport3D dataset (Chang et al., 2017) (see Fig. 3.3). We compare POMP++ against baselines and sota methods with both AVD (Section 3.4.1) and Habitat ObjectNav (Section 3.4.2) under their corresponding evaluation protocol and performance metrics for fair comparison. Moreover, we conduct an ablation study (Section 3.4.3) where we show the impact of the proposed belief reinvigoration strategy and the impact of an imperfect object detector on AVS planning.

3.4.1 Evaluation on AVD

AVD is the largest real-world dataset available for testing AVS, containing 17 scans of 9 real apartments recorded by a robot equipped with an RGB-D camera. For each scene, AVD provides the RGB-D images captured from all possible robot poses, as well as the camera intrinsics and extrinsics. The action space A in AVD is $\{$`Forward`, `Backward`, `Turn_Left`, `Turn_Right`, `Stop`$\}$, with a translation step of 30 cm and a rotation step of 30°.

Performance Metrics In the AVD Benchmark (AVDB), a run is considered successful when the robot terminates its exploration in one of the ending positions specified in AVDB, which are closest to the object with the object appearing in the camera view. There are three main metrics defined for benchmarking different methods:

- **Success Rate (SR)**, is the ratio of successful episodes over the total number of episodes.
- **Average Path Length (APL)** is the average number of poses visited by the robot among the paths that lead to a successful search (a lower value indicates a higher efficiency).
- **Average Shortest Predicted Path Length (ASPPL)** is the average ratio between the length of the shortest possible path to reach a valid destination pose provided by AVDB and the length of the path generated by the method (a larger value indicates a higher absolute efficiency). We also compute the standard deviation of ASPPL to investigate the variability of path efficiency.

Baselines and Comparisons We compare POMP++ against:

- **Random Walk**: a standard baseline where the robot move randomly for a certain number of steps.
- **EAT** (Schmid et al., 2019): a reinforcement learning approach trained on the remaining scenes of AVD. A Region proposal network (RPN) extracting object proposals is combined with a DRQN for policy learning.
- **POMP** (Wang et al., 2020b): a POMCP-based solver that only works with known environments. This method requires as input the 2D floor map and the reachable robot poses.

Note that among the RL-based strategies, we consider EAT (Schmid et al., 2019) and GAPLE (Ye et al., 2019) that both use AVD for evaluation (discussed in Sec. 3.2). However, the evaluation protocol adopted by GAPLE is not documented, while EAT (Schmid et al., 2019) shared their protocol explicitly. For a fair comparison, we follow the evaluation protocol in both EAT (Schmid et al., 2019) and POMP (Wang et al., 2020b), which perform a search on a subset of objects under three test scenes with an increasing difficulty level. The test scenes are: *i*) Home_005_1, the easy scenario, consists only of a kitchen; *ii*) Home_001_2, the medium scenario, the robot has to navigate a living room and an open kitchen; *iii*) Home_003_2 is the hard scenario, where the robot has to explore a living room, a half-open kitchen, a dining room, a hallway, and a bathroom. As in(Schmid et al., 2019) and (Wang et al., 2020b), to only compare the goodness of the motion policy without the nuisances caused by underlying object detectors, we use the ground-truth annotations of the object for the evaluation, and we limit the search to 125 steps. For POMP++, we set the reward value to +1000 for when the object is detected and -1 for when it is not. The impact of a real-world detector on AVD is the subject of another experiment in Section 3.4.3.

Results Discussion Table 3.1 shows the performances achieved by all methods on the three test scenes in AVD. We observe that, on average POMP++ outperforms all the other methods in terms of Success Rate. Even in complex episodes, where the target object is initialized far away from the robot, POMP++ is able to succeed in the search task. Our path efficiency on the successful searches is lower compared to other methods as POMP++ has to explore more areas to look for the object, while POMP (Wang et al., 2020b) exploits a known 2D floor map and EAT has already encoded the environment knowledge via its offline training. The improvement of POMP++ in terms of SR over POMP is mainly contributed by our new belief reinvigoration strategy. We provide more details in Section 3.4.3.

Table 3.1 Results on the three test scenes from AVD using the GT annotations for object detection. We report the results of EAT (Schmid et al., 2019) and POMP (Wang et al., 2020b) as in their original paper. Results are averaged over multiple search episodes as defined in AVD benchmark, with the standard deviations presented within the parentheses. Results using known floor maps are highlighted in *italic*. Best results of methods without known floor maps are highlighted in **bold**.

METHOD	AVD DATASET		
	EASY		
	SR ↑	APL ↓	ASPPL ↑
POMP (known env) (Wang et al., 2020b)	*0.98*	*13.6*	*0.72 (0.26)*
Random Walk	0.32	74	0.19 (0.35)
EAT (Schmid et al., 2019)	0.77	**12.2**	-
POMP++	**1.0**	14.27	**0.70 (0.28)**
METHOD	MEDIUM		
	SR ↑	APL ↓	ASPPL ↑
POMP (known env) (Wang et al., 2020b)	*0.73*	*17.1*	*0.8 (0.23)*
Random Walk	0.11	74.48	0.21 (0.36)
EAT (Schmid et al., 2019)	0.73	**16.2**	-
POMP++	**0.79**	19.4	**0.76 (0.26)**
METHOD	HARD		
	SR ↑	APL ↓	ASPPL ↑
POMP (known env) (Wang et al., 2020b)	*0.56*	*20.5*	*0.72 (0.39)*
Random Walk	0.10	79.27	0.17 (0.18)
EAT (Schmid et al., 2019)	0.58	**22.1**	-
POMP++	**0.84**	29.54	**0.61 (0.33)**
METHOD	AVERAGE		
	SR ↑	APL ↓	ASPPL ↑
POMP (known env) (Wang et al., 2020b)	*0.76*	*17.1*	*0.75 (0.29)*
Random Walk	0.18	75.91	0.19 (0.29)
EAT (Schmid et al., 2019)	0.69	**16.8**	-
POMP++	**0.87**	21.07	**0.68 (0.29)**

3.4.2 Evaluation on Habitat ObjectNav

Habitat (Savva et al., 2019) is a simulation platform designed for embodied AI, where robots can move and observe within realistic 3D environments, *e.g.* 3D scans of real large scenes from Matterport3D (Chang et al., 2017) and Gibson (Xia et al., 2018). In particular, the Habitat ObjectNav Challenge makes use of 11 scans of complex scenes in Matterport3D for validating methods (see the second row in Fig. 3.3). While the robot moves in the Habitat environment, the simulator renders the RGB-D images at each step taken from the current pose, as well as the camera intrinsics and extrinsics. The action space A in Habitat ObjectNav Challenge is the set of four actions $\{$`Forward`, `Turn_Left`, `Turn_Right`, `Stop`$\}$. We set the translation step to 25 cm and the rotation step to 30°.

Performance Metrics In the Habitat ObjectNav Challenge, a success is defined if the distance between the robot and the target object is $\leq$ 1m, and the object is within the camera's field of view, without specifying the ending poses as in AVD. There are three main performance metrics defined for the Habitat ObjectNav Challenge:

- **Success Rate (SR)**, is the ratio of successful runs over the total number of runs, the same as in AVD metrics;
- **Success weighted Path Length (SPL)**, measures the path efficiency of successful episodes[3], defined as $SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(l_i^r, l_i)}$, where l_i is the length of the shortest path between the goal and the target for an episode, l_i^r is the length of path taken by a robot in an episode, and S_i is the binary indicator of success in episode i. Finally,
- **Distance To Success (DTS)** is further defined by (Chaplot et al., 2020a), which measures the distance of the robot from the success threshold boundary when the episode ends. DTS is computed as $\max(\|x_T - x_G\| - d_s, 0)$, where x_T is the robot location at the termination, x_G is the goal location at the end of the episode, and d_s is the success threshold.

Baselines and comparisons We compare POMP++ against the winner entry in the Habitat ObjectNav Challenge, **SemExp** (Chaplot et al., 2020a), as well as baselines reported in the literature:

- **Random walk**: standard baseline where the robot moves randomly for a certain number of steps.

[3]SPL is used as the main criterion in the Habitat ObjectNav Challenge

Table 3.2 Results of POMP++ and baselines evaluated on 11 test scenes of Matterport3D, with the object detector pretrained on MS-COCO.

	Habitat		
Method	SPL ↑	SR ↑	DTS (m)↓
Random	0.005	0.005	8.048
RGBD + RL (Savva et al., 2019)	0.017	0.037	7.654
RGBD + Sem + RL(Mousavian et al., 2019)	0.015	0.031	7.612
Classical Mapping + FBE (Yamauchi, 1997)	0.117	0.311	7.102
Active Neural SLAM(Chaplot et al., 2020b)	0.119	0.321	7.056
SemExp (Chaplot et al., 2020a)	0.144	0.360	6.733
POMP++	**0.148**	**0.420**	**3.726**

- **RGBD + RL** (Savva et al., 2019): a vanilla recurrent RL policy initialised with ResNet18 backbone followed by a GRU.
- **RGBD+Semantics+RL**: an adaptation from (Mousavian et al., 2019) which passes semantic segmentation and object detections along with RGBD input to a recurrent RL policy.
- **Classical Mapping + FBE** (Yamauchi, 1997): classical robotics pipeline for mapping followed by frontier-based exploration (FBE). When the detector finds the object, a local policy reaches the object with an analytical planner.
- **Active Neural SLAM** (Chaplot et al., 2020b): an exploration policy trained to maximise the coverage. Whenever the target object is detected, the same local policy as described above is applied to approach the object.
- **SemExp** (Chaplot et al., 2020a): a RL policy learner with a semantic mapping of the explored environment and embedded object-to-object priors.

We follow the evaluation protocol defined in (Chaplot et al., 2020a), where a subset of object categories that are common between MP3D and MS-COCO is chosen as target: $\{chair, couch, potted\,plant, bed, toilet, tv\}$. For object detection, we used Mask-RCNN with ResNet50 backbone pretrained on MS-COCO. Experiments are performed with 11 scenes in the validation set of Matterport3D, following the configuration in terms of robot starting pose and target object, provided by the Habitat ObjectNav Challenge. The walls are reconstructed online with the ground-truth semantic segmentation. The reward value is set to +1000 for when the object is detected and -1 for when it is not.

Result Discussion Table 3.2 shows the results evaluated on the 11 test scenes in Habitat ObjectNav dataset. Our approach has a SR 6% higher than the best RL approach SemExp (Chaplot et al., 2020a) and 10% higher than the Active Neural SLAM (Chaplot et al., 2020b). In terms of SPL, we are just slightly better than SemExp, which indicates that our path efficiency could be lower given the higher SR we achieve. Yet, our Distance To Success (DTS) is notably lower than all the other approaches, meaning that our failed episodes tend to terminate nearer to the target location.

It is noticeable that the overall SR achieved by all methods are low due to the unsatisfactory 3D reconstruction quality of scenes (*e.g.* incomplete scans) and sometimes confusing object annotations (*e.g.* "bulletin board" as "tv_monitor"). The use of a standard object detector without fine-tuning on the rendered images also contributes to the low performance since the detector is prone to errors. The impact of the object detector on AVS task is studied in Section 3.4.3.

Moreover, we notice that algorithms that use explicit semantic mapping of the environment, such as Classical Mapping+FBE (Yamauchi, 1997), Active Neural SLAM (Chaplot et al., 2020b), SemExp (Chaplot et al., 2020a) and POMP++ are able to perform significantly better than RL-based methods (Savva et al., 2019) that rely on implicit scene representation via visual features extracted from the sequence of observations. However, the scenes in Habitat ObjectNav varies a lot among each other in terms of the scene types and dimensions, covering modern/ancient homes, palaces, and even a spa. The merits of encoding sophisticated scene semantics can be limited, which allows POMP++ to outperform other methods with only 2D wall maps that are reconstructed online.

3.4.3 Ablation study

We mainly evaluate the effectiveness of the proposed belief reinvigoration strategy and the impact of imperfect object detector. We compare POMP++ to POMP (Wang et al., 2020b) and POMP+, which is POMP for known environments with our novel belief reinvigoration strategy, with both ground-truth annotation and the object detector provided by AVD (Ammirato et al., 2017). Table 3.3 reports the averaged results obtained with the hard scene Home_003_2 in AVD. POMP+ with the novel belief reinvigoration strategy improves the SR by 30%, although at the cost of lengthier paths. SR is significantly improved by the new reinvigoration strategy because the original one tends to vanish particles that are not covered by successful simulations, which limits the potential of a successful search, as we explained in Section 3.3.3. POMP++

Table 3.3 Results of POMP++ and POMP+ on the Hard scenario from AVD, with both GT annotations and an object detector (Ammirato et al., 2017).

	AVD (Hard)		
Method	SR ↑	APL ↓	ASPPL ↑
POMP (GT, *known env*)	0.56	20.05	0.72(0.39)
POMP+ (GT, *known env*)	0.86	33.71	0.55(0.35)
POMP++ (GT)	0.84	29.54	0.61(0.33)
POMP (Detector, *known env*)	0.37	22.01	0.63(0.29)
POMP+(Detector, *known env*)	0.45	23.95	0.63(0.41)
POMP++ (Detector)	0.41	22.41	0.63(0.31)

achieves a slightly worse SR compared to POMP+ due to the lack of a known 2D floor map.

Moreover, the performance of the detector imposes a direct impact on the AVS performance, where we can observe the SR is reduced almost by half compared to the results obtained with GT annotations. This is because during the POMCP exploration, the output of the detector is used to either prune candidate object location in case of no detection; or as a stopping condition for the exploration if the target object is detected.

3.4.4 Real-time performance analysis

The complexity of POMP++ is $O(\alpha \times (\beta \times \mu + \delta))$, where α is the number of steps performed in the real environment, β is the number of simulations performed for each step, μ is the max number of steps per simulation, δ is the number of actions to update the state space, the map, the transition and observation models. Note that all parameters can be tuned to satisfy real-time performance when scaling the algorithm to large environments. When experimenting on AVD that contains realistic indoor housing environments, *e.g.* typical homes, we achieve 0.07 seconds per step on average with a standard machine with a 6-cores Intel i7-6800k CPU. Such processing speed is considered sufficient for real-time robotic applications.

3.5 Conclusion

In this chapter, we presented a POMCP-based planner, POMP++, that solves the AVS problem in unknown environments without the need for offline training. We introduced novel modifications to the POMDP's formulation, allowing for the search space to be dynamic, and grow as the robot explores the environment. Following this new formulation, we proposed a novel way to update the belief in the underlying POMDP solver, POMCP, which boosts the correct coverage of the expanding search space. We outperformed the SOTA methods on two benchmark datasets in terms of the success rate by a significant margin.

3.5.1 Limitations

While a POMDP formulation allows for an explainable and training-free strategy for the planner, its use in real-world applications with real robots is not straightforward. The hardest part is building an accurate definition of the underlying model that describes the agent and the environment in which it operates. In this work, our model is quite simple: we model the space as a grid and assume that the agent can move from one grid cell to a neighboring one with a single action. In practice, this is not always the case, as a diagonal movement would not lead to being in the center of the cell. This would need to be accounted for in a real application. Reinforcement Deep Learning can overcome this limitation by being a Model-Free solution for navigation, meaning that the model is learned implicitly by learning to explore the environment. Another limitation is that the proposed solution does not use prior knowledge during the object search. It does not account for whether we are in a room compatible with the searched object and does not check for co-occurring objects to improve its navigation capabilities. In the next chapter, we propose a way to overcome this last issue with an approach that allows us to estimate the location of an object by exploiting the location of other seen objects.