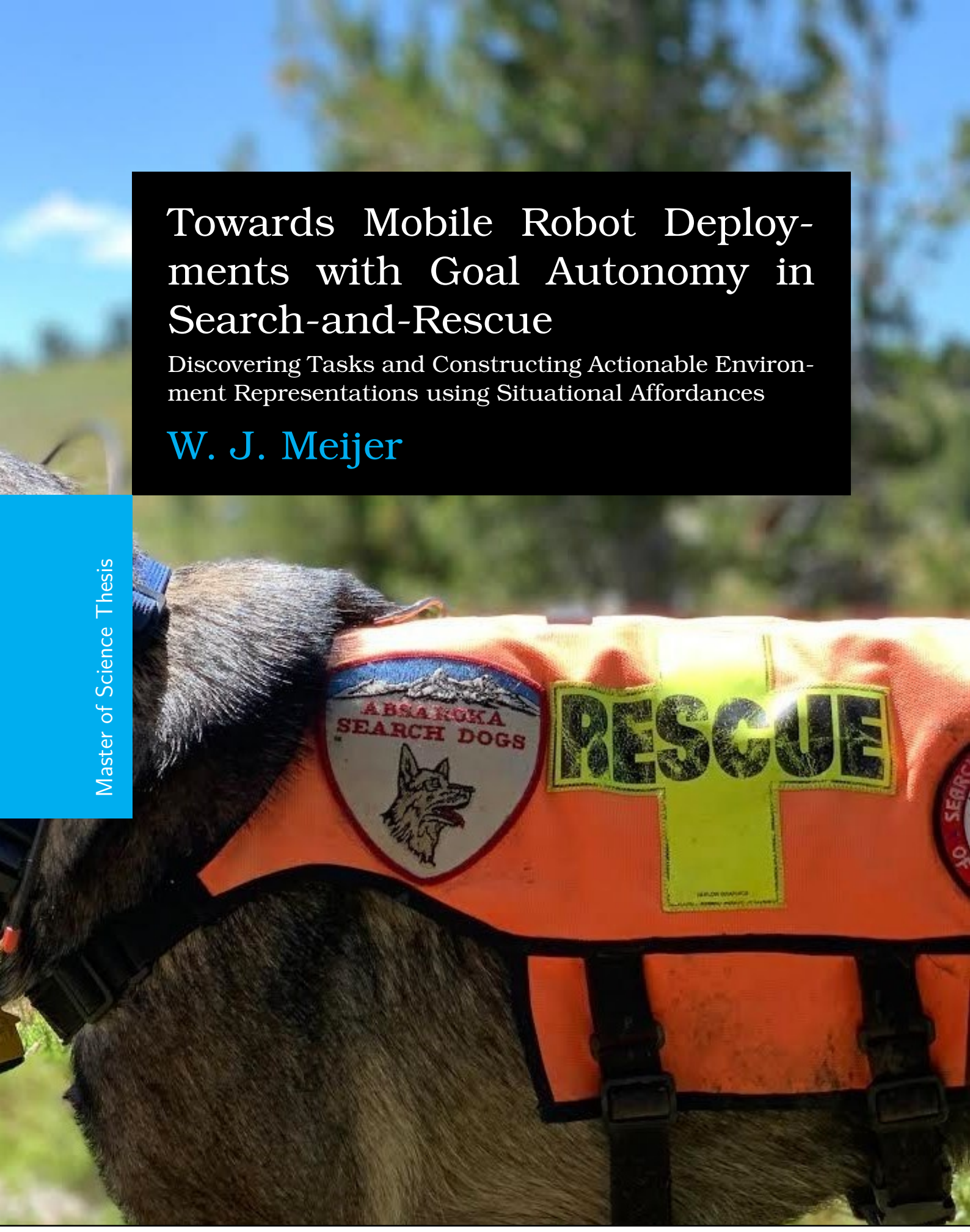
Towards Mobile Robot Deployments with Goal Autonomy in Search-and-Rescue

Discovering Tasks and Constructing Actionable Environment Representations using Situational Affordances

W. J. Meijer

Master of Science Thesis



TU Delft Delft University of Technology

Cognitive Robotics Department

Towards Mobile Robot Deployments with Goal Autonomy in Search-and-Rescue

Discovering Tasks and Constructing Actionable Environment Representations using Situational Affordances

Master of Science Thesis

For the degree of Master of Science in Cognitive Robotics at Delft University of Technology

W. J. Meijer

October 10, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by TNO and the AIRLab. Their cooperation is hereby gratefully acknowledged.

"Science can amuse and fascinate us all, but it is engineering that changes the world."

— *Isaac Asimov, Author of "I Robot" (1950), professor of biochemistry*

Abstract

In this work, we address the challenges of employing robots in the Search-and-Rescue (SAR) domain, where they can benefit rescue workers to quickly obtain Situational Awareness (SA). Missions with autonomous mobile robots are heavily dependent on environmental representations. Representations have been steadily increasing in the richness that they can capture, in addition to geometry we can now represent objects and their interrelations. These richer environmental representations, such as the 3D scene graph, have provided an opportunity to integrate representations with prior knowledge to make them more actionable and improve the SA they provide. The use of such representations for planning is limited. In previous work, the main approach was to augment and extend the scene graph to enable traditional symbolic planning. The limitations of these methods are that they are offline, scale poorly, and require full observability, making them unsuitable for the SAR domain. The main contributions of this work are as follows. First, we propose the behavior-oriented situational graph, a data structure that integrates data-driven perception with prior knowledge following a novel *situational affordance* schema. This schema connects situations with robot behaviors and mission objectives, allowing for autonomous mission planning. Second, we propose an efficient method to obtain task utilities from the proposed behavior-oriented situational graph through planning. Finally, we propose an exploration component to discover and select tasks online in dynamic environments that are potentially partially observable. This work is evaluated in several simulation scenarios, showing improved efficiency in mission completion compared to offline methods for the specific SAR domain. Finally, the methods are implemented and tested in a real-world scenario using a mobile Spot robot, showing its effectiveness in practice.

Preface

This thesis is carried out in the Intelligent Autonomous Systems group as part of an internship at TNO. I would like to thank my thesis advisors Joris Sijs, Corrado Pezzato, and Jeroen Fransman for their assistance and interesting conversations during the whole process. Furthermore, I would like to thank my colleagues at TNO and fellow students at the AIRlab for their thoughtful questions and interesting discussions.

In addition, I would like to thank my girlfriend, brother, and friends for engaging with my ideas and encouraging me throughout the whole process. Another big thank you to my mom and dad. For all the freedom you gave me to pursue my passions, do the things I love, and for the support I always felt.

Finally, I acknowledge the contribution made by the open source community.

Delft, University of Technology — W. J. Meijer
October 10, 2022

Table of Contents

Acronyms

2DSG 2D (Image) Scene Graph. 2, 3, 10

3DSG 3D (Spatial) Scene Graph. 2–4, 6, 34–36

AIRlab Artificial Intelligence for Retail lab. iii

API Application Programming Interfaces. 2, 54

BOS-Graph Behavior-Oriented Situational Graph. v, x, xii, xiii, 5, 7, 21, 22, 24–27, 29–31, 33–46, 49, 50, 52–54, 56–68, 71

BT Behavior Tree. 16

DARPA Defense Advanced Research Projects Agency. 1, 2

MDP Markov Decision Process. 18

NN Neural Network. 2

POMDP Partially Observable Markov Decision Process. 17

SA Situational Awareness. ii, xii, 1, 3–6, 8–12, 18, 19, 21, 22, 34, 38, 53, 65, 66

SAR Search-and-Rescue. ii, xiii, 1–4, 6, 9, 19, 21, 23, 24, 33, 37, 52, 53, 55, 56, 60, 64–69

SG Scene Graph. xiii, 2, 4, 13, 34, 35, 38, 66

SLAM Simultaneous Localization And Mapping. 2, 55

SNOW Safe autoNomous systems operating in the Open World. 1

TaMP Task and Motion Planning. 2, 17

TNO Technisch Natuurwetenschappelijk Onderzoeksinstituut. iii, 1, 59

Glossary

affordance Traditional affordances are the action possibilities that an object provides an agent and the effects thereof . 2, 19, 66

authoring incorporating prior-knowledge in the form of manually designing, coding, scripting, tuning and testing of software artefacts (ex. behaviors, pipelines, schemas). 2, 8, 16, 19, 30, 32, 37, 38, 66

autonomy (metric) Number of human interventions necessary as a function of the complexity of the environment and task domain. 12

environment representation In order for an artificial agent to operate effectively it needs to build some internal representation of the external environment. 5, 11, 19, 22, 34, 38, 52, 64, 65, 67

goal An abstract outcome, within a volume of time and space, that is broad and long-term. 24

goal autonomy An agents ability to pro-actively generate its own tasks given generic objectives. 5, 22, 65, 67

mission A set of goals. 22, 24

objective Defines measurable unparameterized plan outcomes in the short term that contribute to achieving the overall goal. 4, 24

observability *Full observability* indicates the the full state of the world is directly accessible by the agent. *Partial observability* indicates that the full state of the world is not accessible by the agent, only (local) parts of the world state. 3

plan a sequence of behaviors that is projected and parameterized using a model to achieve a task. 2, 24

plan domain A plan domain is defined over a mission and is defined as which sequence of behaviors is possible in a plan. 8, 70, 71

robustness The quality of a system to operate as desired in response to disturbances and uncertainty. 29

schema A database schema is a set of rules that governs the logical configuration of a database. All instances in the database have to be compliant with the schema.. xii, 27–29

situational affordance Situational affordances are the behavior possibilities that a situation provides an agent and the effects thereof on mission objectives. . 2, 14, 34, 53, 58, 67

task Defines measurable parametrised plan outcomes in the short term that contribute to achieving an overall goal. 2, 4, 24

Nomenclature

List of Symbols

F	Generic graph search algorithm
π	Plan
$\mathbb{P}$	Planning problem
$\mathbb{T}$	Task allocation problem
U	Utility function

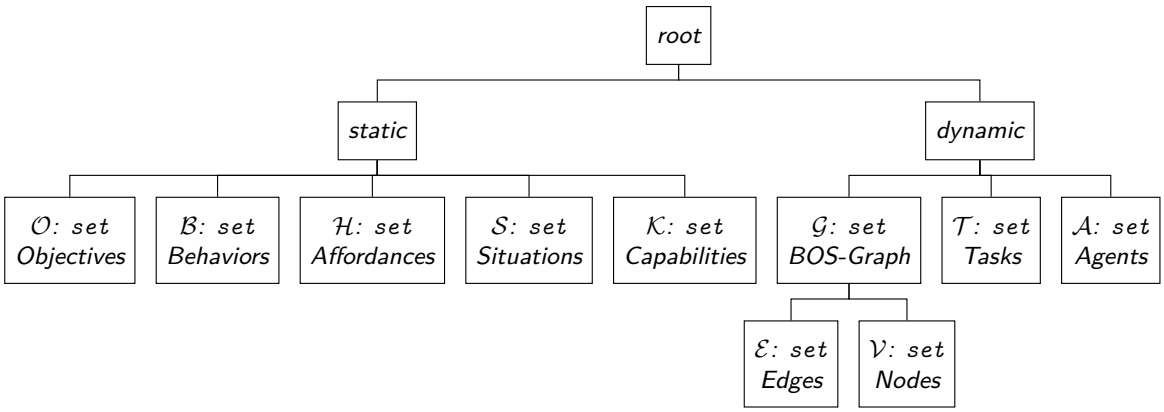


Figure 1: Overview of all sets introduced.

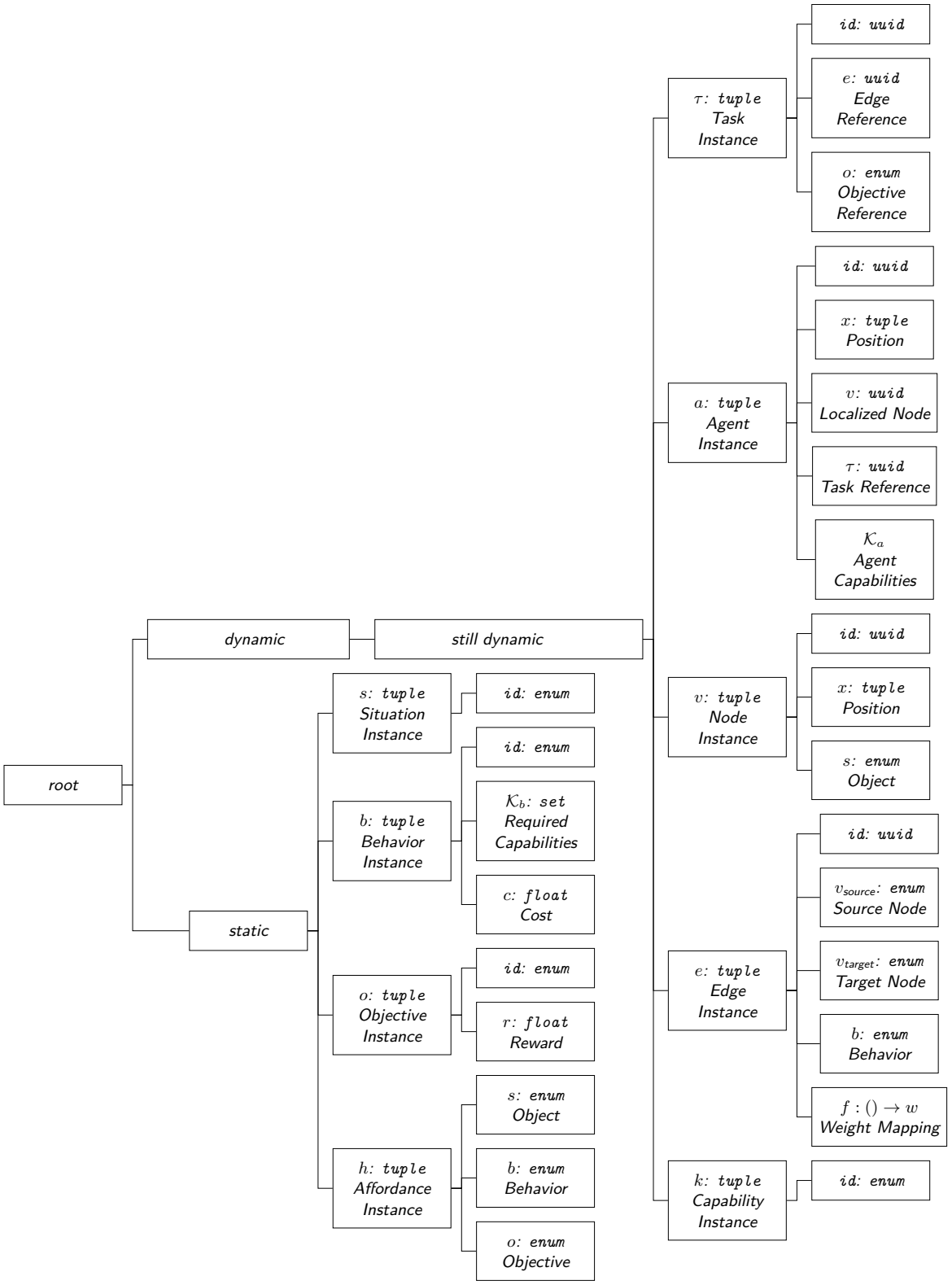


Figure 2: Hierarchical overview of all data introduced.

List of Figures

List of Algorithms

Chapter 1

Introduction

This introductory chapter focuses on the motivations behind this research, posing the three research questions that this thesis addresses. After briefly reporting on the current state-of-the-art in the research area of actionable environment representations for robotics, the main contributions of this work are highlighted. The chapter is then concluded with the outline of the document.

1-1 Research Motivation

Dogs have been man's best friend for 35,000 years [3]. They provide us with companionship and help us with important and intense work. Search-and-Rescue (SAR) missions are intense, and the environments in which they take place typically present significant challenges to quickly obtain important information, i.e. to obtain Situational Awareness (SA). In time-sensitive disaster scenarios, first responders and their dog companions commonly face increased technical challenges, including difficult terrain, unstable structures, degraded environmental conditions, and expansive areas of operation. We could help them by providing them with better tools to quickly build SA. Autonomous mobile robots are a prime candidate to be that tool and their effective deployment is the topic of active research [4, 5, 6]. To be successful in these deployments, many subsystems must work together. Countless 'really cool on paper' technologies are described in the robotics literature, but there is a severe lack of robots that can operate reliably for at least an hour without human intervention. This diminishes the primary benefit of freeing up or at least significantly augmenting human rescue workers' hands and minds.

Project SNOW is such a research project at TNO where the aim is to investigate methods to improve self-awareness, situational awareness, and tactical awareness of mobile robots through a SAR use case. Another prominent research project is the recent SubTerranean challenge [7] organized by Defense Advanced Research Projects Agency (DARPA), this was a competition where teams had to perform object location tasks in large-scale underground environments. The SubT challenge did not include manipulation tasks. Manipulation is

required to enable information gathering in certain closed-off areas. Sometimes we need to open doors, extinguish fires, or clear debris. This illustrates that reliable mobile manipulation in unseen environments remains primarily a purely academic endeavor.

The academic field that deals with mobile manipulation tasks is called Task and Motion Planning (TaMP). Here the aim is to jointly plan sequences of actions, *plans*, that achieve *tasks* while jointly considering the geometric motions required to execute these actions. TaMP methods suffer serious fundamental problems such as the *symbol grounding problem* [8, 9], the *frame problem* [10] and bad computational scaling of solvers. These downsides prohibit their value in real-world use cases. Due to these limitations, the companies producing the most commercially mature agile mobile robot platforms [11, 2, 12] focus on providing tools and APIs to manually script or *author* missions as linear paths through a known environment with predefined behaviors along the way. The missions are often authored using teleoperation, here the pilot controls the robot with a controller to author the mission. Assuming that the environment remains mostly static, the authored mission can be periodically played back. Considering our objective, freeing up the hands of rescue workers, this immediately shows why the commercially dominant mission authoring paradigm does not work for the SAR domain.

The DARPA SubT challenge led to a huge boom of publications on the systems engineering to reliably deploy robots as mobile sensor platforms in real-world indoor SAR scenarios [13, 14, 15, 16, 17, 18]. Consequently, it has been shown that robots can now generate extremely precise maps and that object detection is possible. However, there is a lack of methods for correlating the potential interactions of environmental objects. Consequently, despite robots' ability to create amazing, beautiful, and accurate maps, robots are not as good at comprehending the environment.

Affordances is the research field from environmental psychology that connects agents with the action possibilities that an environment provides and the effects of those actions. In robotics, this is often formalized as an (object, (action, effect)) tuple [1]. In this work, we investigate this from a novel *situational affordances* perspective, which we formalize as (situation, (behavior, objective)) tuples.

To actually link sensor data in the form of images and video with these situational affordances, 2D (Image) Scene Graph (2DSG)s and 3D (Spatial) Scene Graph (3DSG)s appear promising. 2DSG capture situations in images, while 3DSG aim to capture situations of complete environments. 3DSG were first proposed in 2019 [19]. Being a hierarchical graph composed of metric, semantic, and topological information 3DSGs aim to be the data structure that implements a unifying representation of the environment, with outstanding explanatory power regarding elements in the current situation.

The pipeline to create 3DSGs relies on robust Simultaneous Localization And Mapping (SLAM), semantic segmentation, and multi-view pose estimation to work together seamlessly. Modern SLAM techniques are powered by factor-graph optimization and effective fusion of different sensor modalities. Semantic segmentation and pose estimation are enabled by the Neural Network (NN) revolution of the past decade. SGs are relevant, not only due to their explanatory power regarding elements in the environment but also because they provide an opportunity to facilitate robot task planning.

From a SAR perspective, the further contextualization of such graph-based environmental representation pipelines through mission-specific prior knowledge could deliver a higher level

of situational awareness compared to existing representations. Additionally, it is expected that by offloading more responsibility to the perception system it should be possible to facilitate planning. In this thesis, we focus on how we can best combine the outputs of these data-driven pipelines with prior knowledge to facilitate robot planning to move toward effective deployments in unseen environments. We propose a new environment representation with an accompanying pipeline and demonstrate it on a physical robot.

1-2 State-of-the-art of actionable environment representations for search-and-rescue

Given the motivation to provide and acquire a higher level of SA for rescue workers while freeing up their hands, the following sections start at the research gap of **Situational Affordances** as an entry point for prior knowledge of the mission context. We then work back up through **Task Discovery** to handle partially observable environments, to **Goal Autonomy** which ensures we actually free up hands. Evaluating the research domain in this way will provide us with insight into the state-of-the-art for constructing and using actionable environment representations for SAR. Actionable refers to a low barrier to taking action. A more actionable environment representation will make it explicit what can be done next in the context of the deployment of the robot. It provides actionable insight into the environment. If the environment provides opportunities for the agent to perform useful behaviors, how may these behaviors be expressed explicitly?

Situational Affordances The main gap we identified in the literature is the lack of research into affordances at the situational level for robotics purposes. In robotics, affordances are mainly considered as low-level (effect, (object, action)) tuples. Scene graphs come in two flavors, 2D (Image) Scene Graph (2DSG) [20] and 3D (Spatial) Scene Graph (3DSG) [21, 22]. What they have in common is that scene graphs allow us to detect and classify situations. Because of scene graph developments it is increasingly possible to reason about scenes at the situation level instead of about scenes at the object level. This opens up an opportunity to consider situational affordances for robots. In psychology situational affordances are strongly context-dependent affordances [23], what is interesting is in our literature search we could not find them being used for robotics purposes in any earlier work. Situational affordances could be a useful framework to directly link perception to useful robotic behaviors that achieve generic objectives, which is the research objective of this thesis.

Task Discovery Having introduced the literature gap around situational affordances, which stand to link perception of situations with behaviors and tasks, we now review the state of the art for discovering and representing tasks. This is paramount for the requirement to handle unseen environments, in literature these are called *partially observable* environments. In order to fully observe an unknown environment, exploration is required. Exploration is a common research topic in robotics [24, 25, 15, 26], but it is almost always guided by geometric information gain. Semantic information gain exploration [27, 28, 29, 30, 31] is a novel research topic that aims to use higher level semantic features of the environment to guide information.

In [32] affordances are embedded in obstacles in a static map to allow the robot to plan a path that includes pushing and pulling obstacles out of the way. This is important for detecting situations where a task for a bulldoze robot could be clearing obstacles for another lighter-weight robot to continue its exploration.

In [33] the robot discovers tasks when it discovers a victim, these tasks are then assigned to the appropriate robot in its team. This is the only task it can discover, and this method directly links object detection with a task.

Then looking at task discovery from the context of SGs we see that Amiri et al. present object search guided by probabilistic conditioning using a globally constructed abstract scene graph [34]. This single-object search task is conducted online in a small partially observable environment. The limitations of this work are that while the object locations are unknown, the map of the world is fully known. Its only 6 discretized waypoints. In the context of SAR situational awareness, the resulting global scene graph is in an abstract space and provides almost no actionability. Also, the method only searches for a single object; it does not consider tasks.

Goal Autonomy Now if we combine the detection of task opportunities with the capability to discover and search environments, we have acquired the base components for a system with a limited sense of Goal Autonomy as described by Galindo and Saffiotti in [35]. In their work Goal Autonomy is described as: the robot's ability to proactively generate its own goals given generic motivations. We will update this definition to match the terms used in the rest of this work. We describe goal autonomy: the robot's ability to pro-actively generate its own tasks given generic objectives.

We have looked into methods that instead of directly acting upon the generated tasks, aggregate them in a comprehensive environmental representation. This allows for the prioritization or planning of subsequent tasks to guide the exploration vs exploitation tradeoff. As we have seen, 3DSGs claim to be some of the most actionable representations currently available. However, to enable planning over them, they need to be manually augmented quite intensively [36]. Taskography was discovered after starting on this thesis [36] by Agia et al., it presents a number of augmented 3DSGs to provide a planning benchmark. The main limitation with this work in the SAR context is that it is completely offline and assumes full observability. Furthermore, the planning is only conducted in simulation. This leaves a clear opportunity to create directly actionable environment representations online in real-world partially observable environments with a physical mobile robot. Furthermore, [36] proposes a method to prune scene graphs to make planning more effective. Also, this pruning is interesting, it is interesting to investigate which assumptions allow us to take this pruning to the max, and ideally prune at perception time.

Conclusion In the literature on scene graphs, the SA that is being extracted is not actionable beyond navigation purposes. Secondly, affordances are only used in the context of low-level actions and their low-level effects, not on their effects on higher-level objectives, in the context of mission goals. These problems and gaps provide opportunities for this research since solving them improves the autonomy of robots. To endow mobile robots with improved SA a promising approach is to aggregate mission-critical objective affordances in a map.

1-3 Research questions

This work aims to question the typical approach to world representation and planning. We investigate how more actionable representations of the environment can provide a higher level of situational awareness and free up rescue workers' hands through improved goal autonomy. The problem statement is formulated as follows:

How can we leverage prior knowledge and assumptions on robust perception of situations to construct and exploit more actionable environment representations?

To solve this problem, the following research questions will be addressed:

1. **Assuming that we can robustly perceive situations, how can we best structure prior knowledge to create an information pipeline to construct more actionable environment representations?**
   The aim of this question is to pin down what actionability is in a robotics context and how prior knowledge can be structured to improve actionability.

2. **How can we exploit actionable environment representations to make robot task selection and planning more light weight?**
   If we introduce more context in our system to make it actionable, we hypothesize that task planning can be made more lightweight because the search space has decreased by increased context. The aim of this question is to ensure the novel world representation remains directly and practically useable by the agent.

3. **How can we discover possible tasks and behaviors online in a real-world unseen environment on the Spot robot?**
   This question seeks to provide an analysis of solving the practical problem at hand. If we can move toward this, it will be much easier to deploy mobile robots on larger scales.

1-4 Contributions

The goal of this research consists of developing a world representation that makes mobile robots more effective in performing missions in the real world by improving their SA.

- **The proposal for a new research topic of situational affordances in robotics that will promote the tighter connection between perception and robot task planning.** Situational affordances are a prime candidate to structure prior knowledge to provide mission context for mobile robot deployments in domains where the primary goals are information collection.
- **A novel approach to actionable environment representation for mobile robots, the Behavior-Oriented Situational Graph, and its accompanying affordance-based construction pipeline are presented.** By making some strong but commonly supported assumptions and exploiting prior knowledge in the form of situational affordances the BOS-Graph represents behaviors in its edges and situations in its nodes.

This allows it to provide a higher level of actionability in terms of Situational Awareness than comparable graph-based world representations such as 3D (Spatial) Scene Graphs [19, 22] and Situational Graphs [37]. A better comprehension of the scene is realized because tasks and behaviors are represented explicitly. Within this higher context, more lightweight task planning is enabled than is possible with contemporary methods [36].

Practical contributions:

- **An open-source implementation[1] of the proposed algorithms, that works out of the box with Boston Dynamics Spot robots [2].** We provide open source code of our pipeline, to promote reproducible research. The software contains the reference implementations for a planner and exploration behavior, these implementations are easily extensible with more advanced methods to facilitate further development and real-world deployments. The software has clearly defined entry points for prior knowledge, making it straightforward to extend it to new use cases in new domains.
- **Extensive verification and validation of the proposed pipeline, both simulated and experimental in a real-world Search-and-Rescue scenario on physical robot hardware.** We show that the method performs in unstructured unseen environments with challenging geometric conditions. Task specification in the form of situational affordances is invariant to initial conditions because tasks are dynamically discovered in the environment. The results from the experiment suggest that the proposed pipeline is an effective specification for autonomous missions in unseen environments.

1-5 Thesis Outline

This document is organized as follows. Chapter 2 is intended to make the thesis as self-contained as possible. We provide the necessary background knowledge about the main topics related to this thesis: planning, scene graphs, and affordances from robotics in addition to situational awareness from human factors.

The structure of the content chapters Chapter 3, Chapter 4 and Chapter 5 is illustrated in Figure 1-1.

Finally, Chapter 6 concludes the work presented in this document, and it summarises the answers to the research questions previously posed. Besides, the author included a number of suggestions for future research in this direction, pointing out the main challenges and the questions that still remain unanswered.

[1] See https://github.com/h0uter/situational-graph for code and videos.

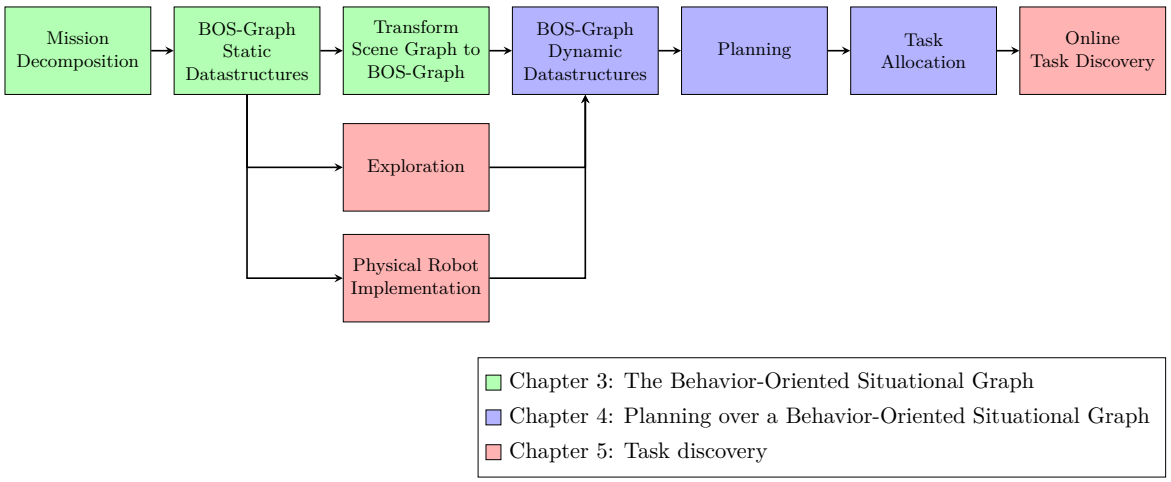


Figure 1-1: Diagram of thesis outline. Arrows correspond to the order of steps in the proposed pipeline. In Chapter 3 (green) we present the information pipeline to construct the BOS-Graph. Then in Chapter 4 (blue) we present a reference implementation for a planner over the BOS-Graph. Subsequently, in Chapter 5 (red) we provide a reference implementation of exploration to realize online task discovery and BOS-Graph construction in simulation and on the real robot.

Chapter 2

Background

In this chapter, we introduce the background and context to understand the challenges of joining contemporary robotics research fields. First a framework is necessary to discuss and evaluate the level of Situational Awareness (SA), this is introduced in Section 2-1. Section 2-3 provides an overview of the upcoming research field of scene graphs, whose aim is to obtain unified world representations capturing semantic, metric and topological features. Second, Section 2-4 provides details on the efforts to integrate the perspective of affordances from environmental psychology into robotics applications. Third, we discuss authoring the dominant paradigm in industry for robot missions in Section 2-5. Finally, a brief overview of the research field of planning in robotics is given in Section 2-6 This provides a context for discussing pipelines for autonomous task discovery in the following chapter.

2-1 Situational awareness

SA is key for any agent operating in novel and unstructured environments. It has been extensively studied in human factors because it is such a key competence in many plan domains such as aircrafts, air traffic control, large-systems operations, tactical and strategic systems, and many others.

What is situational awareness? SA is a crucial construct on which decision-making and performance in complex dynamical systems depend. Especially in dynamic environments, many decisions are required across a fairly narrow space of time and tasks are dependent on an ongoing, up-to-date analysis of the environment. Because the state of the environment is constantly changing, a major part of the operator's job becomes that of obtaining and maintaining good SA. Without SA, effective decision making becomes very difficult [38, 39].

Endsley developed a framework to assess situational awareness [38], therein SA is defined as:

Definition 2-1.1 (Situational awareness)**.** *The perception of the elements of the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.*

The rest of the framework is shown in Figure 2-1, note the distinction of level one to three of SAR.

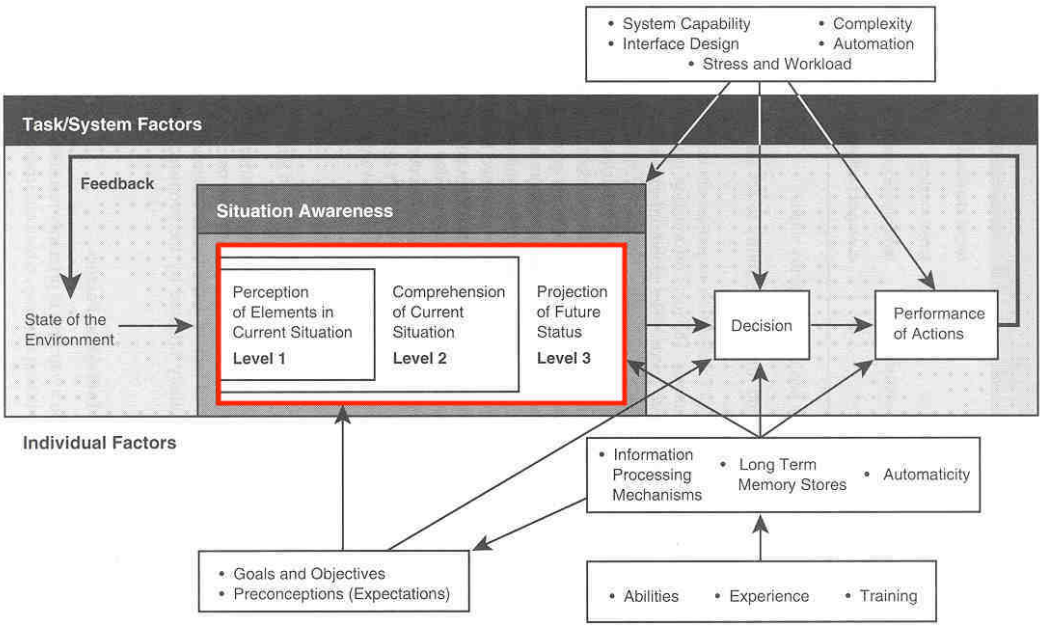


Figure 2-1: Endsleys framework of SA in dynamic decision-making from research in the human factors domain [38]. Please note the three distinct levels of situational awareness.

Situational awareness in robotics Likewise, SA is important for robots to operate in novel, dynamic and unstructured environments such as in the SAR domain. Unlike in factories, where robots are commonly deployed and where repetition is sufficient, in these more complex environments what the robot needs to do in order to achieve its objectives is dependent on the context of the environment, i.e. the situation it finds itself in.

In this paragraph we discuss how we use the framework with 3 levels of Endsley [38] to evaluate SA in a robotics context.

Definition 2-1.2 (Level 1 Situational Awareness)**.** Perception of elements in Current Situation.

Objects in the environment can be detected and categorized in an objective manner, which is what level one SA indicates. At level one, context is not taken into consideration. The majority of a typical real-world robotic system's perception of its surroundings occurs at level one or more commonly below. Object and geometric features in the environment are identified and compiled into a world representation.

Definition 2-1.3 (Level 2 Situational Awareness)**.** Comprehension of the Current Situation.

Level 2 SA extends this to comprehension of the current situation, taking into account some context. This indicates that the agent knows how perceived items relate to mission objectives. It is able to extract tasks from situations. Specifically, this context is crucial. There is no

universal context; therefore, it is essential to determine which context is relevant. For a robot to be deployed effectively, its context need only capture the processes we want it to perform. If we can connect the object instances to the processes that the robot is deployed to accomplish, we can now provide a more comprehensive understanding of the current situation. If this can be accomplished, we will be able to perceive the environment from the limited perspective of the agent's deployment. Now, the agent comprehends the environment through the lens of its own processes.

Definition 2-1.4 (Level 3 Situational Awareness)**.** Projection of Future Status.

Level 3 extends level 2 to also predict future states of the world and the mission. Only in toy problems is the robot able to operate at level 3. To advance to level 3 SA, it is necessary to be able to model how the agent can influence the environment and thus alter the situation. For this, it would be beneficial if we can reason about multiple states of world simultaneously.

2-2 2D (image) scene graphs

This section will explain 2D (Image) Scene Graph (2DSG). This is the research field of representing situations and images.

Computer vision for detecting and classifying objects Detecting and classifying objects has seen a tremendous increase in viable methods since engineers Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton submitted a program called AlexNet to the 2012 ImageNet [40] Large Scale Visual Recognition Challenge (LSVRC). The program is a breakthrough for computer vision — halving the existing error rate to just 16%. The idea that computers will be able to perceive our world starts to become possible. This breakthrough sparked the neural network revolution, where computer vision continued to be one of the most successful application domains for these techniques.

Computer vision for detecting situations in the past decade, computer vision techniques have continued to evolve. Especially in the autonomous vehicle domain, environmental perception and scene understanding are being pushed to the limits in order to create a comprehensive and integrated understanding of the environment around the vehicle. Taking into account other dynamic agents and requiring us to move past perceiving independent objects towards perceiving *situations*, as we discussed in Section 2-1.

A modern challenge is Visual Question Answering (VQA) [41], here the aim is to combine vision and language models to create a system that can answer a wide range of questions about an image scene. These methods will continue to improve, allowing us to move toward the detection and classification of situations. 2D image scene graphs are often used as the backbone for VQA methods. A 2D image scene graph is a structured representation of a scene that can clearly express the objects, attributes, and relationships between objects in the scene. As computer vision technology continues to develop, people look forward to a higher level of understanding and reasoning about visual scenes. For example, given an image, we want to not only detect and recognize objects in the image but also understand the relationship

between objects therein. Chang et al. provide a review of the state of the art with regards to the research field of 2D Image scene graphs [20].

There are many underlying techniques to obtain scene graphs. For the assumptions made in this work it is important to know the common output structure of scene graphs. The output structure is usually a set of ⟨`subject, predicate, object`⟩ triplets. Chaining these triplets together results in a directed graph. In Figure 2-2 is shown for a traffic scene to illustrate this.

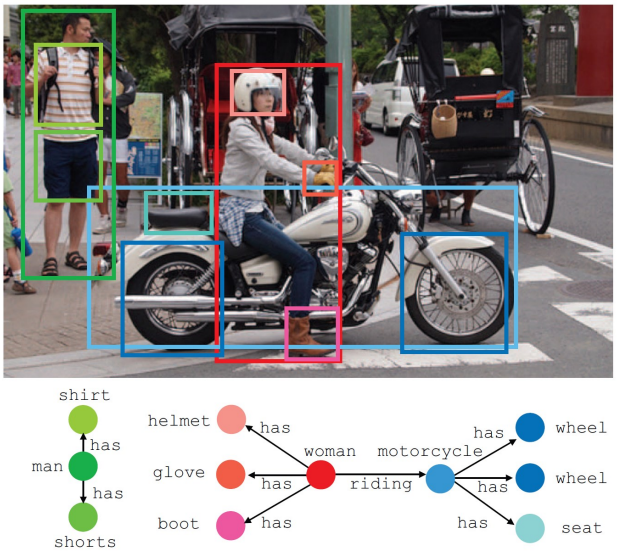


Figure 2-2: An example of a 2D scene graph of a photo containing a woman, bike and helmet which are localized in the image with bounding boxes. Color Coded above and the relationships between those entities, such as riding, the relation between woman and motorcycle or has the relation between man and shirt [42].

In this work we make some strong assumptions on the ability of the perception system of our agent to detect situations. These assumptions are motivated by the rapid advances in these 2D image scene graph techniques. This work focusses on how to aggregate and make usefull for planning these situations in an environment representation for autonomous missions with mobile robots.

2-3 3D (spatial) scene graphs

Mobile robots require rich semantic descriptions of the key elements of a scene to understand the situation around them and to localize themselves therein, in other words, to obtain SA.

Conventional methods to obtain environment representations have come a long way, but they are limited in several ways. Geometric LiDAR SLAM methods are essential for safe navigation but are unable to identify semantic elements which hamper the specification and use of high-level mission objectives. Newer semantic SLAM methods use semantically labeled

object detections as landmarks, but they do not model the relations between these elements which would improve the situational understanding of the robot [43]. These modern SLAM methods already employ (factor) graph structures in their back-end to jointly optimize sensor data to estimate the optimal combination of map data and localization state. An active field of research is on how to extend these graphs in the backend with higher-level features of the environment. Scene graphs are such a promising research field aimed at tackling some of these issues and extending these graph models that form the backbone of SLAM, so they are more useful for other robotic subsystems.

3D Scene graphs are an emerging field of research with great potential to represent situations in a joint model comprising geometric, semantic, and relational/topological dimensions, this is a lot richer way than is conventional in robotics [19]. Improvements here will likely lead to improved SA and autonomy (metric) of robots. It also allows the semantic elements to be constructed into a hierarchy, as is shown in Figure 2-3. Such hierarchical scene graphs are a promising method to enable robots to understand and navigate the environment similarly to humans, using high-level abstractions (such as rooms and doors) and the interconnections between them (doors connecting rooms).

More formally, a 3D scene graph is a hierarchical multigraph $G_{3DSG} = (V, A)$ with $k \in 1 \dots K$ levels, where $V^k \in V$ denotes the set of vertices at level k. Edges originating from a vertex $v \in V^k$ may only terminate in $V^{k-1} \cup V^k \cup V^{k+1}$ (i.e. edges connect nodes within one level of eachother.) [36]. More specifically because these hierarchical models partition space at a variety of levels of abstraction, they enable efficient reasoning over large spatial scales. Additionally, there are four use cases where their benefits are clear: Obstacle avoidance and planning, human-robot interaction, long-term autonomy, and prediction [44]. Scene graphs model the environment as a directed graph with nodes representing objects or places and edges representing relationships, depicted in Figure 2-3. Scene graphs model spatial or logical relationships. By representing the world hierarchically, it can answer "what's where?" Scene graphs are mostly academic and validated in simulations. Because these methods are new, they have some limitations.

Limitations of scene graphs in the context of situational awareness Work on spatial scene graph engines and pipelines such as Kimera [45] and more recently Hydra [46] are promising, but how they can best be exploited for robotic planning remains largely unexplored in research. In their current state they only capture the hierarchical composition of space and the spatial embedding of objects therein. They are descriptive of the situation, in the sense of level 1 in Figure 2-1, meeting level 2 only for spatial questions such as "what is where?" and dynamic scene graphs [44] answer "what is where when?" However, they do not capture situations in a way that allows answers questions like:

- "What needs to be done?"
- "What tasks are available to the agent and specifically what tasks is it capable of performing?"
- "Given my capabilities and position, how does task A compare to task B in terms of utility?"

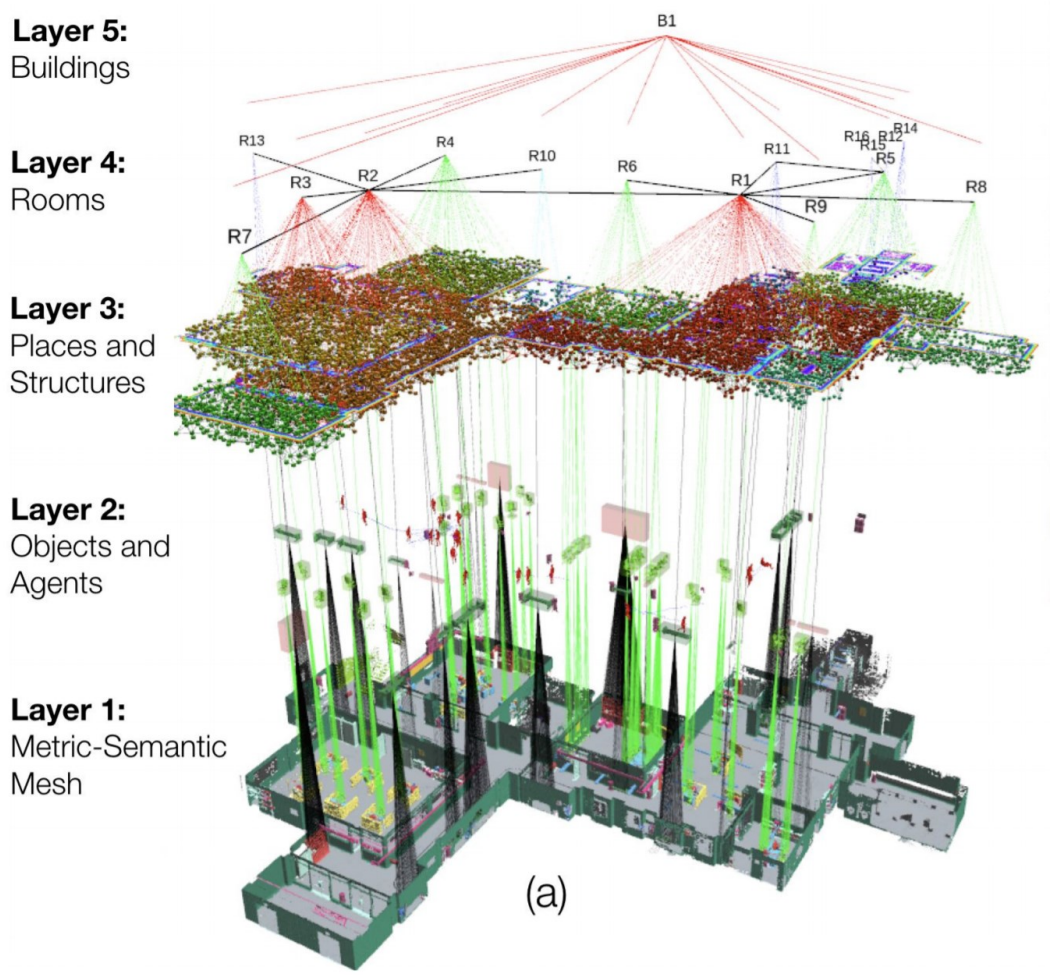


Figure 2-3: Example of a hierarchical 3D dynamic scene graph. The layers at the top can be further extended to neighborhoods, provinces, countries, continents, etc. The idea of this thesis is to add an actionability dimension built on affordances instead.

Specifically, current spatial scene graphs are limited in how they are contextualized in a robot team's mission. In general, SGs are not that actionable. What is missing is a connection with the behaviors available to the robot and the objectives of the mission. Therefore it is interesting to research how to extend the scene graphs so they can be useful for specific missions performed by robotic agents with specific capabilities.

2-4 Affordances

Using affordances in robotics tasks usually refers to the problem of perceiving a target object, identifying what action is feasible with it, and the effect of applying this action. This makes affordances a key attribute of what must be perceived by a robotic agent in order to effectively interact with known and novel objects. Historically, the concept derives from the literature in psychology and cognitive science [47]. Gibson originally defined affordance to refer to all

"action possibilities" latent in the environment, objectively measurable, and independent of the individual's ability to recognize those possibilities.

Perceived affordances In 1988, Donald Norman used the term affordances Human Machine Interaction and made it popular in the Interaction Design field. Later he clarified he was not talking about 'normal' affordances, he was refering to *perceived affordance*. The difference here is that the affordance now becomes dependent on the capabilities of the agent perceiving, making the concept subjective and therefore taking it out of the realm of hard science.

Affordances in robotics The debate on how to exactly formalize affordances for robotics is ongoing and Figure 2-4 shows some of the current proposals.

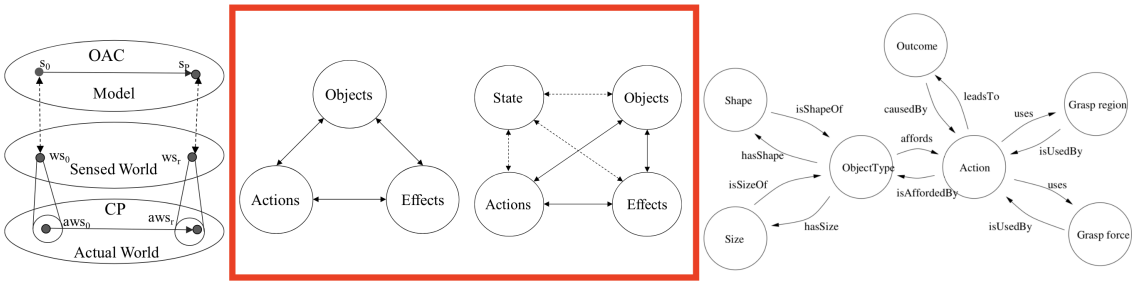


Figure 2-4: Illustration of number of competing proposals for formalizations of affordances from [1]. Please note that in this thesis we look at affordances on the behavior level, not the action level.

The value of affordances for robotics has been qualitatively proven. Affordances have mostly been researched in the context of manipulation and navigation tasks. Here, the results show that methods that use affordances as part of their task show that the inclusion of the concept improves agent performance in applications such as navigation, action prediction for collaborative tasks and manipulation [48]. A typical robotics affordances pipeline is shown in Figure 2-5. A database contains apriori known affordance relations, by obtaining objects from a scene we can filter the affordances in the database to obtain the affordances provided by the current scene.

Opportunity for affordances perspective In exploration-centric missions, one of the main goals is to obtain information about the environment. If we want the robot to perform useful behavior to achieve useful tasks, it should be able to detect these opportunities in novel environments. The theory of affordances provides an interesting perspective for how to structure prior knowledge to detect those opportunities. Moreover, we aggregate the opportunities in a world representation and then exploit them.

Identifying these opportunities to perform useful processes can be done at different abstraction levels. The literature focuses mainly on object-level affordances. These object-level affordances are generally very temporally and spatially local in nature. This thesis has identified an opportunity to approach affordances from a higher-level perspective of "What objective does this situation afford me to complete in the context of my goals and mission at large?". This work investigates situation-level affordances, which we call situational affordances.

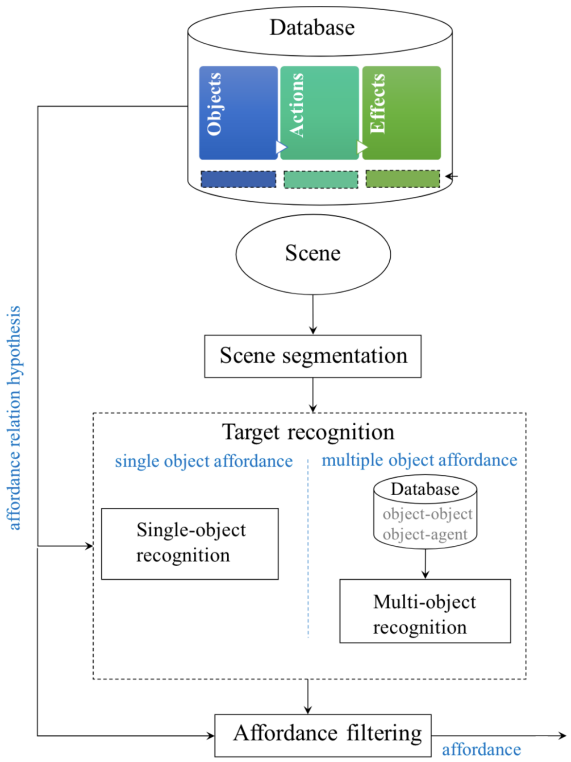


Figure 2-5: Flowchart for a typical full a priori affordance information. There is a databse containing the known relation of the target object, actions, and effects. [1]

2-5 Authoring in robotics

In the context of robotics, authoring refers to methods allowing end users to create defined robot behaviors [49]. The general process starts with a design period where an initial behavior is created, then the robot can be deployed in the real world and its behavior tested and refined in additional programming steps if needed. When the desired requirements are achieved, the authoring process is finished and the robot is ready to be deployed to interact autonomously. Industry for mobile manipulators generally takes an approach where they provide tools for behavior and tas authoring to end users. Additionally, there are tools for authoring complete missions by allowing one to orchestrate such tasks in specific sequences and rerun these sequences periodically.

2-5-1 Behavior and task authoring

Capability, usability, and robustness are the three key characteristics identified for a system for authoring robot task behaviors by [50]. First, a system should be able to perform behaviors to achieve a wide variety of tasks. Second, end-users should be able to understand the system's capabilities and efficiently create new behaviors that meet their needs. Finally, the behaviors should be robust to variation, and repeated executions should produce the expected result.

Behavior trees are a popular method to design and represent behaviors of agents. They were originally mainly used to design the behavior of adversaries in video games. In the past decade, they have seen an increased popularity in the robotics community. PyTrees [51] and BehaviorTreeCPP [52] are the most popular open-source implementations of the approach. BTs can be classified as a deterministic algorithm with real-time reactivity. BTs have many nice properties such as modularity, stability guarantees, and interpretability.

Behavior trees can also be generated from plans composed by a planner. The most common approach to synthesize a BT from a planner consists of two steps: first, the planner computes a plan to solve a given task, then a planner-specific algorithm converts the plan into a BT that is finally used to control the agent (a robot in many cases) [53].

2-5-2 Mission authoring

Mission authoring is when you spatially specify what a robot should do in an environment. The current trend in industry is to provide tools and an API to facilitate authoring for end-users.

Boston Dynamics AutoWalk missions Boston dynamics provides the Autowalk system to design autonomous missions with the Spot robot. The primary use case for this system is missions to perform periodic inspection of assets in industrial facilities. The most common way to use AutoWalk is to teleoperate Spot through the facility and manually specify points of interest along the route. The teleoperator can then manually specify the actions that the robot should perform at each point of interest. Actions are primarily taking photos of assets, but can also include simple manipulation tasks, such as opening or closing valves. An example of a resulting authored mission is shown in Figure 2-6.

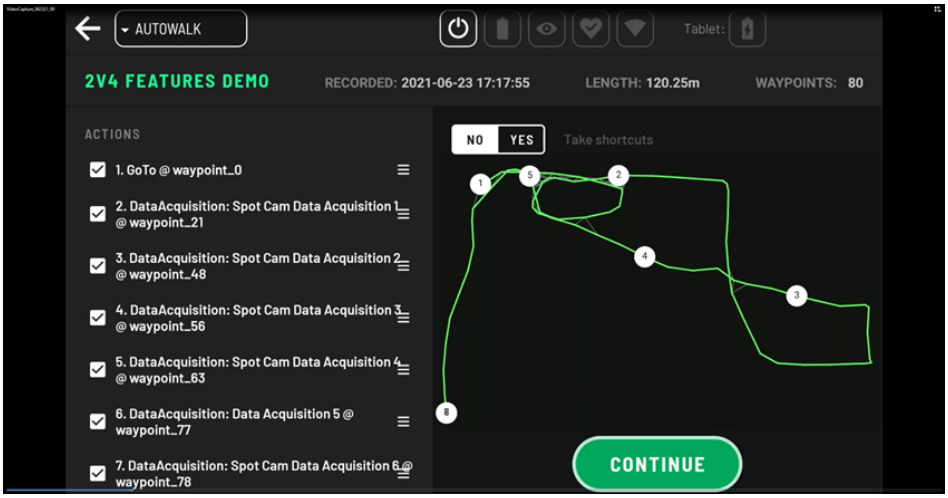


Figure 2-6: Interface for managing AutoWalk [54] missions. Bright green lines indicate the path that the robot will take when executing the mission. Dark green lines are paths that the robot will not take unless the robot's original path is blocked. Numbered dots correspond with Action(s) the robot will perform during the mission. Dark green dots are Actions the robot will not perform.[54]

2-6 Planning in robotics

This section will provide a brief overview of the relevant methods for the planning of robot tasks. Robot task planning is the research field of finding a plan, defined as a sequence of robot actions, which achieves a particular task. There are several approaches to robot task planning techniques. Some techniques have focused on constructing more effective representations to plan upon [55]. Symbolic or Classical planning is less slow than probabilistic planning, but cannot reason about multiple action outcomes, unreliable observations, or multiple possible worlds. Thus, it produces a linear plan that succeeds only if all the actions in the plan succeed. Probabilistic or Decision-theoretic planning, by contrast, produces a policy that maximizes the probability of success from any belief state the robot might reach during plan execution, taking into account the probabilities of every possible action outcome in those states. Decision-theoretic planning thus handles multiple action outcomes (Markov decision processes, MDPs) and unreliable or partial observations (POMDPs), but at an unfeasible computational cost given the size of typical robot domains.

2-6-1 Symbolic task planning

Task Planning: Autonomously reasoning about the state of the world using an internal *model* and coming up with a sequence of *actions*, or a *plan*, to achieve a *goal*.

The main reason why task planning is not as popular as other areas of robotics is that it is overkill for most commercially available applications. Commercial robots mainly automate simple, repetitive tasks that do not require higher-level planning.

Planning Domain Definition Language (PDDL) PDDL is intended to express the "physics" of a domain, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are.

Symbolic task planning is search Two types of search are most common. Forward search, where we start from the initial state and expand a graph until a goal state is reached. And alternatively backward search, where we start with the goal state and search backwards to reach the intitial state. Multiple search algorithms are possible: depth-first search, breadth-first search etc. It is also important to prune the search graph to avoid cycles and expanding unnecessary actions.

A common strategy to speed up search is the use of *heuristics*, proxies to evaluate our expansion candidates by. The search heuristics are of great interest in the planning literature.

Limitations Symbolic planners operate on a purely symbolic representation of the world, this breaks down if sub-symbolic details can cause plans to fail. The most common sub-symbolic details are geometric features of the environment related to navigation and manipulation. These geometric problems have their own class of solution methods; this research area is called motion planning. It quickly becomes apparent why it paramount to jointly consider task and motion planning to solve real-world problems. This is its own research field, appropriately named Task and Motion Planning (TaMP).

Another downside is that it is difficult to include numerical optimization in symbolic planning. Another challenge is the grounding problem, because the representation is completely symbolic it is not straightforward to ground plans and objects geometrically in the world. The final downside is that plans are made completely offline. Because of that we cannot exploit new information such as making use of shortcuts or new opportunities.

2-6-2 Probabalistic planners

Probabalistic planners do not generate plans, they generate *policies*, general plans for every possible state.

Markov decision processess MDPs are based on the assumption of the Markov property, which states that *The future is independent of the past, given the present.* Given states and costs and transition probabilities, this allows the use of dynamic programming to formulate policies.

Limitations While the Markov decision process is a very general problem formulation, it has significant downsides. Calculating these policies for real-world problems quickly becomes computationally intractable. Probabalistic methods scale quite poorly due to the curse of dimensionality.

2-6-3 Joint task and motion planning

There are also approaches that integrate task and motion planning [56]. So full TaMP solvers combine discrete symbolic planners with continuous motion planners. Sometimes discrete symbolic plans turn out to be infeasible when we try to find the motion plans. PDDL stream [57] combines continuous space planners based on sampling such as RRT with symbolic planners such as PDDL. PDDL Stream is the successor of [58]

2-6-4 Taxonomy of methods

Briefly, a taxonomy of the field is shown in Figure 2-7. It should be noted that there is a clear divide between planning approaches and authoring approaches. There are little planning methods that embrace authoring and vice versa. This gap in methods provides a clear opportunity, to develop a method which brings the best of both worlds.

2-7 Concluding remarks

In conclusion, Situational Awareness (SA) is a key competency of any agent to operate effectively in novel dynamic environments, and for humans there exist frameworks to quantify SA at a high level. However, it is debatable how this should be applied in the mobile robotics domain.

Scene graphs are a promising and comprehensive novel environment representation; however, the level of SA they provide is still limited according to the SA framework of the previous section. This leaves room to investigate how to raise the level of SA. This would allow us to investigate what information is required to allow the creation of more actionable environment representations, and what their accompanying pipelines should look like. We conclude that scene graphs are a key ingredient to provide SA to future SAR rescue workers, and that they provide opportunities to be extended further with an information pipeline to make them more actionable and provide improved SA.

One promising research area for the creation of these information pipelines is affordances in robotics. Research into affordances aims to capture the possibilities for action provided by an environment and their effects. However, we conclude that how affordances should be formalized is still a very active research discussion.

Finally, we provide a brief overview of authoring and planning paradigms to get robots to actually perform useful behaviors. Authoring is the commercial standard, but it is mainly limited by the requirement of human effort if the environment has changed too much. This fact makes this paradigm unsuitable for unseen environments. Full planning, on the other hand, should theoretically allow generalizability to novel environments. However, in practice, they are mostly limited to toy problems in lab settings because of several practical problems. Mainly, they require full observability and their solvers have trouble with scaling computationally to a typical SAR domain. We conclude that while investigating how to raise the level of SA of a mobile robot environment representation it will be beneficial to jointly consider how it will be used by a planning system on the robot.

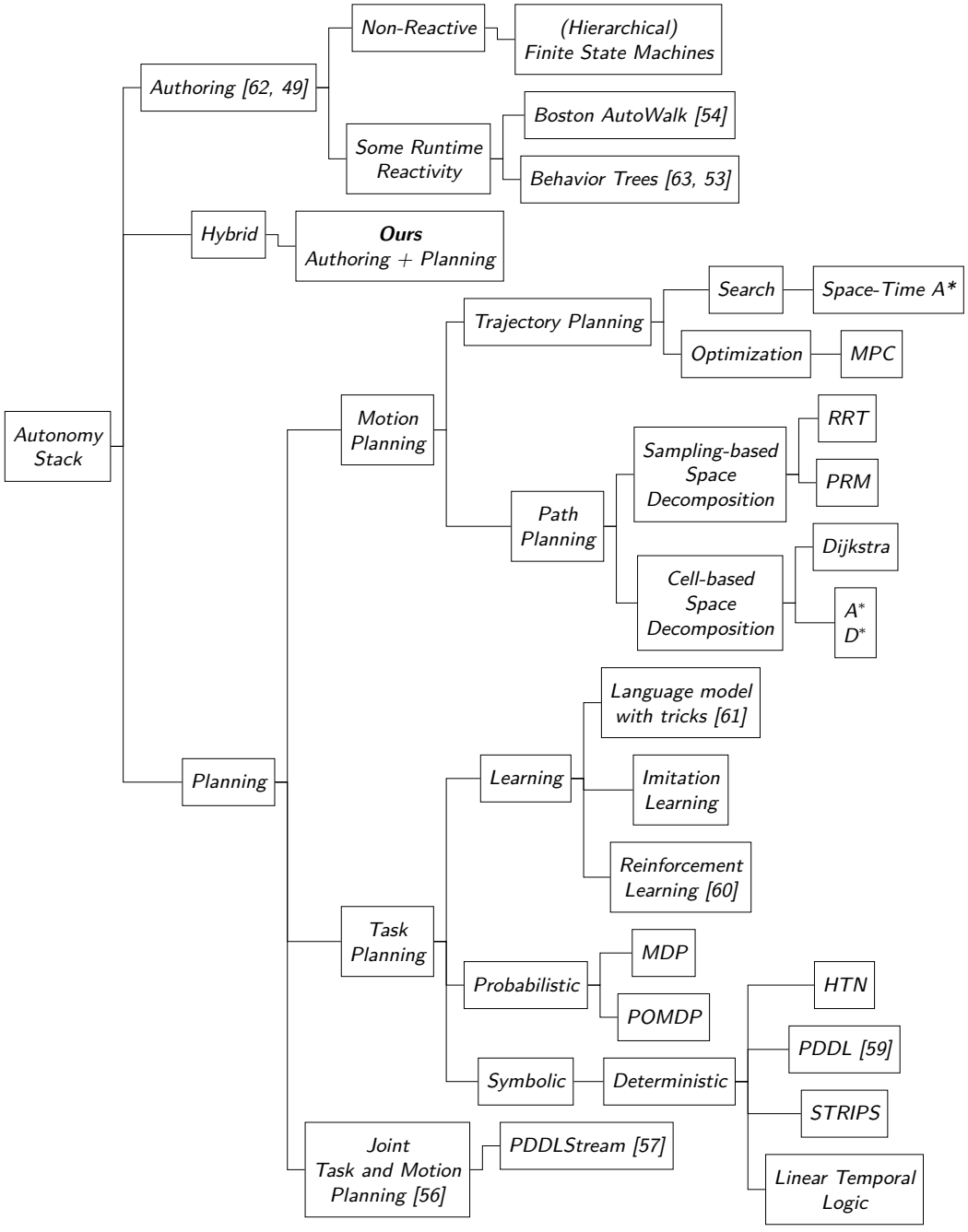


Figure 2-7: Taxonomy of autonomy approaches in autonomous missions for mobile manipulators.

Chapter 3

The Behavior-Oriented Situational Graph

In this chapter, we describe how the BOS-Graph is built. A useful analogy is to compare it with a database, the dynamic data structures are the database instances, and the static data structures are the database schema. First, we state and scope the problem we address in Section 3-1. Then Section 3-2 provides the graph backbone for BOS-Graph, and provides the reader with an overview of the high-level interrelation of the data structures that follow. Section 3-3 proposes how missions should be decomposed into goals and objectives. Then Section 3-4 will show how affordance relations provide the essential schema for our prior knowledge. Section 3-5 details our implementation-agnostic inclusion of behaviors, and Section 3-6 details our expectations for situation perception. Finally, we qualitatively evaluate how this provides a higher level of SA in Section 3-8.

Notation Note that tuples $\langle \dots \rangle$ are ordered sequences of elements, using angle brackets. Sets $\{\dots\}$ are unordered collections of elements using curly brackets. Additionally, note that when indexing properties of nested data structures or data structures that reference other data structures, we employ a dot indexing notation similar to most object-oriented programming languages as shown in Equation 3-1.

$$\begin{aligned} a.b &:= \{b : b \in a\} \\ a.b.c &:= \{c : c \in \{b : b \in a\}\} \end{aligned} \tag{3-1}$$

3-1 Problem statement

Let us take a conservative perspective on mobile robot deployments. Real-world deployments are primarily limited to "go from A to B and do C" type domains. This fits well with the SAR domain, where mobile robot deployments should contribute to rapid acquisition of Situational

Awareness (SA). This is in contrast to domains where the precise order of a large number of actions is key such as cooking or assembly.

Our aim is to investigate a novel environment representation that can provide a higher level of SA. Both for the rescue workers as well as for the agent themselves to improve their goal autonomy. To raise the level of SA we need to represent a comprehension of the elements in the environment following Definition 2-1.3. In the context of robotics we interpret that as described in Assumption 1.

Assumption 1. To comprehend a situation from a goal autonomy perspective is to be aware of the environment's available relevant behaviors for tasks, or, in essence, the opportunities towards progress on objectives available to the agent.

Mission decomposition framework We need to define how to decompose a mission for our specific purposes. The proposed hierarchical mission decomposition framework is illustrated in Figure 3-1. As shown in Figure 3-1 from top to bottom, a mission is decomposed into a set of goals. Goals are abstract and long-term. Goals are decomposed into objectives, general short-term outcomes that contribute to the goal. A parametrized instance of such an objective in a specific situation is a task. A task can be achieved by performing a plan, and every plan is composed of a sequence of behaviors.

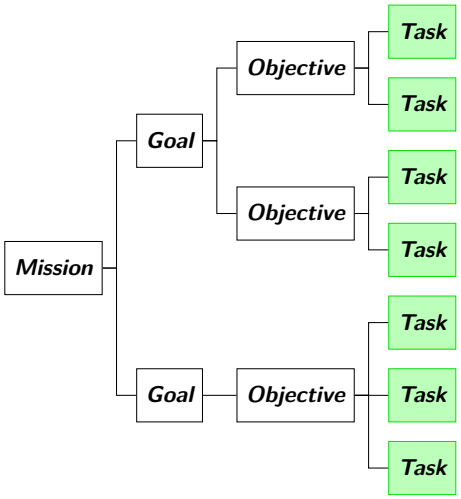


Figure 3-1: The mission decomposition framework for the proposed method. Inspired by [39] and [56]. Prior to the mission, it is decomposed into goals, which are decomposed into objectives. This decomposition is done manually and constitutes part of the authoring investment. At runtime, the robots can instantiate those objectives to tasks (green) to create the BOS-Graph. The proposed method can generate plans to accomplish those tasks, thus contributing to the mission.

To limit the scope of this thesis to environment representations, we make a number of assumptions on the deployment domain.

Assumption 2. Goals are chosen so that their progress always increases monotonically. Achieving a task corresponding to an objective is always beneficial to the goal.

Because the most important real-world use case of robots involves data collection, we assume that it is always beneficial to collect more data. This assumption allows us to define objectives as abstract schemas for concrete local outcomes which are always beneficial to a global goal.

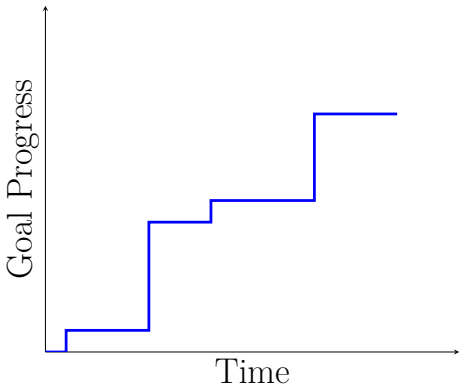


Figure 3-2: Goal specification has to meet the requirement that goal progress is an monotonic increasing function.

Assumption 3. For every objective o there exists a behavior b whose successful outcome achieves that objective.

$$\forall o \; \exists b \quad \text{s.t.} \quad b.\text{success} = o \tag{3-2}$$

3-1-1 Example: simple mission decomposition

This subsection shows an example decomposition for a toy problem, the example is illustrated in Figure 3-3. The agent has to obtain an assessment of each victim encountered in the environment. The black left side in Figure 3-3 shows the decomposition of the mission into goals, which are then decomposed into objectives. This decomposition is done manually through prior knowledge. An instance, aka a task is shown in green in Figure 3-3. A generic graph map that corresponds to this state of the mission is shown in Figure 3-4. For this toy problem we assume full observability, the agent already knows the victim is at wp3. This is motivated by the fact that another agent without the capability for assessment has provided us with this graph. In the subsequent chapters we will describe how we can obtain such a graph online, but in this chapter we focus on known graphs.

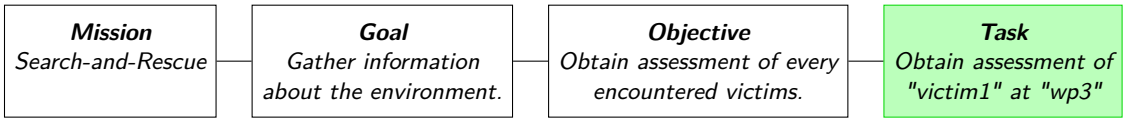


Figure 3-3: Example decomposition of a mission for a simplified mission. In green are the instantiations and in black, the mission schema is shown.

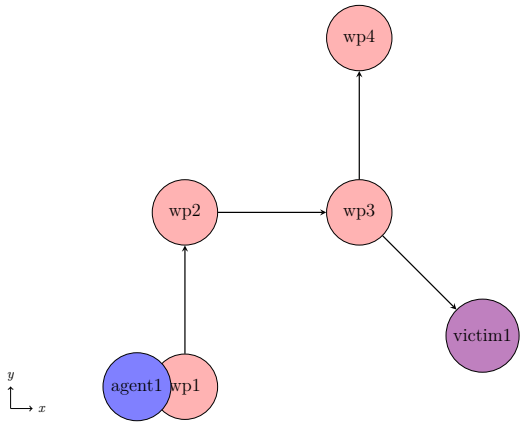


Figure 3-4: A generic graph representation corresponding to the fully observable state of the mission as in Figure 3-3. The agent is at wp1, an unknown victim is at wp3.

The following concrete examples of the terminology are linked to their exact definitions in the glossary. These examples correspond to the illustrated hierarchical decompositon in Figure 3-3.

- Example of a **mission** is to perform Search-and-Rescue within a volume of space and time.
- Example of a **goal** is to gather information about the environment.
- Example of an **objective** is the generic statement to obtain an assessment of the well-being of every encountered victim.
- Example of a **task** is the specific statement to assess the well-being of a specific victim at a specific place. It contributes to the aforementioned goal.
- Example of a **plan** is a dynamically generated sequence of parametrized behaviors to approach a specific victim and assess their well being.

3-2 The BOS-Graph

This section introduces the graph backbone to connect all subsequent data structures. The aim is to provide the reader with a high-level overview; to this end, we present a database diagram of how the data structures are connected high level.

The extended directed multigraph The graph used to represent the BOS-Graph is a directed multigraph, also called a quiver in mathematics. A standard directed multigraph can be defined as follows. The BOS-Graph is a set of node instances $\mathcal{V}$ and edge instances $\mathcal{E}$.

$$\mathcal{G} := \{\mathcal{V}, \mathcal{E}\} \tag{3-3}$$

Every node v in $\mathcal{V}$ has an identifier `id`, a situation s, and a global position x. The definition of a situation s will be detailed in the subsequent Section 3-6.

$$\begin{aligned} v &:= \langle \texttt{id}, s, x : \quad \texttt{id} \in \mathbb{Z}^+, \quad x \in \mathbb{R}^2 \rangle \\ \mathcal{V} &= \{v_i\}^{i=1,\dots,N_\mathcal{V}} \end{aligned} \tag{3-4}$$

Every edge e in $\mathcal{E}$ essentially represents a parametrized behavior the robot can perform from the source to, or targeted at, the target node v_{target}. Each e has an identifier `id`, a source node v_{source}, a target node v_{target}, and a behavior b.

$$\begin{aligned} e &:= \langle \texttt{id}, v_{\text{source}}, v_{\text{target}}, b, f : \quad \texttt{id} \in \mathbb{Z}^+, \quad b \in \mathcal{B}, \quad w \in \mathbb{Q} \rangle \\ \mathcal{E} &= \{e_j\}^{j=1,\dots,N_\mathcal{E}} \end{aligned} \tag{3-5}$$

The edge also has a function f that maps the positions of the source and target nodes and the cost of the behavior b to the weight of the edge w. The weight of the edge is determined in terms of the resources required to perform it, such as time and battery.

$$f : (b.c, v_{\text{source}}.x, v_{\text{target}}.x) \rightarrow w \tag{3-6}$$

Overview of the BOS-Graph data structures First, we introduce the dynamic data structures at the top of the diagram, the graph. In the following sections, we will extend it with the static data structures at the bottom of the diagram in Figure 4-2. The key static data structure is the affordance structure, which contains the relations between situations in $\mathcal{S}$, behaviors in $\mathcal{B}$ and objectives in $\mathcal{O}$.

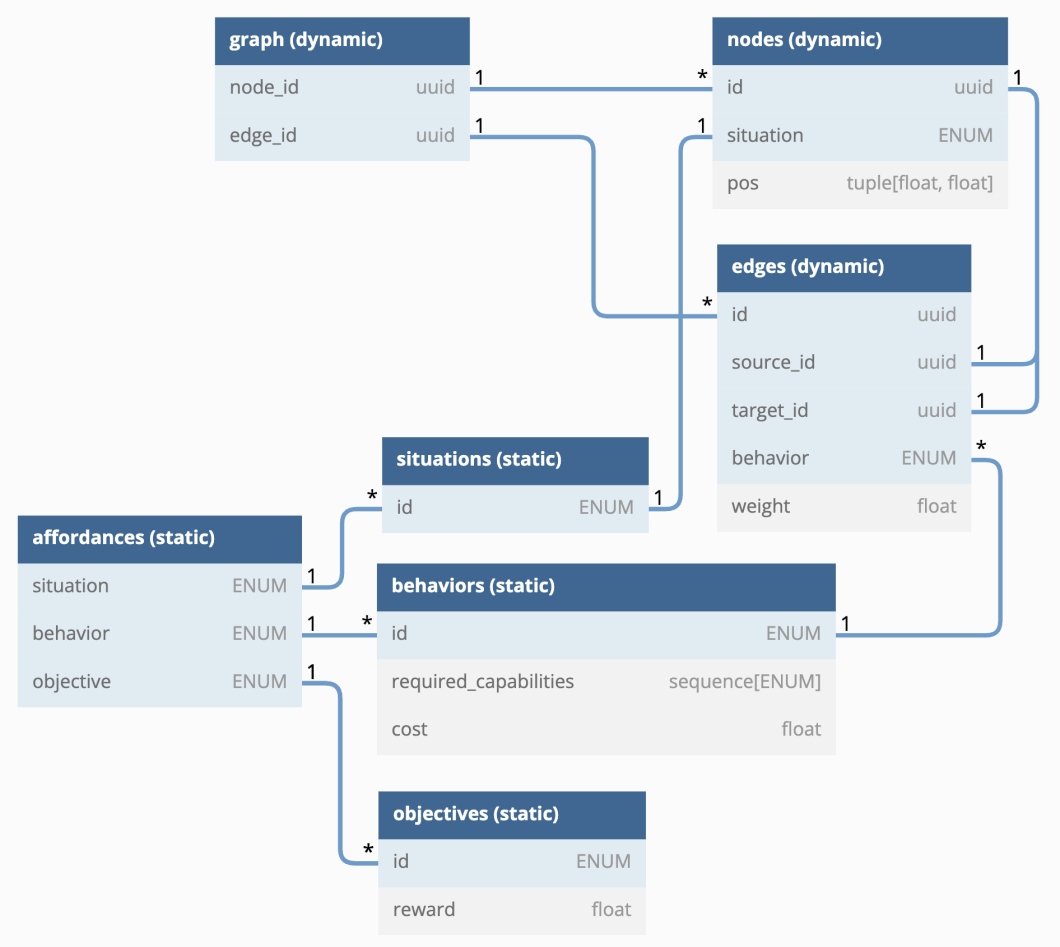


Figure 3-5: Database diagram of the data structures involved in the BOS-Graph without planning. This diagram serves as an overview of the data structures we introduce in the rest of the chapter. The main data structure is the affordance, which connects situations, behaviors and objectives. Each node corresponds to a situation instance. Each edge corresponds to a specific parameterized behavior. ENUMs are apriori known identifiers, uuids are dynamically generated identifiers. We discuss rewards and show the full diagram with planning in Chapter 4.

3-3 Authoring objectives

This section details the data structure we start with when defining our mission, the objective. An objective is a general outcome that is always beneficial to a goal (following Assumption 2). This section will detail how the objectives should be authored.

Objectives formalized An objective o in the set of all objectives $\mathcal{O}$ is defined as a tuple. Each o has an identifier `id` and a reward r.

$$\begin{aligned} o &:= \langle \texttt{id}, r \rangle \\ \mathcal{O} &= \{o_i\}^{i=1,\dots,N_{\mathcal{O}}} \end{aligned} \tag{3-7}$$

3-4 Authoring situational affordances

Having defined our objectives in the previous section, we now require the data structure to connect the objective to behaviors (what should be done) and situations (when should it be done). This section will detail and formalize situational affordances. Situational affordances are important because we use them to provide a new perspective on how we could define robot missions. Practically speaking, affordances are essentially the schema that defines which edges can connect which nodes.

Traditional affordances Affordances traditionally capture the relation between objects, actions, and effects as shown in Figure 3-6.

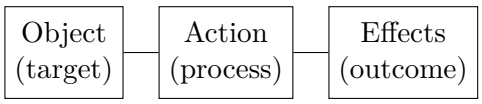


Figure 3-6: Traditional affordances capture the relations between objects, actions and effects.

Towards perceived affordance in mission context Mission level affordances are the relations between situations, the processes to which they can be subject, and the effects those processes will have on the mission. Traditional affordances were extended to *perceived affordances* by Donald Norman [**?**]. This made affordances subjective; in this work we also propose to move towards perceived affordances as shown in Figure 3-7. But not perceived affordances from the perspective of an individual agent, but perceived affordances from the perspective of the mission. The effects are replaced by objectives, which are effects in the context of a mission.

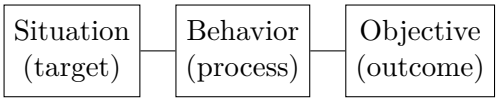


Figure 3-7: The schema of affordance relations between abstract situations, behavior, and objectives. This provides a template for creating instantiations during the mission. Corresponds to the right side of Table 3-1

Example A victim situation affords agents to perform assessment behavior that achieves an information gathering task. Another example: a closed-door situation affords the agent the open-door behavior; the outcome of that behavior is that we can obtain new frontiers and continue the exploration.

Task instantiation using affordances during mission While the mission is carried out, the affordance schema is used to instantiate tasks. This instantiation is illustrated in Figure 3-8. Instances of situations are represented by nodes and instances of behaviors are represented by edges in the BOS-Graph.

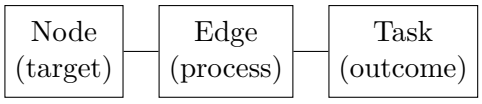


Figure 3-8: An instantiation following the schema in Figure 3-7. An instance links a situation node with a task and the edge to achieve that task. Corresponds to the left side of Table 3-1.

Situational affordances formalized An affordance h is the relation between a situation s, a behavior b, and a mission objective o. The affordance is formalized as a trituple. They are the key unified entry point for authored prior knowledge in the proposed framework. The set of all authored affordances $\mathcal{H}$ contains the blueprint of the work that the robot can perform in the world during its deployment.

Assumption 4. Before deployment, we can already say a lot about the types of situation relevant to the different behaviors the robot is able to execute in the world.

To perform meaningful work, in the form of authored behaviors, the robot needs to recognize situations and know which objective o_i the specific situation s_i relates to and the specific behavior b_i it can apply in that situation. This is precisely the information captured in the affordance.

$$\begin{aligned} h &:= \langle s_i, b_i, o_i : \quad s_i \in \mathcal{S}, \quad b_i \in \mathcal{B}, \quad o_i \in \mathcal{O} \rangle \\ \mathcal{H} &= \{h_i\}^{i=1,\dots,N_{\mathcal{H}}} \end{aligned} \tag{3-8}$$

Affordances without objective Some affordances contain the zero objective $\varnothing$. This allows us to specify which edges are allowed between which nodes, without instantiating tasks. One example is the goto behavior, which in itself does not necessarily correspond to a task; however, we do need to know that we can connect waypoint nodes with this edge. This information is encoded in an affordance with a zero objective element, as shown in Equation 3-9. This allows us to make the distinction between two sets of affordances, affordances which cause tasks to be instantiated $\mathcal{H}_{\text{objective}}$ in addition to their edge. And, on the other hand, affordances which instantiate edges without tasks $\mathcal{H}_{\text{objective}}$.

$$\begin{aligned} \mathcal{H}_{\text{supportive}} &:= \langle s_i, b_i, \varnothing : \quad s_i \in \mathcal{S}, \quad b_i \in \mathcal{B} \rangle \\ \mathcal{H}_{\text{objective}} &= \mathcal{H} \setminus \mathcal{H}_{\text{supportive}} \end{aligned} \tag{3-9}$$

Summary Table 3-1 clarifies the distinction between processes and outcomes on one hand, and instances and schemas on the other hand. Schemas represent the prior knowledge of what work we expect the robot to find in the world and how it should approach that. Instances are concrete situations in the world that have been encountered.

Table 3-1: Overview of schema vs instance view on affordances. Instances are parametrised and correspond to concrete situations. The schema connects unparametrised templates.

	Parametrised Instance	Unparametrised Schema
Target	Node	Situation
Process	Edge	Behavior
Outcome	Task	Objective

3-5 Authoring behaviors

In order to achieve objectives, the agent needs to take action, behaviors are high-level and self-contained, and we generally assume that they can accomplish an objective (following Assumption 3). We describe two important aspects of how we use behaviors in this method. First, we describe how a behavior is defined in this work. Second, some examples of the possibilities for implementing behaviors are given.

3-5-1 How do we define a behavior

In classical task planning, an action is defined as a process that transforms the state of the world [56]. *Preconditions* specify logical predicates which must be true in a given state to allow the action to be taken from that state. *Postconditions* or *effects* specify the changes in state that occur as a result of executing the action. We propose a much looser definition because we have looser requirements on the planner.

Assumption 5. Behaviors are well-tested closed-loop implementations that generalize to a wide range of initial conditions and are robust to local disturbances.

This assumption is based on the behavior provided by Boston Dynamics for the open-door objective [64]. This behavior has been extensively tested to work for a wide range of doors, and we assume that behaviors can be developed to a similar level of maturity for other objectives.

Behavior preconditions The BOS-Graph captures spatial preconditions for behaviors. The source node of an edge corresponds to the precondition where the agent needs to be to perform the behavior. Preconditions are thus expressed implicitly as waypoint nodes in the BOS-Graph.

Behavior postconditions We take a more high-level view on behaviors in this work. Instead of rigidly defining how they change the state of the world, we define them as a process that can change the BOS-Graph. This means it can add and/or remove nodes and edges to the BOS-Graph. The types of nodes added determine which edges are added according to the affordances. As noted in Equation 3-10 a behavior is a function that maps a BOS-Graph to a post BOS-Graph.

$$f_{\text{behavior}} : \mathcal{G} \rightarrow \mathcal{G}_{\text{post}} \tag{3-10}$$

Separation of concern Furthermore, we take an implementation agnostic perspective to behaviors, we separate the concern of how they are implemented. The implementation of behavior knows about the BOS-Graph, but the BOS-Graph and the planner do not know the implementation of the behavior. From the perspective of the planner, behaviors are just edges on a graph. And from the perspective of the plan, they are just ids that are passed sequentially to an executor. The executor looks up the actual implementation of the behavior and runs it. The implementation checks if its preconditions are fulfilled and, after running, checks if it completed successfully. How this check is performed is a detail that whose responsibility we delegate to the implementation. In essence our method performs the check for this precondition at perception time, so that we do not need to consider it at planning time.

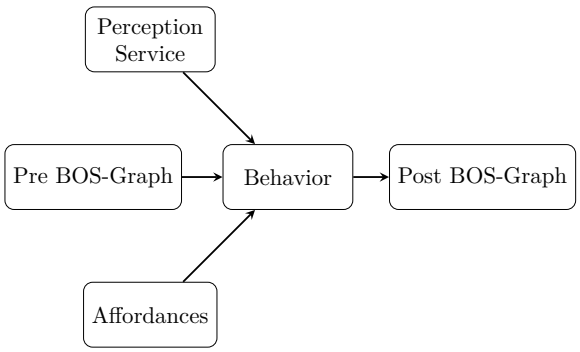


Figure 3-9: Diagram of how behaviors are defined. Behaviors mutate a pre BOS-Graph into a post BOS-Graph based on the schema defined in the affordances.

Behaviors as edges Similar to how each node references a situation, each edge references a behavior. This reference indicates the type of behavior that the edge represents. The source and target nodes of the edge provide the parameters for the unparameterized behavior template, turning it into a parameterized behavior. Only this source and target node are required to parameterize a behavior. It corresponds to the preconditions of where the agent has to be located and the target of its behavior. The work in [65] uses edges as behaviors; however, only deep learning navigation behaviors were considered in that work. A behavior b is defined as a tuple that contains a unique identifier `id`, the set of capabilities required to perform the behavior $\mathcal{K}_b$, and finally the cost factor of the behavior c. The set of all behaviors is then $\mathcal{B}$.

$$\begin{aligned} b &:= \langle \texttt{id}, \mathcal{K}_b, c \rangle \\ \mathcal{B} &= \{b_i\}^{i=1,\dots,N_{\mathcal{B}}} \end{aligned} \tag{3-11}$$

Behavior implementation and authoring Some examples of how behaviors could be implemented are discussed next. Behaviors can be made as complex and reactive as desired.

Fully-authored means that the behavior is a procedural description. This can be a sequential procedure similar to that of a computer program. There also exist more advanced methods for authoring, such as behavior trees [53]. Here, the result behavior possesses some local reactivity which allows it to respond to some degree of disturbances.

There is no reason why the behavior could not be a lower-level planner itself. The scalability issues of classical and probabilistic planners become a lot less of an issue if we only attempt to use them for very local problems. For example, the behavior to open a door could be solved with a task and motion planner, such as PDDLStream [57], that operates only on the local perception scene instead of some global environment representation.

Learned methods are on the rise, here there is typically some end-to-end black box system which can go directly from sensor data to manipulator control. However, pure learning methods are often very limited in the time or step horizon over which they operate, motivating research into hybrid methods [66].

3-6 Authoring situations

Now that we have defined our objectives and what behaviors can achieve those objectives we need to define the situations that provide the opportunity to apply these behaviors. In this section, we describe the prior knowledge required to describe a situation for the pipeline. We make a slight detour to discuss the background that supports our assumptions on the perception system that can detect these situations.

The simplest situation Before considering how we wish to define complex situations, let us first investigate the simplest situation. The simplest situation is a single object in a specific location as shown in Figure 3-11a. Or, more practically, the simplest situation is that some object is in the wrong place. A civilian should not be in a disaster environment; therefore, the detection of a civilian anywhere is an interesting situation. Detection of fires in an industrial setting is also a highly relevant situation that generally require immediate intervention.

Defining objects Now, let us focus on detecting objects and how we define and represent objects in BOS-Graph. We divide objects into two groups purely for the reader's conceptual clarity. World objects $\mathcal{L}_{world}$ are semantically significant local phenomena that we can detect in the environment. Or, more colloquially, "things you can point your finger at that have a human readable name" [67]. In our plan domain, the relevant world objects are primarily the things we want the agent to collect information about. World objects are commonly detected using modern computer vision pipelines.

The second type of object are virtual objects $\mathcal{L}_{virtual}$. These are constructs relevant for the agent and the mission that do not exist physically. The agent uses them to make sense of its environment for practical behavior, such as navigation and exploration. We need these virtual objects because they provide the target parametrization for behavior edges.

$$\begin{aligned} l &:= \langle \texttt{id} \rangle \\ \mathcal{L} &= \{l_i\}^{i=1,\dots,N_{\mathcal{L}}} \\ \mathcal{L}_{world} &\subset \mathcal{L} \\ \mathcal{L}_{virtual} &\subset \mathcal{L} \\ \mathcal{L}_{world} \cap \mathcal{L}_{virtual} &= \varnothing \\ \mathcal{L}_{world} \cup \mathcal{L}_{virtual} &= \mathcal{L} \end{aligned} \tag{3-12}$$

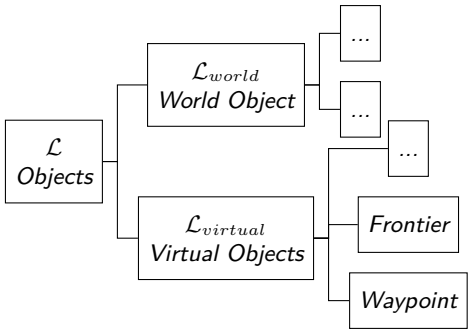


Figure 3-10: The taxonomy of objects $\mathcal{L}$ for each mission. Waypoints indicate where the agent has been, while frontiers indicate where they have not yet traveled. This distinction between virtual and real-world objects is made solely for the reader's conceptual clarity.

Basic situations Multiple objects co-existing in space and/or sharing conceptual relationships can characterize more complex situations. A person who is close to a fire is a different situation from a person who is not close to a fire. Obviously, we must specify in advance which situations are important and relevant to our mission objectives and ensure that our perception system is capable of detecting and classifying them. Due to the fact that perception is not the focus of this thesis, we simply assume that we can detect situations.

Assumption 6. We assume that the agent has a perception system capable of robustly detecting and classifying situations.

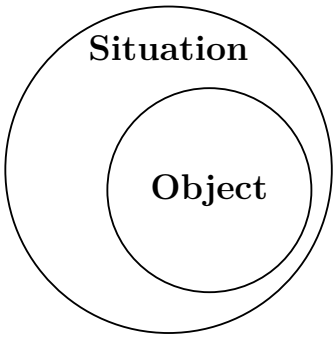
This assumption is based on the rapid advancement in the research field of 2D image scene graphs. As we discussed in Section 2-2, 2D image scene graphs use data-driven methods to extract the relationships between objects within a scene. The assumption's principal risk relates to the ability to assign discrete situation labels to the scene graph.

Situations formalized Each node v contains a reference to a situation. A situation s in the set of all apriori known situations $\mathcal{S}$ is defined as a tuple with a single identifier `id`.

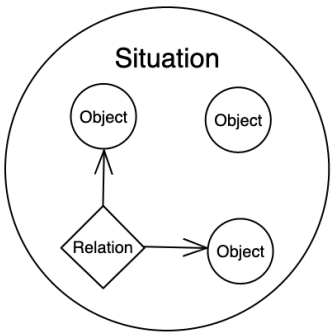
$$\begin{aligned} s &:= \langle \texttt{id} \rangle \\ \mathcal{S} &= \{s_i\}^{i=1,\dots,N_{\mathcal{S}}} \end{aligned} \tag{3-13}$$

These are what will be authored by the end-user to define which situations are important for the specific mission for which the robot will be deployed.

This authoring could consist of labeling particular situations, as illustrated in Figure 3-11. A situation is then defined as a set of objects in a volume of space and a set of relations shared by specific objects.



(a) The simplest situation. It is an object being at a certain position in space and time.



(b) More complex situation. A reference for how we could label situations based on the output of a scene graph pipeline. The output graph of Figure 2-2 could be pattern matched against a situation defined in this way to perform clasification.

Figure 3-11: Situations as defined in this work.

More complex situations Obviously, situations can be considerably more complicated than the example we provided above. Figure 3-12 displays some examples of situational scale. Expanding the kinds of situation that can be captured by the BOS-Graph would be an intriguing topic for future research.

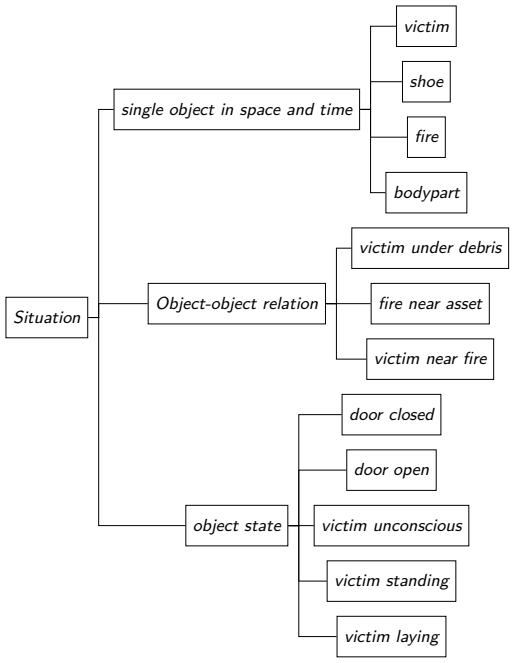


Figure 3-12: Examples of simpler and more complex situations relevant within the SAR domain.

This slight detour provided the background for the assumption that we can recognize and categorize situations. This brings us back to the focal point of this thesis, which is how we

can aggregate these detected situations into a environment representation that makes them actionable for autonomous missions.

3-7 Scene graph to BOS-Graph pipeline for simple situations

Simple situations are situations composed of one object. The aim of this section is to show that using our pipeline for simple situations, the information required to construct the BOS-Graph is already present in a 3D (Spatial) Scene Graph (3DSG). In virtual simulation conditions 3DSGs can already be extracted in a data-driven way. The BOS-Graph provides a higher level of SA, to support this we will quickly review Chapter 2. 3DSG are a novel method for representing 3D scenes that incorporates hierarchical, metric, and object-level information [22]. The scene graph is hierarchically structured in 5 layers as was shown in Figure 2-3 before. However, in the context of a mission, it contains a great deal of irrelevant or not yet actionable information; therefore, we wish to further condense the 3DSG to provide direct insight into the mission status. It can be argued that scene graphs only provide level 1 SA because they only provide an overview of where objects are located in a spatial hierarchy. We want a representation that gives us increased situational awareness in the context of goal autonomy. For level 2 situational awareness, we must comprehend the surrounding environment, which we will interpret following Assumption 1.

Equation 3-14 shows high level how we can map a SG $\mathcal{G}_{\text{SG}}$ to the BOS-Graph $\mathcal{G}_{\text{BOSG}}$ using prior knowledge in the form of situational affordances $\mathcal{H}$.

$$f(\mathcal{H}) : \mathcal{G}_{\text{SG}} \rightarrow \mathcal{G}_{\text{BOSG}} \tag{3-14}$$

For each edge e_{SG} in *Layer 2: Objects and Agents* of a 3DSG we can use the object labels and affordances $\mathcal{H}$ to extend the edges with the possible behaviors to create BOS-Graph edges e_{BOSG}, as shown in Equation 3-15.

$$e_{BOSG} = \langle e_{SG}, b : b \in \{h : h \in \mathcal{H} \wedge h.s = e_{SG}.v_{\text{target}}.s\}\rangle, \quad \forall e_{SG} \in \mathcal{G}_{SG,\text{layer 2}} \tag{3-15}$$

The edge weights are determined using the weight mapping function f that takes the cost of the behavior and the positions of the target and source node of the edge as inputs.

$$e_{BOSG}.w = f(b.c, e_{SG}.v_{\text{source}}.x, e_{SG}.v_{\text{target}}.x) \tag{3-16}$$

Furthermore, in *Layer 3: Places* all edges connecting *place nodes* can be extended in a similar way. This can be used to make the type of traversal that is possible explicit (walking vs. driving vs. flying). This would be especially useful in deployments of legged robots together with wheeled robots in environments containing stairs.

3-7-1 Example: simple pipeline

This example will demonstrate the transformation with a toy problem. We start with a simplified scene graph, where we have squashed layer 2 and 3 together. This makes it essentially a navigation graph with object landmark nodes. This input graph is shown in Figure 3-13.

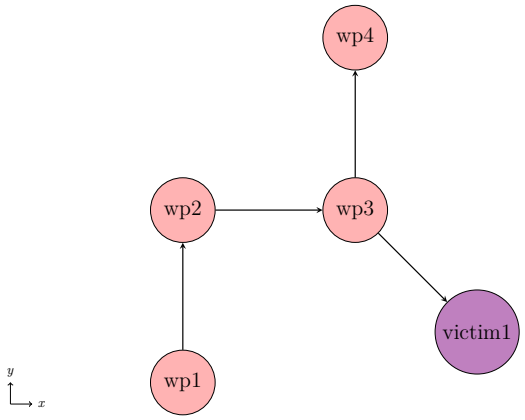


Figure 3-13: An illustration of a simplified SG that we we use as the input for Equation 3-15 as an example.

Using the proposed transformation from the previous section the 3DSG can be transformed to a BOS-Graph as shown in Figure 3-14. This BOS-Graph provides a higher level of situational awareness, because we have contextualized it in our mission objectives.

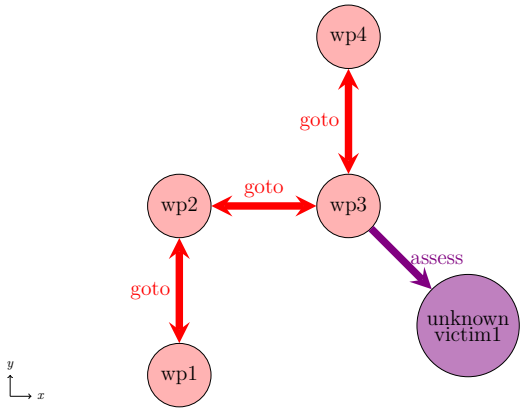


Figure 3-14: An illustration of the BOS-Graph we obtain as the output of Equation 3-15 for a simplified example.

3-8 Qualitative evaluation

In this section we will evaluate the conceptual pipeline proposed in this chapter. Because the pipeline is conceptual, we will evaluate it qualitatively.

3-8-1 Evaluation of the increase in situational awareness compared to spatial scene graphs

Here we shall evaluate if we can state that the situational awareness is increased if the BOS-Graph is used. This evaluation is based on the framework of Endsley [38], as described in Chapter 2.

With a 3DSG we can ask and answer questions like "what is where?". However this does not provide insight within the context of the mission yet. By using the affordances to transform it into a BOS-Graph we can now ask questions like "what can be done where?" and "What are the effects on my goals if I do X there?". Also "What tasks are available and how do they compare in terms of usefulness?" can now be asked. The edges of the BOS-Graph give insight into the space of relevant behaviors in a specific environment.

Insight for human operator Imagine a fireman who arrives late to the incident, by looking at the BOS-Graph he or she will get instant insight into the progress of the mission and its current state. Additionally, because the agent can express its plans in terms of the BOS-Graph, it also becomes an environmentally contextualized language to explain what the agent "thinks" it is doing.

The BOS-Graph is the schema in which we can describe repeat missions Once a BOS-Graph has been obtained for an environment, it can be used intuitively to define linear missions. In environments where the agent will be deployed periodically and where the environment can change between deployments, this is especially useful. Compared to AutoWalk (Section 2-5-2), the work involved in redefining a linear mission now consists of selecting a new sequence of behaviors over BOS-Graph, rather than having to manually teleoperate the new mission from scratch.

3-8-2 Benefit to other autonomy subsystems

Other subsystems can also benefit from the centralized and contextualized environment representation in the BOS-Graph.

Comprehensive task utility heuristic We now have a representation that can naturally provide the utility function (though graph search) for each task and agent pair. This is the key information needed to perform task allocation. Task allocation is a complex topic, with many algorithms to handle the computational cost of the combinatorial task allocation problem. However, for these task allocation solvers, the saying also goes: "garbage in, garbage out". The BOS-Graph provides a cheap and high-fidelity heuristic for such task allocation solvers. This enables moving from offline task allocation to online task allocation because we can continue to evaluate the utility of each task online.

3-8-3 Comparison with AutoWalk

Here we will compare our method against Autowalk by Boston Dynamics (**??**.

The main limitation of AutoWalk is that we need to manually record a mission through teleoperation. Essentially creating a sequence of GoTos and TakeRecording actions. To mitigate this limitation, prior knowledge has been made independent of the method of constructing the representation.

In summary, prior knowledge enters the pipeline only at the following different points.

1. **Situations $\mathcal{S}$**
   Set of situations, composed of objects that share particular relations in the environment. Computer vision and reasoning pipelines must be created that can go from images to the classification of situations. This will require large training datasets, which can be expensive to obtain. Modern data augmentation and simulation techniques, such as domain randomization, could help to obtain the training data for these pipelines.

2. **Behaviors $\mathcal{B}$**
   We need the costs of the behaviors $\langle c \in \mathcal{B}.c \rangle$, the capabilities they require, and how they mutate the BOS-Graph upon success or failure. Behaviors can be developed, tested, and extended independently. Different software developer teams can work on different behaviors, independently of each other parallezing their human effort and avoiding bottlenecks.

3. **Objectives $\mathcal{O}$**
   For each objective the reward has to be set $\langle r \in \mathcal{O}.r \rangle$. Objectives are essentially abstract tasks. Objectives specify abstract desirable outcomes in the context of a mission. The relative importance of objectives should be authored by tuning their rewards.

4. **Affordances $\mathcal{H}$**
   Triplets linking situations, behaviors, and objectives. Affordances are basically "for every X, do Y, to obtain Z" statements. These relations need to be authored manually. The mission is specified in terms of these affordances.

If, instead of authoring the entire mission, we decompose it into 4 parts, the parts can be reused. This decomposition is illustrated in Figure 3-15.

Figure 3-15: Affordance authoring. The decomposition into authored affordances which pair together authored objectives, behaviors and situations provides a more reusable mission decomposition.

Exhibit: novel environment The robot can be deployed in a novel environment without any new authoring efforts, as opposed to AutoWalk. This makes the method suitable for the SAR domain.

Exhibit: novel objective If we wish to add a novel objective to the mission, we need to author a new affordance which includes this the outcome to satisfy the objective. The end-user also needs to tune the reward parameter of the new objective. It is likely that the novel objective also requires a new behavior to achieve it. This would require the authoring of a new behavior in the form of development and testing. It is also likely that we will need to detect a new situation to trigger the behavior.

Exhibit: novel goal A novel goal also requires determining the objectives that contribute to that goal.

Discussion The proposed method outperforms Autowalk in terms of reusability and can easily be used to perform missions in novel or frequently changing environments. The main downside of authoring remains the human effort required to do it.

3-9 Concluding remarks

In this chapter, we provide the information pipeline to transform a Scene Graph (SG) into a BOS-Graph. With this analysis, we address the first research question: **Assuming that we can robustly perceive situations, how can we best structure prior knowledge to create an information pipeline to construct more actionable environment representations?** The method of decomposing missions for this method has been proposed. The required prior knowledge and how the data should be structured have been described. In this work we assume we can detect situations, the algorithm for transforming a SG into a BOS-Graph under this assumption has been described.

The BOS-Graph is a novel actionable environment representation, which increases the level of situational awareness provided compared to contemporary environment representations. We conduct a qualitative comparison between the SA provided by the BOS-Graph and that provided by the SG. We perform a qualitative evaluation of how the SA provided by the BOS-Graph compares against the SA of the SG.

The results in this chapter indicate that while the BOS-Graph provides a higher level of SA, it is less versatile outside of the intended domain. This is consistent with expectancy, as a higher context always necessitates further assumptions.

Chapter 4

Planning over a Behavior-Oriented Situational Graph

This chapter presents a reference implementation for task planning on the BOS-Graph. In this chapter, we make the assumption that a BOS-Graph is available, so we can demonstrate the task allocation pipeline. First, in Section 4-1 the dynamic data structures used for planning are exhibited. Second, in Section 4-2 we formally describe the planning and task allocation problem. In Section 4-3 the algorithm for performing the planning, task allocation, and plan execution is described. The method is illustrated using a static world example in Section 4-4 and a dynamic world example in Section 4-5. Finally, the chapter concludes with a summary of the benefits and limitations of the proposed framework.

4-1 BOS-Graph dynamic data-structures

The diagram in Figure 4-1 provides an overview of the dynamic data structures that are filled during the mission. The data structures contain references to each other using *universal unique identifiers* (uuid); this prevents unnecessary duplication of data. The dynamic data structures are filled with data using perception systems during the mission. The static data structures of the previous chapter, Chapter 3, are different because they contain authored prior or expert knowledge, similar to a schema. Instead these dynamic data structures contain online information about the current environment, similar to instances. This distinction allows the authored efforts to be reused in novel environments, in the form of prior knowledge in the static data structures. This allows us to apply the proposed pipeline to unseen environments, on the condition that the schema (and thus knowledge) applies just as well to novel environments. We thereby assume we still operate within the same task domain.

As mentioned before, note that when indexing properties of nested data structures or data structures that reference other data structures, we employ a dot indexing notation similar to most object-oriented programming languages as shown in Equation 3-1.

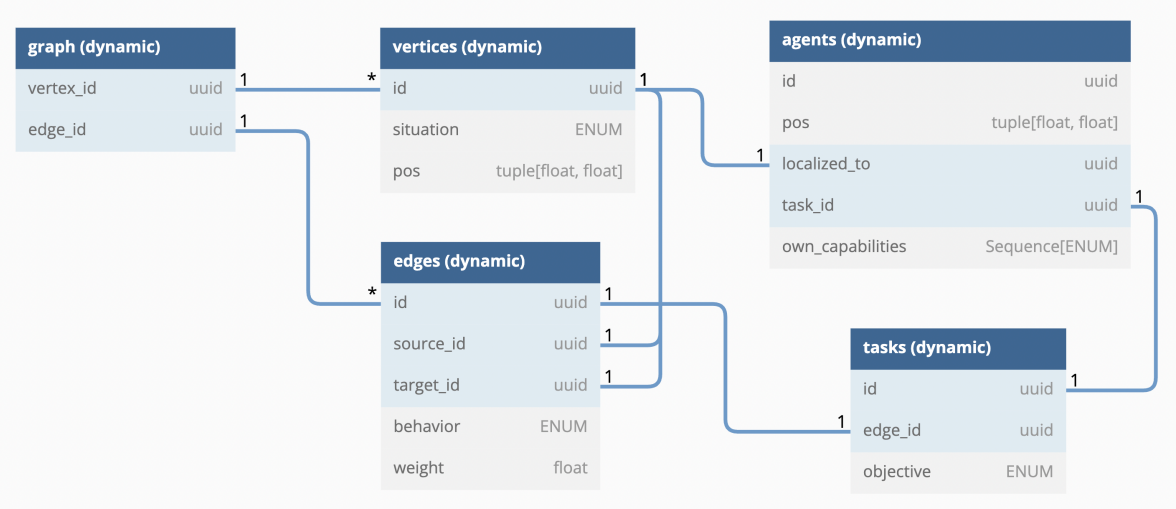


Figure 4-1: Database diagram of the dynamic data structures involved. The graph contains nodes and edges. Nodes (vertices) correspond to situations. Edges correspond to behaviors that the agent can perform from the source node targeted at or towards the target node. A set of agents is performing the mission. There is a set of currently available tasks which have been discovered in the environment. A task has an associated edge, but not every edge has an associated task. Each agent can be assigned a task.

4-1-1 Tasks

We define tasks as outcomes, not processes. In general, tasks could be discrete, such as "obtain the assessment of victim X", or continuous such as "patrol area Y". In this work we only consider discrete tasks; the exploration is discretized in specific "obtain exploration of frontier X" tasks as we will see later. A task τ in the set of all tasks $\mathcal{T}$ is defined as a tuple. Each τ consists of an identifier `id`, an edge e and an objective o. Hence, a task is defined as what to do and where to do it, which we represent with an edge.

$$\begin{aligned} \tau &:= \langle \texttt{id}, e, o \rangle \\ \mathcal{T} &= \{\tau_i\}^{i=1,\dots,N_\tau} \end{aligned} \tag{4-1}$$

How do tasks relate to objectives and behaviors? We should repeat the assumption that for each task there exists a behavior whose success effects accomplish that task. Tasks are instances of objectives; an objective could be 'asses every victim encountered' while the resulting task instance would then be to 'obtain the assessment of a specific victim'. This assumption allows us to directly link tasks with behaviors represented as edges in the BOS-Graph.

4-1-2 Agents

An agent a in the set of all agents $\mathcal{A}$ is represented by a tuple. The tuple consists of a unique identifier `id`, a global position x, a node to which it is localized on the BOS-Graph v, a set of its capabilities $\mathcal{K}_a$, and finally its currently assigned task τ.

$$a = \{\texttt{id}, x, v, \mathcal{K}_a, \tau\}$$
$$\mathcal{A} = \{a_i\}^{i=1,\dots,N_\mathcal{A}} \tag{4-2}$$

4-1-3 Capabilities

To perform a specific behavior an agent has to possess the capability k required to perform that behavior. The set of all possible required capabilities $\mathcal{K}$ is obtained from union of the agent capabilities $\mathcal{K}_a$ and behavior capabilities $\mathcal{K}_b$.

$$\mathcal{K} = \mathcal{A}.\mathcal{K}_a \cup \mathcal{B}.\mathcal{K}_b \tag{4-3}$$

Examples include the capability to open doors, asses victims visually, assess victims by audio.

4-1-4 Overview of the dynamic data structures

Having introduced the dynamic data structures in this chapter, we can now combine them with the static data structures of Chapter 3. An overview of the data structures involved in the pipeline is given in Figure 4-2.

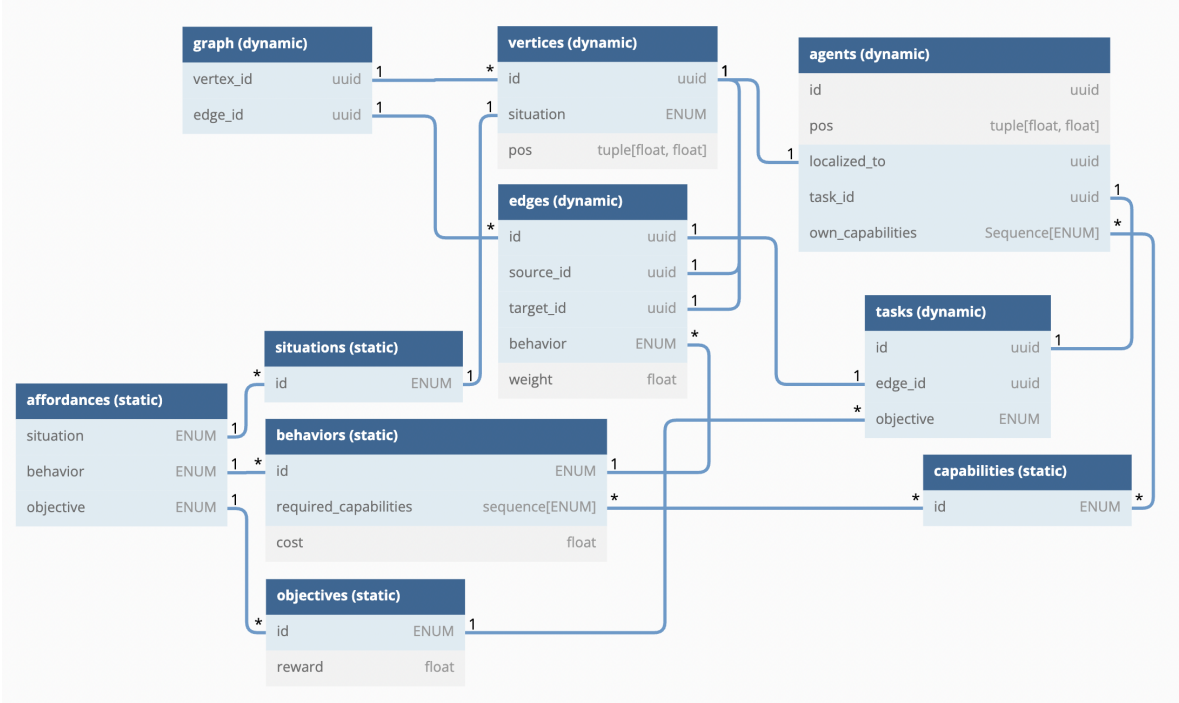


Figure 4-2: Overview of the data structures involved in the BOS-Graph as a database diagram. Please note that the data structures at the top are the only ones that change during the mission (dynamic), while the others contain prior knowledge and remain static. The affordance is the primary data structure, connecting objects, behavior skeletons, and objectives. Every node possesses an object type. For each behavior template, certain agent capabilities must be met. All capabilities are known apriori. Each objective has a fixed reward; by linking each task to its objective, the reward can be used to identify the most efficient task.

4-2 Planning problem on the BOS-Graph

This section will formalize the BOS-Graph, the planning problem, and the plan domain for which the method is applicable. Formalization is heavily based on set theory and function mappings. The data structures involved are divided into static structures that contain authored prior knowledge and dynamic structures that are populated using data-driven perception techniques as the mission progresses.

4-2-1 Planning problem

This section addresses how we can go from a BOS-Graph $\mathcal{G}$, a task τ, and an agent a to a plan for that agent to achieve that task $\pi_{a,\tau}$. This planning problem is captured by the planning problem tuple $\mathbb{P}$ in Equation 4-4.

$$\mathbb{P} : \{\mathcal{G}, \tau, a\} \rightarrow \pi_{a,\tau} \tag{4-4}$$

Finding the plan for a task To find the optimal plan for a specific task on the BOS-Graph there exist many graph search algorithms. The plan is some notion of the "shortest path" on the graph between the node to which the agent is localized $a.v$ and the target node of the edge of the task $\tau.e.v_{\text{target}}$ based on edge weights. This target node is inferred from the edge of the task. F is a generic graph search algorithm that finds this shortest path on the graph.

$$\pi_{a,\tau} = F(\mathcal{G}, a.v, \tau.e.v_{\text{target}}) \tag{4-5}$$

Costs and weights This paragraph explains how costs c and weights w are used in the planning pipeline. Firstly, A cost factor $e.b.c$ is associated with every behavior b. The cost factor c depends on the resources required to execute the behavior, most of the time either execution time or energy. Secondly, the weight of edges $e.w$ in the graph is determined by a function f using the cost factor $e.b.c$ of the behavior and the geometric properties of the edge, in this work only length $v_{\text{source}}.x - v_{\text{target}}.x$. We note that it would be interesting to extend this with terrain traversability and risk. The sum of these edge weights is the cost of a plan $\pi.c$ as shown in Equation 4-6.

$$\begin{aligned} e.w &= f(e.b.c, e.v_{\text{source}}.x, e.v_{\text{target}}.x) \\ \pi.c &= \sum_{e \in \pi_{a,\tau}} e.w \end{aligned} \tag{4-6}$$

Plan topology The plan topology is important because the proposed method only solves a specific class of plans. A plan $\pi_{a,\tau}$ is constrained in two ways. The source node at the first edge of the plan should be equal to the node to which the agent is localized $a.v$. The final edge of the plan must be the edge of the task $\tau.e$. This is defined in Equation 4-7. Note that the plan is a sequence of edges, not nodes. Also note that an overview of the BOS-Graph data structures is available in Figure 4-2.

$$\pi_{a,\tau} = \langle e_1, ..., e_{n-1}, e_n \rangle : \quad e_1 \in \{e : e.v_{\text{source}} = a.v\} \quad e_n = \tau.e \tag{4-7}$$

4-2-2 Task allocation problem

This subsection addresses how the planning from the previous section can be used to determine a utility for each task, allowing us to compare the tasks and select the optimal one for the agent.

Single agent task allocation Given a BOS-Graph $\mathcal{G}$, a solution to the task allocation problem $\mathbb{T}$ is to find the plan π corresponding to the optimal task τ^* for agent a. In the proposed method, task assignment is based on evaluating the plans for all tasks τ the agent is capable of.

$$\mathbb{T} : \{\mathcal{G}, \mathcal{T}, a\} \rightarrow \pi_{a,\tau^*} \tag{4-8}$$

The Utility function U that is used to evaluate tasks for a specific agent and graph is defined in Equation 4-9. Utility is the reward for the objective of the task $\tau.o.r$ divided by the cost of the plan for that task $\pi.c$ (from Equation 4-6). The cost of the plan is defined as the sum of all edge weights of the edges in that plan.

$$U(\tau) = \frac{\tau.o.r}{\pi_{a,\tau}.c} \tag{4-9}$$

It is now possible to optimize for utility $U(\tau)$ over all tasks in $\mathcal{T}$ to find the optimal task for that agent. Finding this optimal task τ^* solves the task allocation problem $\mathbb{T}$ as defined in Equation 4-8.

$$\tau^* = \arg\min_{\tau} U(\tau), \quad \forall \tau \in \mathcal{T} \tag{4-10}$$

The plan for the optimal task can now also be calculated directly, combining Equation 4-10 with Equation 4-5.

$$\pi_{a,\tau^*} = F(\mathcal{G}, a.v, \tau^*.e.v_{\text{target}}) \tag{4-11}$$

Selecting tasks for multiple agents is an active research area that focuses on computational methods to optimally and efficiently assign tasks to teams of agents [68, 69].

4-3 The Algorithm

This section describes a reference implementation of a pipeline that uses the BOS-Graph for task allocation, task planning, and plan execution. The following algorithm is run after each behavior execution, allowing it to exploit new information.

High-level overview of proposed pipeline The proposed pipeline consists of three main steps; completing a mission effectively involves looping through these steps until all tasks have been completed.

1. Filtering the BOS-Graph to the agents capabilities
2. Finding the optimal task and plan
3. Plan execution

Algorithm 1 provides the overview of the proposed method. For each agent, the following steps are taken at each step. First, the BOS-Graph is filtered based on the agent's capabilities $\mathcal{K}$. Then, based on the current location of the agent, all tasks are evaluated based on their utility U to determine the optimal task τ^* for that agent. Once the optimal task has been determined, a plan π is obtained for that task. The first step of that plan is executed by the plan_executor. After executing this behavior, we process the new information to obtain the successor graph $\mathcal{G}_{post}$.

Algorithm 1 High-level overview of the implementation of the reference planner.

```
for a in A do
    G' = filter(G, a)
    τ* = find_optimal_task(a, G', T)
    π = find_plan_for_task(a, G', τ*)
    G_post = plan_executor(a, G', π)
```

4-3-1 Filtering the BOS-Graph to the agents capabilities

The first step in the pipeline is filtering the BOS-Graph to the capabilities of the agent. Filtering the BOS-Graph is required to constrain the planner to only find plans that match the capabilities of the current agent. Filtering means that we obtain an augmented graph where all edges that do not match the capabilities of the agent are removed. For example, remove the edges corresponding to opening doors for agents that do not possess the required manipulation capabilities.

Filtering the BOS-Graph to the capabilities of the agent is defined in Equation 4-12. $\mathcal{E}_{capable}$ is the set of edges for which the capabilities required for the behaviors in those edges are a subset of the capabilities of the agent $a.\mathcal{K}_a$. To perform the filtering operation, we find all edges which the agent is incapable of and subtract those edges from the graph to obtain our filtered graph $\mathcal{G}'$.

$$\begin{aligned} \mathcal{E}_{capable} &= \{e : e.b.\mathcal{K}_b \subset a.\mathcal{K}_a\} \\ \mathcal{E}_{incapable} &= \mathcal{E} \setminus \mathcal{E}_{capable} \\ \mathcal{G}' &= \mathcal{G} \setminus \mathcal{E}_{incapable} \end{aligned} \tag{4-12}$$

4-3-2 Finding the optimal task and plan

The challenge now becomes to select a task and obtain a plan using the filtered BOS-Graph. The planning method used is based on graph search. Because the edges in the graph represent behaviors, a path over the graph is a sequence of behaviors. Such a path over the BOS-Graph i.e. the sequence of behaviors, is a plan π.

The algorithm that determines an agent's optimal task and plan is depicted in Algorithm 2. The first step is the selection of tasks, in which the optimal task τ^* is determined. To compare tasks, we evaluate the utility for each task. Given the fact that the cost of the plan $\pi_{a,\tau}.c$ is required to calculate this utility, we simultaneously keep track of the plan with the highest utility, and in the end we return it as the optimal plan π^*.

Algorithm 2 Finding the plan for the optimal task by evaluating the plans for each task.

```
1: function FIND_OPTIMAL_TASK_AND_PLAN(𝒢, a, 𝒯)
2:     best_plan_utility = 0
3:     task_nodes = [ τ.v for τ in 𝒯]
4:     path_costs, paths = Dijkstra(𝒢, task_nodes, a)
5:     for τ in 𝒯 do
6:         current_plan_utility = τ.r / path_costs[τ.v]
7:         if current_plan_utility > best_plan_utility then
8:             π* = paths[τ.v]
9:             best_plan_utility = plan_utility
10:    return π*
```

4-3-3 Plan execution

Finally, the plan has to be executed, the pseudo-code for the executor is shown in Algorithm 3. At each step, the upcoming edge of the plan is obtained. Then, the behavior associated with that edge is executed. If the behavior returns a failure the upcoming edge is removed from the BOS-Graph and the plan executor returns failure. If a task is associated with this edge, then that task is also removed. In the next step the planner will now have to find a different path, and perhaps now a different task is more optimal. If the behavior returns success, we determine whether this was the last step in the plan. If it was, we consider the task completed and remove it from the task list $\mathcal{T}$. If the plan contains additional steps, we return success from the executor.

Algorithm 3 The steps for executing a step of a plan.

```
function PLAN_EXECUTOR(π, G, a, H) → Plan_Step_Result_Enum
    upcoming_edge = π[0]
    next_behavior = upcoming_edge.b
    behavior_result = next_behavior.run(G, a, H)
    if behavior_result != SUCCESS then
        G.remove(upcoming_edge)
        return FAILURE
    else
        if length(π) == 1 then
            destroy_task()
            return COMPLETED
        return STEP_SUCCESS
```

4-4 Example: planning in a known static world

In this section, the planning component is tested in a simple scenario to illustrate its functioning. The agent receives a BOS-Graph representation of a static world, as shown in Figure 4-3. The agent is tasked with assessing unknown victims in the environment. We show that the planner with the BOS-Graph can be used to effectively allocate tasks by evaluating the utility of each task, based on its plan cost and reward. Both victim assessment tasks result from the "obtain assessment all victims" objective, which has a reward of 10 here.

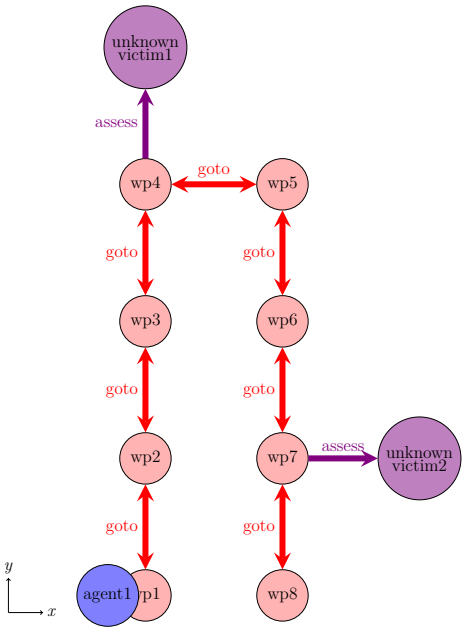


Figure 4-3: Illustration of the BOS-Graph of scenario 1 at time step 0. The agent is at wp1 and there are two unknown victims present.

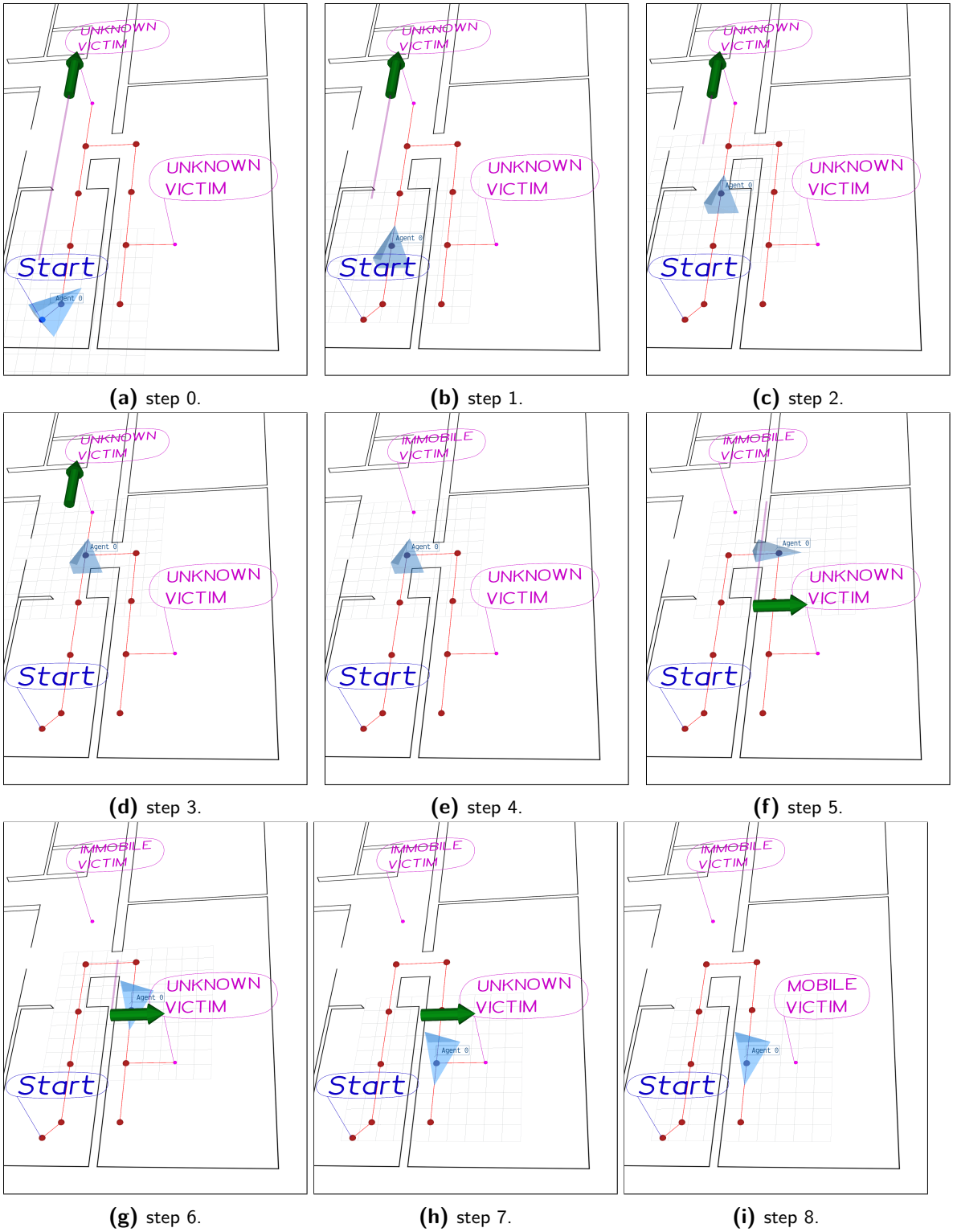
The utility of each task from the perspective of a robot is based on its rewards and plan cost.

Table 4-1 gives a short overview of the utilities for task 0: "assess victim 1" and task 1: "assess victim 2".

Table 4-1: overview of the the task utilities in Figure 4-3 at time step 0. In this scenario assessing victim 1 has higher utility than victim 2. Utility is computed according to Equation 4-9.

Task (τ)	Reward (r)	Cost (c)	Utility (U)
assess victim 1	10	4	2.5
assess victim 2	10	7	1.43

Simulation procedure The scenario is simulated stepwise in the simulator. Screenshots of simulation steps are shown in Figure 4-4.



(a) step 0. **(b)** step 1. **(c)** step 2.

(d) step 3. **(e)** step 4. **(f)** step 5.

(g) step 6. **(h)** step 7. **(i)** step 8.

Figure 4-4: The progression of example scenario 1. The agent is the blue pyramid. Victim situations are the pink nodes. The red line towards the green arrow is the plan. The agent goes for the first unknown victim from step 0 to step 4. At step 4 the victim is assessed. Note how the victim label changes from "UNKNOWN VICTIM" to "IMMOBILE VICTIM". Subsequently the agent goes to assess the second victim.

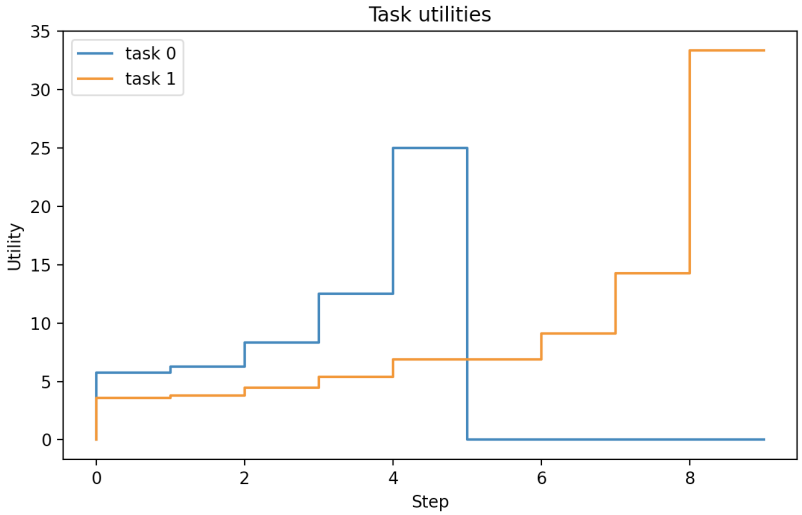


Figure 4-5: The utility plot for Scenario 1. At first, task 0 (assessing victim 1) has a higher utility than task 1 (assessing victim 2).

Figure 4-5 shows the task utility at each time step based on reward and cost. The cost is dynamically determined by graph search on the BOS-Graph, the reward is obtained from the objective $o.r$ associated with the task.

Discussion This example illustrates how Algorithm 2 can evaluate the BOS-Graph to generate plans. And then use these plans to assign the optimal task to each agent. The question of how to balance reward and cost is left to the end user. This work aims to provide a set of tuneable parameters which can then be adjusted based on the particulars of the desired use case.

4-5 Example: planning in a known dynamic world

This section demonstrates the planner in a dynamic environment. The scenario is that the known representation of the world no longer corresponds to the world because the world has changed. A new opening has emerged and the pipeline can take advantage of this new information. It updates the BOS-Graph, rendering the previous task no longer optimal; consequently, we dynamically switch to the optimal task.

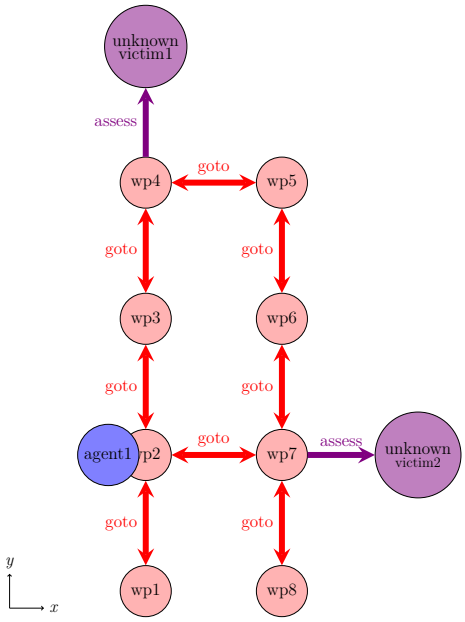
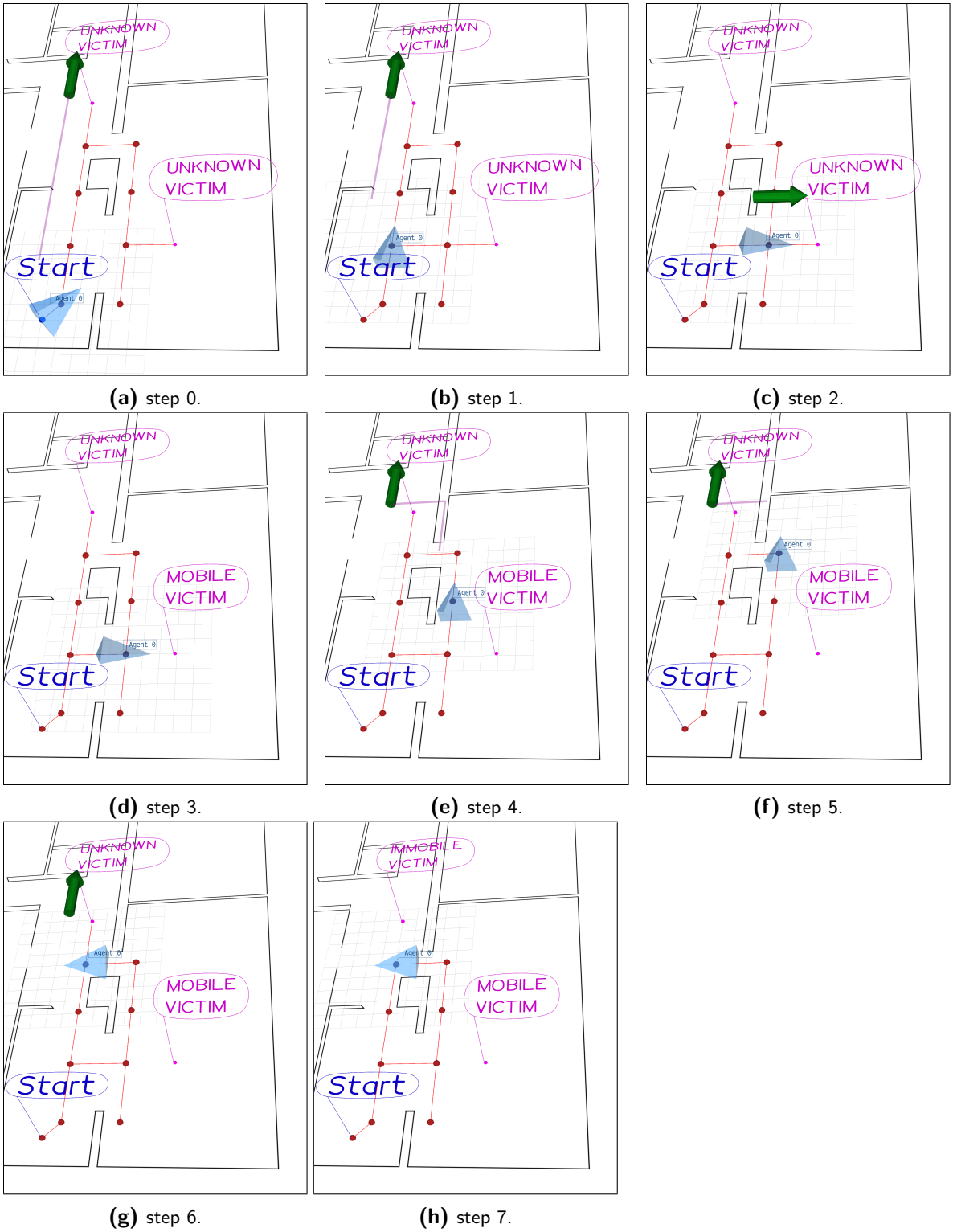


Figure 4-6: The BOS-Graph of scenario 2 at step 1. Note that it is different from the scenario in Figure 4-3 in that a new shortcut is formed between wp2 and wp7.

Simulation procedure While executing the plan, we obtain new information, and the new information can be exploited to dynamically update the plan during execution. In this scenario, we look at a dynamic scenario; here the world has changed and no longer matches the BOS-Graph. While executing the plan we obtain new information, the new information can be exploited to dynamically change the plan during execution.



(a) step 0. **(b)** step 1. **(c)** step 2.

(d) step 3. **(e)** step 4. **(f)** step 5.

(g) step 6. **(h)** step 7.

Figure 4-7: The progression of example scenario 2. The agent is the blue piramid. The red line towards the green arrow is the plan. Victim situations are the pink nodes.

The agent initially chooses to assess victim 1, however upon performing its plan the agent

observes that a shortcut has become available between wp2 and wp7. This shortcut is added to the BOS-Graph. When reevaluating the plans for both tasks at Step 2, the cost for has become lower. Improving its utility for the agent, as shown in Figure 4-8. The agent switches its task and decides to assess victim 2 instead.

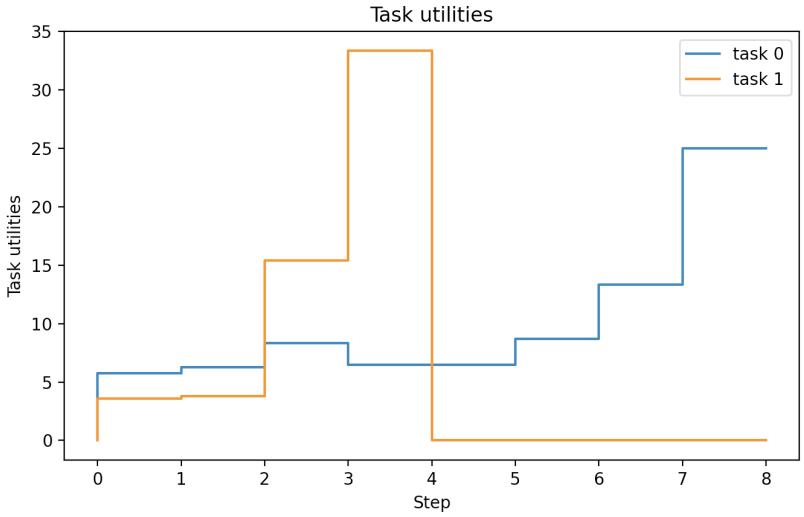


Figure 4-8: The utility plot for Scenario 2. Note that at time step 2 the agent perceives the shortcut on the graph and the utility for task 1 changes. The agent switches its task and decides to assess victim 2 instead. Compared to scenario 1 we now require one less step to assess both victims.

Discussion Because we employ Dijkstra, we can effectively compute the task utilities, allowing us to do so at each step online. The advantage of this is that we can directly utilize new information in the form of updates to the BOS-Graph. This enables robots to respond more quickly to dynamic environments and to discard old plans in favor of new ones. Comparing the results of Figure 4-8 with those of Section 4-4 reveals that assessing both victims required one less step. The task allocation of this method is greedy and is therefore not guaranteed to be optimal. However, on average, it is preferable to act on new information in the SAR domain because we have already justified the greedy assumption. In addition to being required to operate in dynamic environments, we will also see an increase in average performance.

4-6 Concluding Remarks

In this chapter, we described how the BOS-Graph can be used for planning. With this analysis, we addressed the second research question: **How can we exploit actionable environment representations to make robot task selection and planning more light weight?** Since planning is not the focus of this thesis, we provide a simple reference implementation of a planner. We described how BOS-Graph can be used for task planning, task allocation, and task execution using this reference planner. The defined dynamic data structures capture the agent, tasks, and their interdependence through capabilities. The planning and task allocation problems have been formalized. The algorithm has been decomposed to provide insight into its individual steps.

The provided reference planner is somewhat limited compared to other planning methods in the robotics domain such as symbolic planners, since it can only plan for "go from A to B and do C" type tasks. This matches our main type of goal in SAR, which is the collection of information.

Using two examples of toy problems, the proposed planning pipeline was demonstrated. The first example demonstrated how utility calculation guides the assignment of tasks in a static environment. The second example demonstrated that the provided reference planner can be executed at each step due to its low computational cost. The system can directly exploit novel environmental data, typically resulting in more effective missions that require fewer steps to achieve tasks.

From these two examples, we can conclude that using our novel situational affordances perspective we can create a world representation that can easily be integrated with planners, for example with the simple reference planner we provided. However, more interestingly, the work in this thesis could also be extended with probabilistic planners to accommodate the uncertainty of the real world.

Finally, we can also conclude that we now provide SA in the context of the mission goals and capabilities of our agents, because the BOS-Graph essentially maps all possible behaviors in its edges.

Chapter 5

Task discovery

This chapter presents the synthesis between representation from Chapter 3 and planning from Chapter 4 to realize a task discovery pipeline. The pipeline allows the robot to explore unseen environments and create an inventory of available tasks in the context of mission objectives using its authored prior knowledge, structured as a situational affordance. This task inventory is aggregated in the Behavior-Oriented Situational Graph (BOS-Graph). Section 5-2 details the reference exploration algorithm that has been implemented. Section 5-3 details the procedure for online task discovery while performing the mission. This is followed by extensive experimental validation on a physical Spot robot in Section 5-4. The chapter is then concluded with a summary of the benefits and limitations of the implemented method.

5-1 Robotic platform, assumptions, and limitations

This section will introduce the robotic platform that was used for experimental validation in the real world, in addition to the more general assumptions we make about any suitable platform and the available autonomy services running on it.

The Spot robotic platform by Boston Dynamic The experiments are performed on a Boston Dynamics Spot platform [2] as pictured in Figure 5-1, minus the arm. Spot is well suited for exploring unstructured and human-inhabited environments because of its quadruped locomotion. Compared to wheeled robots, legged locomotion allows much better navigation of cluttered and uneven terrain, including stairs. Spot comes equipped with a number of on-board autonomy services. Such as localization and collision avoidance. These services can be accessed through a well documented API[1] , which is one of the factors that really sets the spot robot apart from its commercial competition.

[1]For an overview of the API concepts see: https://dev.bostondynamics.com/docs/concepts/readme

Figure 5-1: The Boston Dynamics Spot robot [2] equipped with an arm, turning it into a mobile manipulator. Mobile manipulation is essential in an urban SAR scenario that contains doors. The arm end-effector also contains a camera, allowing it to collect images in hard to reach places.

Assumptions autonomy services To limit the scope of this thesis, we make some strong assumptions about the autonomy services available on our robotic platform. An centralized overview of these assumptions, repeating some of the earlier assumptions, is provided below.

- **Localization service**
  We assume that robust localization is possible in unseen environments. This equals a robust Simultaneous Localization And Mapping (SLAM) pipeline that operates in the background. At any time, we may query the robot's position in world coordinates.
- **Collision avoidance**
  We assume the robot can maintain a safety bubble in the form of a free space margin around itself. When the robot's current commands cause a collision, it can provide feedback. This feedback allows us to infer whether a behavior has been executed unsuccessfully.
- **Perception service**
  The perception service has two obligations to fulfil. Firstly, we begin with the assumption that the robot maintains a perception scene. This perception scene can be queried for objects that have been detected in the immediate vicinity of the robot at present. Secondly, the perception service provides an occupancy map of the perception scene. This information is essential to decide where the robot can explore.
- **Target-tracking service**
  We assume that object detections, that is, targets, can be uniquely identified spatially and temporally by tracking. This is essential to prevent multiple detections of the same situation.

The localization and collision avoidance assumptions are consistent with the services available on the Spot platform. Using vision and odometry, Spot provides localization. The robust perception and target tracking service go beyond the onboard services provided by Spots. We will demonstrate how fiducial markers are used as placeholders for perception and target tracking services in a subsequent section.

Local environment representation So far, we have discussed how to represent the global environment in the BOS-Graph. To construct this sparse global representation, information from the dense local perception scene must be aggregated. The dense local environment representation consists of both a 3D point cloud and poses of detected objects. This 3D point cloud is aggregated into a 2.5D cell map that represents the traversability of the local environment.

Limitations of Spot The robot in its base configuration with 4 stereo cameras has a limited sensor range with a radius of 2 meters. This sensor range is severely limiting for the SAR use case. A sensor range that could capture at least rooms would be desirable. As we show in the subsequent section, this limited sensor severely limits the efficiency of placing frontiers, as we cannot use next-best-view or coverage methods effectively.

5-2 Mobile robot exploration

In order to build the BOS-Graph online, we need exploration. In the previous section, the robotic platform was evaluated; with that information, we can now devise the algorithm for a reference exploration behavior implementation[2] on the Spot robot. To be clear, Spot does not support exploration out-of-the-box.

Considerations for exploration algorithm Exploration algorithms require a representation of the environment. They need to track the explored and unexplored areas of the environment. A typical way to do this is through frontier-based methods [70]. Frontiers are essentially the boundaries between explored and unexplored areas. Exploring a frontier results in a number of new frontiers, unless it's a dead-end. A planner then selects the next best frontier to explore among all available ones. There are several methods to evaluate the utility of a frontier. The most basic version is cost-to-go [25], here the utility is dependent only on the cost associated with visiting that frontier. More elaborate exploration methods, such as next-best-view [25], use information theory to include the expected information gained from each frontier in the selection process. Even more advanced is semantic information gain exploration [27, 31], here semantic information is used to guide the exploration. Due to the lack of sensor range in the base model of the Spot robot and exploration not being the main focus of the thesis, cost-to-go exploration was chosen in this work. The proposed pipeline can be easily extended with these more advanced methods, therefore, we note it in future work.

The exploration behavior algorithm Algorithm 4 shows the pseudocode for the exploration behavior. The first step is to goto the selected frontier node. Then if we successfully reach it we remove the frontier from the BOS-Graph. We add a waypoint at the pose of the agent, extending the BOS-Graph. Afterwards, a local grid is obtained from the agent. The local grid (occupancy map) is used to sample new frontiers, and these new frontiers are added to the BOS-Graph in line 9. Then, we use the local grid to check for shortcuts between the

[2] A video showcasing the reference exploration behavior that was implemented on the real robot can be seen on https://youtu.be/DFg-cLQyx2w.

existing waypoints and the new waypoint. Finally, we prune frontiers that are too close to the new waypoint.

Algorithm 4 The steps involved in the discretized exploration behavior

```
1:  function EXPLORE(𝒢, a, current_edge)
2:      result = goto(current_edge.v_target.x)
3:      if result == SUCCESS then
4:          𝒢.remove_frontier()
5:          𝒢.add_waypoint_from_pose(a)
6:          lg = a.get_local_grid()
7:
8:          new_frontiers = sample_new_frontiers(lg)
9:          add_new_frontiers_to_tosg(new_frontiers, lg, tosg, agent)
10:
11:         add_shortcut_edges_between_wps_on_lg(lg, 𝒢, a)
12:         prune_frontiers(𝒢)
13:     else
14:         𝒢.remove_frontier(current_edge.v_target)
15:         goto(current_edge.v_source.x)
```

Sampling frontiers Sampling new frontiers is done using the local grid. We sample points in a ring that connects to the agent without collisions on the local grid. An illustration of the first sampling step of a mission is shown in Figure 5-2.

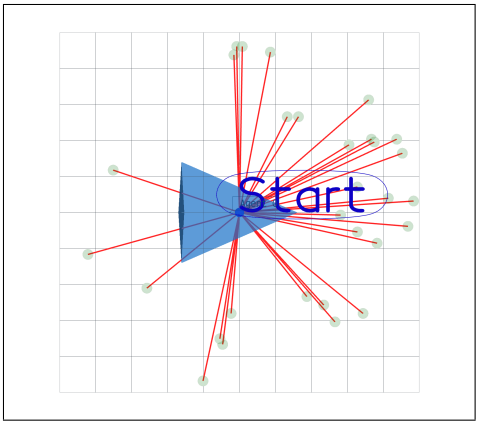


Figure 5-2: Illustration of the sampling procedure. Frontiers (green nodes) are sampled uniformly randomly in a ring around the robot (Blue Triangle). The red lines are the edges corresponding to goto actions connecting the waypoint the robot is at with the frontiers. The grid cells indicate the limited perception range of the robot where it can obtain its local cell map.

5-3 Task discovery

This section describes how to construct the BOS-Graph iteratively in an unknown environment. Because we now have the exploration behavior, the BOS-Graph can be constructed

iteratively rather than by transforming a scene graph into a BOS-Graph as we did in Section 3-7. Now, discovering tasks is as simple as monitoring the perception scene while continuously evaluating the most optimal task and executing a behavior step towards it.

Reference task discovery pipeline This section describes the task discovery pipeline. It describes how we go from object detection to situation detection and move to new nodes and edges on the graph. Although this thesis does not focus on perception and we make the assumption that we can detect situations (from Assumption 6), we provide a rough outline of what such a situational perception pipeline could look like below.

1. On a 360-degree camera, using a continuously running perception model detect object classes around the robot. Return detection event when a specific object relevant for one of our situations of interest is detected.

2. Process detection events to verify detected object instances are not part of any existing situation using target tracking service.

3. Start a more expensive specialized model such as scene graphs with neural motifs [42] to obtain a 2D scene graph of the current image scene.

4. Perform some form of automated reasoning over the resulting scene to obtain situation classification s.

5. Using the affordances database $\mathcal{H}$:

   (a) Update the graph with a node v for the detected situation s.

   (b) Update the graph with edges corresponding to the behaviors of the affordances $h.b$ (for notation see Equation 3-1).

   (c) Instantiate a task τ according to the objective of the affordance $h.o$.

6. Run the planning pipeline to select the best subsequent task τ^*.

The steps above describe a rough proposal for a sequential pipeline for going from perception to new nodes and edges in the BOS-Graph. We assume that in the coming decade we will see methods that can solve steps 1-4 jointly. This pipeline supports Assumption 6, which states that we can detect situations in the environment.

From situation detection to edges and tasks In this work, we focus on the instantiation of nodes and edges in the graph, given a detected new situation (Assumption 6). In Figure 5-3 the process to add new nodes and edges to the graph using situational affordances is illustrated. Once a situation has been detected, we use the affordances to instantiate the appropriate edges in between the situation node and the waypoint node the robot is currently at.

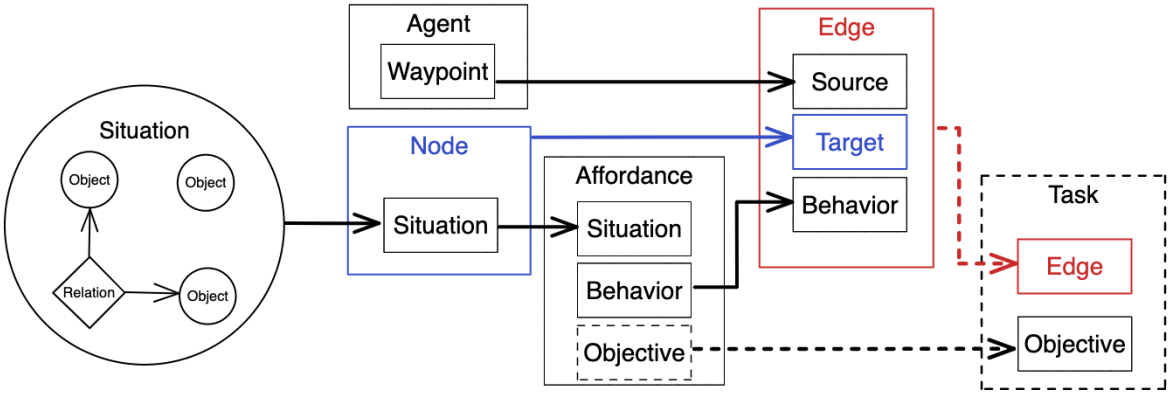


Figure 5-3: Affordance information pipeline. Dotted arrows are optional, depending on affordance. When a new situation is perceived we add a new node to the BOS-Graph. Using the affordances database we can then infer the edge to add to the BOS-Graph. The edge is added between the current position of the agent and the situation node. Depending on whether the affordance contains an objective a task is also created containing a reference to the edge. Blue stressed that a node becomes a target. Red stresses that tasks are associated with an edge.

Example: task discovery without task execution An intriguing use case is that of pure exploration. What would happen if we defined an agent with zero capabilities? The expected result is a representation of the environment of all meaningful processes as specified within the context of the mission. If we could autonomously detect all gauges that may require reading, all valves that may require opening and closing, and all assets that may require visual inspection, this would be extremely helpful, for instance, when inspecting a factory. This representation enables an operator to construct a mission using the BOS-Graph. In other words, all possible robot missions are contained within the BOS-Graph and can be expressed using BOS-Graph elements.

5-4 Experiment: real robot task discovery

Real robot test scenarios were performed using Snowboy, one of the two Boston Dynamics mobile Spot robots at TNO, as the main protagonist. The experimental mission was conducted in a real-world environment exactly as shown on the floor plan and 3d model in Figure 5-4. The robot is tasked with exploring the environment, finding 4 victims, and assessing the well-being of these victims.

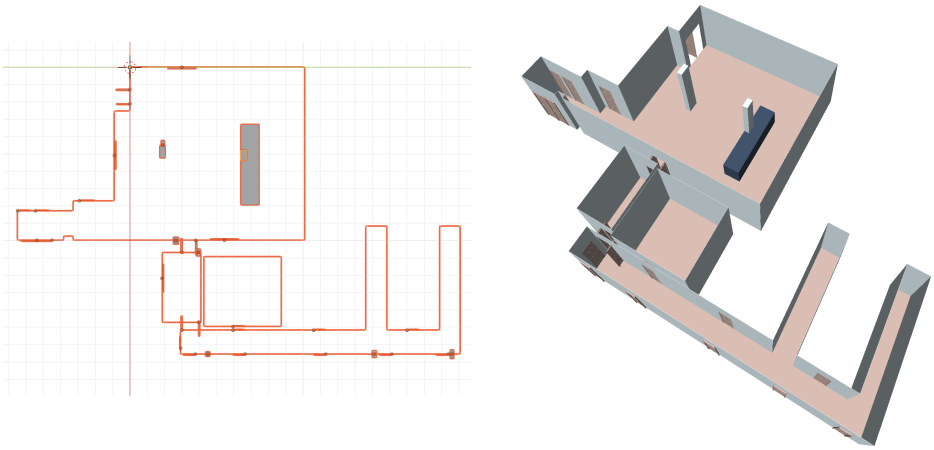


Figure 5-4: The floor plan and 3D model of the real world environment. Note the long narrow corridor and the open area.

Experimental procedure Fiducial markers are used to simulate the detection events of situations Figure 5-5, as the implementation of a perception pipeline is outside the scope of this work. Fiducials are visual markers composed of black and white squares, similar to QR codes used on mobile phones.

Figure 5-5: Fiducials are placed around the environment to simulate the situations we have defined.

Results In this scenario, the robot is deployed for a SAR mission on the real robot. Figure 5-7 shows the progression of the mission on Spot as captured by the BOS-Graph. The mission is in a test environment, as shown in Figure 5-4 and Figure 5-5.

In general, the agent tends to go to new frontiers. This is not the case if a victim has been found, then the agent prefers to go and assess that victim. Here there is a single robot with the capability to assess the victim, so the assessment task is always allocated at the next step after it has been discovered. The behaviors become more interesting when we deploy multiple

agents with different capabilities because then the discovery of a task by an incapable robot can result in another robot stopping its exploration to address that task.

Running the mission took 216 steps and 693.30 seconds. 85 nodes and close to 400 edges were accumulated in the BOS-Graph during the mission, as shown in Figure 5-6.

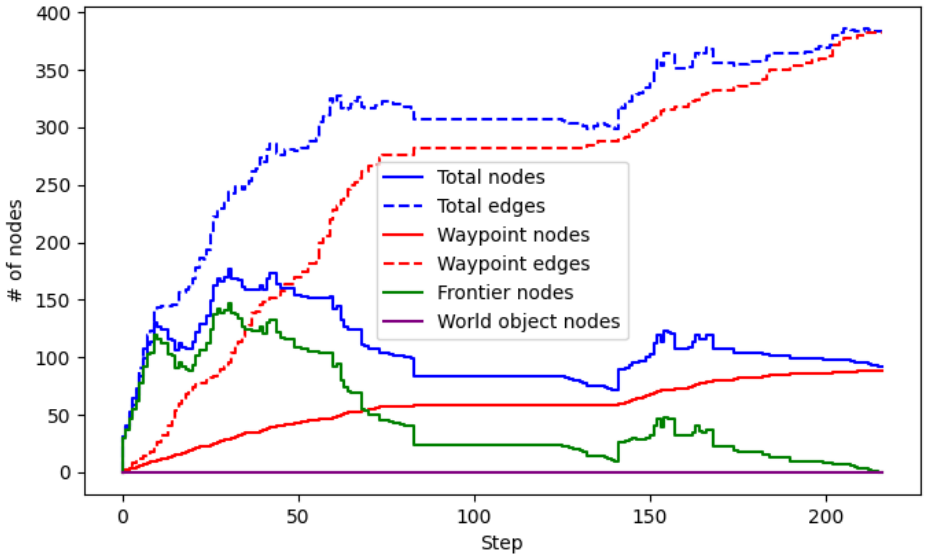
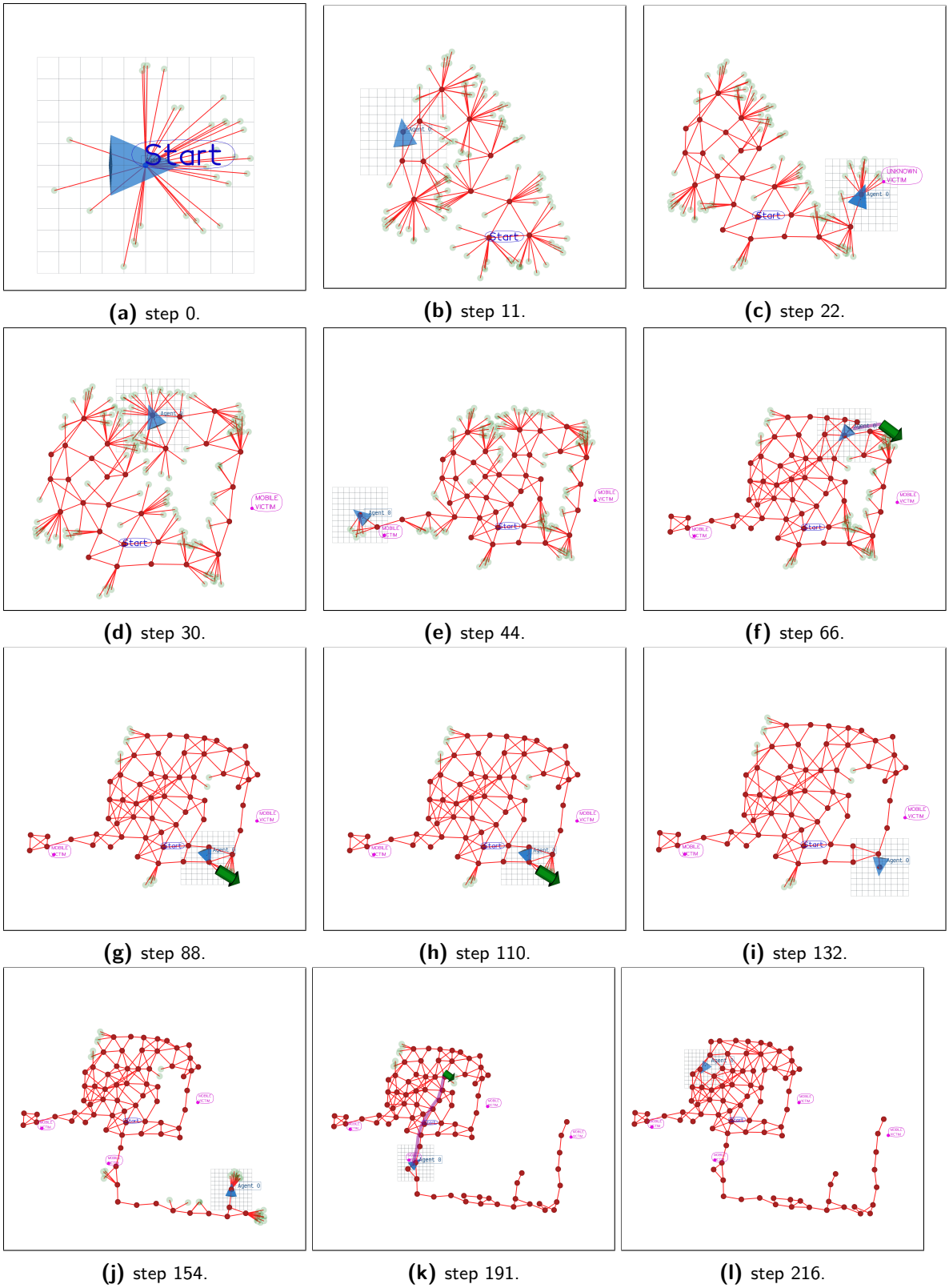


Figure 5-6: The number of nodes accumulated in the BOS-Graph while running the real-world mission. (Spot got stuck between step 85 and 130, leading to a flat section.)

In Figure 5-7 the progress of carrying out the mission and constructing the BOS-Graph is illustrated. Some notable moments include the start in Figure 5-7a. The robot exploring around the bar in the environment and finding and assessing the first victim shown in Figure 5-7c to Figure 5-7d. Finding the second victim in the rear hallway in Figure 5-7e. The robot then got stuck at step 88 in Figure 5-7g trying to relocalize to a node after failing to visit a frontier. At step 132 in Figure 5-7i the agent was able to free itself again, successfully relocalizing. Subsequently, in Figure 5-7k the robot has completely explored the narrow tunnel, finding and helping victim 3 and 4. Finally, in step 216 the mission is completed as shown in Figure 5-7l, of which an enlarged illustration is provided in Figure 5-8.



(a) step 0. **(b)** step 11. **(c)** step 22.

(d) step 30. **(e)** step 44. **(f)** step 66.

(g) step 88. **(h)** step 110. **(i)** step 132.

(j) step 154. **(k)** step 191. **(l)** step 216.

Figure 5-7: The progression of building a BOS-Graph in the real-world scenario. The agent is the blue pyramid. Green nodes are frontiers. The red line towards the green arrow is the plan. Identified victims are pink.

Comparison with AutoWalk GraphNav map The BOS-Graph resulting from the experiment shown in Figure 5-8 is compared with the AutoWalk graph recorded on the Spot robot as shown in Figure 5-9. The first thing to note in Figure 5-9 is that the recorded graph does not exactly match the floor plan. This mismatch means that the recorded graph is not metrically consistent, but this is not a problem for AutoWalk navigation on Spot[3].

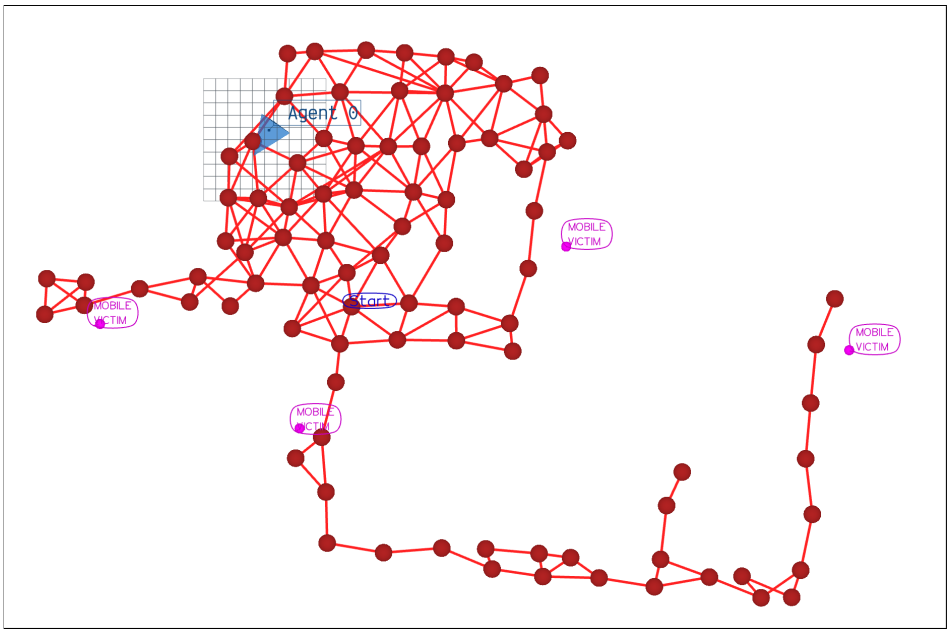


Figure 5-8: The final BOS-Graph constructed from a real world mission. Note the identified victims in pink, the agent in blue (grid is its sensor range), the red nodes are waypoints.

However, the part that is interesting for our work is that the presented AutoWalk GraphNav is just a navigation graph. This means it is not actionable at all. If we compare this with the BOS-Graph as shown in Figure 5-8, we see that there are also actionable situations composed of victims that need to be assessed using the assess behavior. The assess behavior is represented in the edges connecting the victim situation node with its waypoint nodes. In the figure, these assess behaviors have been performed already, so the edges have been removed.

[3]Because sensor data is also stored at each node the robot is still able to effectively localize to and navigate using this metrically inconsistent map by taking it one step at a time [71].

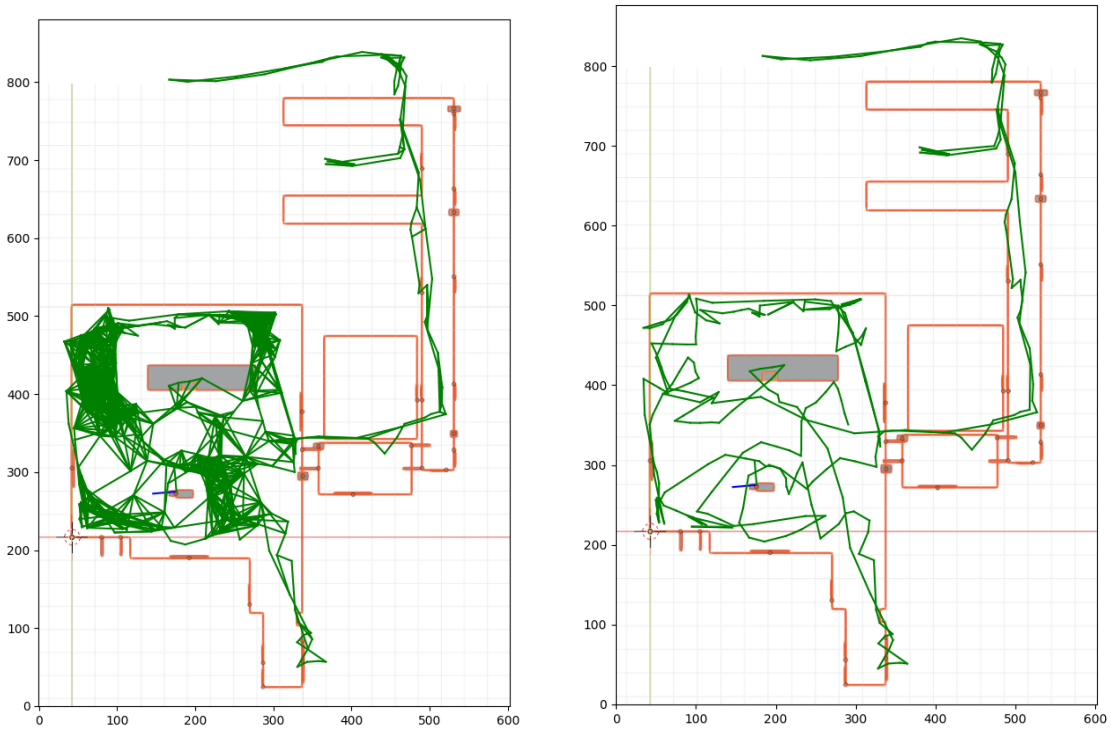


Figure 5-9: The recorded AutoWalk GraphNav map during execution of our autonomous pipeline. Left with loop closures and right without. Put very simply, loop closures are when additional edges are inferred between nearby nodes.

Discussion The proposed pipeline gives the robot a clear sense of goal autonomy. The simulation setup is successfully transferred to the real robot. Without prior knowledge of the geometry of the environment, the robot can autonomously discover tasks in its environment and make plans based on all the behaviors that are available in its environment representation, the BOS-Graph. Fiducials are used as placeholders for computer vision methods for situation or object detectors. They also facilitate object tracking, as the markers are uniquely identifiable. In a real setup, we would require additional object-tracking methods. Additionally, exploration performance could improve if we had a longer sensor range. This would allow more local coverage planning, which can improve efficiency in terms of lower time to complete the mission.

When we compared our method with the AutoWalk graph of Boston Dynamics, several benefits of our method become apparent, especially for an industrial inspection use case rather than a SAR use case. The main bottleneck to creating an AutoWalk mission is that we require a pilot to teleoperate the robot. Rather than literally specifying the sequence of actions as a line graph, instead, we autonomously obtain the complex graph of possible behaviors based on situation detections and the affordance database. This eliminates the need for a human pilot. Every possible AutoWalk mission in the environment can now be represented as a path over the BOS-Graph. Due to the naive cost-to-go frontier-based exploration implementation with limited sensor range, the BOS-Graph generation has several limitations. With a larger sensor range and a more advanced exploration method, information could be gathered more

efficiently, resulting in quicker exploration.

5-5 Concluding remarks

This chapter provides the synthesis of Chapter 3 and Chapter 4, which we extend with exploration. This answers the third research question: **How can we discover possible tasks and behaviors online in a real-world unseen environment on the Spot robot?**

First, a reference implementation of exploration was motivated and described. Using Spot's local terrain map, we sample frontiers in the robots' direct surroundings. These frontiers are validated and aggregated using the global BOS-Graph. Second, all pieces of the pipeline were combined to perform task discovery, which allows us to construct the BOS-Graph online and perform our experiments.

Subsequently, the pipeline to perform task discovery to realize goal autonomy was extensively experimentally evaluated. The experiments were carried out in a real-world SAR scenario with the physical Spot Robot. This real-world scenario shows that the method can successfully transfer from the simulation to the real world, given our assumptions about the perception system. In performing these experiments we have demonstrated not only our conceptual idea for a novel environment representation but also the real-world practical pipeline to construct it online.

From the experiments, it is concluded that the level of SA during the mission is raised to within the range of level 2: comprehension of the environment. This improvement is not only for the rescue worker, who has clear insight into the status of the SAR mission progress and the situations that require action and which behaviors are useful in those situations, but also for the mobile robot. Now the robot maintains a representation containing all tasks spatially grounded in the environment as well as the behaviors it shall and can perform for each task. Our novel representation enables the straightforward comparison of tasks.

Additionally, if an agent loses certain capabilities during its mission, for example, a manipulator defect occurs, it is directly understood in terms of the BOS-Graph. Through the filtering step, manipulator-related tasks are intuitively removed.

Finally, we also compare our method to the AutoWalk authoring solution provided out-of-the-box by Boston Dynamics. We propose that our framework will also generalize outside of the SAR domain because we decomposed the authoring and combine it with exploration and planning. Applications include performing industrial inspections in dynamic or unknown environments, such as construction sites, factories, or disaster scenarios.

Chapter 6

Conclusions

This conclusive chapter provides a summary of the content presented in this work and the answers to the initial research questions. In addition, several recommendations are presented for future work and current problems to be resolved.

6-1 Summary

In this work, we have shown how the perspective of situational advantages is used to create the Behavior-Oriented Situational Graph (BOS-Graph) that aids Search-and-Rescue (SAR) missions.

In **Chapter 1** we provided a general introduction to using mobile robots for SAR, presented the research questions, and gave an outline of the thesis.

In **Chapter 2** we presented background on situational awareness, spatial scene graphs, affordances, authoring and planning in robotics. This provided further context to our research gap.

In **Chapter 3** we described the information pipeline to construct BOS-Graphs from Scene Graphs. This involved the specification of the prior knowledge components: affordances, objectives, behaviors, and situations. Our qualitative analysis showed how we can realize a higher level of Situational Awareness (SA).

In **Chapter 4** we integrated the output of Chapter 3, the BOS-Graph, with a reference implementation of a planner. The planner is capable of assigning tasks efficiently, allowing it to be run at each step. We showed that, on average, online task allocation is beneficial in dynamic environments, resulting in fewer steps required in general.

In **Chapter 5** we extend the methods of Chapter 3 and Chapter 4 with exploration to enable online task discovery. This combination is extensively experimentally validated and verified on the Spot robot in the real world, showing that it works effectively.

6-2 The answers to the research questions

Assuming that we can robustly perceive situations, how can we best structure prior knowledge to create an information pipeline to construct more actionable environment representations?
A pipeline that can discover tasks in an environment by detecting situations can rapidly acquire situational awareness in a Search-and-Rescue scenario. Relevant behaviors are aggregated as edges and situations as nodes in a graph structure. The resulting representation provides a comprehension of the environment, in the sense that the current space of next possible actions has been made explicit. This makes the representation more actionable in terms of SAR, as defined by [38]. A prime candidate for introducing this mapping is the novel perspective of situational affordances. By structuring our prior knowledge in this format, it provides a unified way to link situations with behaviors with mission objectives.

How can we exploit actionable environment representations to make robot task selection and planning more light weight?
By mapping the space of possible behaviors, we severely limit the search space for planning. Based on the strong assumption on the domain made in this work, it becomes possible to employ basic graph search methods to find plans very effectively. Although planning was not the focus, the reference planner implemented in this work was able to brute-force evaluate all possible tasks based on their plans at each time step. This allowed for online task re-allocation, even for large BOS-Graph with several hundred nodes and edges.

How can we discover possible tasks and behaviors online in a real-world unseen environment on the Spot robot?
By providing a reference implementation of a basic exploration behavior, the online task discovery and online construction of the BOS-Graph on the physical Spot robot were realized. This allowed us to extensively validate the proposed pipeline in a real-world SAR scenario, proving that it is an effective framework to specify SAR missions. The proposed framework based on situational affordances provides the robot with a sense of goal autonomy.

6-3 Future challenges and recommendations

This section aims to give an overview of the fundamental limitations of the current method, as well as challenges towards deployment. Furthermore, we present recommendations for improvements considering these limitations. Finally some ideas for future extensions are discussed.

6-3-1 Recommendations for further research

This subsection details the fundamental limitations of the proposed work and recommendations to mitigate these challenges, as well as ideas for future research.

Integration with situational detection and classification systems The main future challenge would be to address the core assumptions made regarding the availability of the situation detection and classification-capable perception system. The fiducial-based perception system needs to be replaced with an actual scene graph engine and situation processing pipeline.

Include probabilistic components Perception systems and detectors tend not to be binary, as we have assumed in this work, they are usually probabilistic. These probabilities of detection could be used to weigh the rewards in the task allocation of the planner; this would allow the end user to give preference for tasks associated with very certainly detected tasks, even if they have a lower base reward.

Obtaining situational affordances at a scale One of the main bottlenecks of the current method is the manual specification of situational affordances. Various recent studies have explored the use of data-driven learning methods to obtain traditional affordances at scale [48, 61], both at the object and at the pixel level. To scale the proposed method to a wider variety of situations, it is imperative that we also scale the methods to obtain situational affordances. The main challenge here is to obtain the training data. One method to obtain this would be through randomized simulation. Given an objective and robust behaviors that stand to accomplish those objectives, we generate controlled randomized situations and measure whether applying the behavior results in progress towards the task.

Generalization to other domains The proposed pipeline is useful for deployment domains other than SAR, it would be interesting to investigate deployments there. Potential domains are automated inspection in dynamic environments such as construction sites.

6-3-2 Recommendations towards deployment

This section details the practical limitations of the work, that were kept out of scope in this thesis, but are important to move towards deploying the work in the real world to help real people. Possible solutions are presented, along with ideas for potential extensions.

Expand the BOS-Graph to 3D One of the main benefits of the proposed method is its scalability. Because we use a graph instead of cells or voxels, searching over the representation remains computationally tractable for additional dimensions. Especially for domestic SAR missions, it is important that the agent can go up and down the stairs in a home.

Multi-agent task allocation algorithm Although our method allows deployment of multiple agents, we did not consider multi-agent task allocation. The resulting naive task assignment is far from optimal. However, it would be straightforward to expand the algorithm with modern multi-agent task allocation methods [68, 69]. A multi-agent task allocation problem would take the form as described in Equation 6-1. Given a BOS-Graph $\mathcal{G}$, a set of tasks $\mathcal{T}$ and a set of agents $\mathcal{A}$, find the set of plans Π^* corresponding to the optimal assignment of each task τ to each agent a in $\mathcal{A}$.

$$\mathbb{T}_{\text{multi-agent}} : \{\mathcal{G}, \mathcal{T}, \mathcal{A}\} \rightarrow \Pi^*_{\mathcal{A},\mathcal{T}} \tag{6-1}$$

This work already provides the functionality to filter the graph to the capabilities of a specific agent. So, it would be interesting to extend it to multi-agent task allocation.

More advanced graph search Considering the current reference planner, caching could be used to speed up task allocation. Depending on the number of agents and tasks, it might be beneficial to calculate paths from each task to each node and store those, rather than calculating paths from each agent at each time step. Recent work on the one-to-many pathfinding problem provides insight into when which method should be used [72]. Additionally, algorithms such as D already extend A by repairing paths if the environment changes rather than completely replanning [73].

More sophisticated exploration behavior implementation If we equip Spot with longer-range sensors, we can obtain a larger local grid. This would allow us to sample frontiers more effectively using information-gain exploration methods such as next-best-view. An additional benefit would be that we could start applying local coverage planners in the exploration behavior, further improving efficiency.

Provide interface for operator In the pipeline presented, we assume zero prior knowledge of the spatial layout of the deployment environment. In real-world scenarios, certain areas may have semantic meaning, which makes them more likely to contain search targets. An example could be bedrooms in a night-time SAR scenario in a domestic setting. Therefore, it could be beneficial for an operator to add hotspot nodes to the graph before deployment. This could bias exploration towards those hotspot nodes.

Appendix A

Bounding the plan domain

Let us be precise about what types of plans the proposed method can generate. We extend this to formalizing what kinds of planning problems can be solved and what kinds of planning problems cannot be solved. This is important because the proposed method works only for problems that can be solved by plans following this topology.

Definition A-0.1 (Plan domain)**.** *Its defined as which sequences of types of behaviors are possible in a plan.*

The simplest but also the most common plan domain is that of "go from 1 to 2 and do 3". Exploration also falls within this plan domain as it is essentially going to the next frontier and performing some data processing steps to add new frontiers.

Or, more formally, "do b_1 from v_1 to v_2 and do b_2 from v_2 targeted at v_3" type plans. The plan domain is defined as the tuple of the sets of allowed behaviors between v_1 and v_2 and of allowed behaviors $\mathcal{B}$ between v_2 and v_3. The plan domain will be further formalized in the next paragraph.

Formalisation of Plan Domain Given a plan with the following sequence of edges:

$$\pi = \langle e_1, ..., e_n, e_j \rangle \tag{A-1}$$

Then the plan domain is defined as the sequence of sets of behaviors which can be mapped to the sequences of edges of the plan. So, for example:

$$\begin{aligned} &D_{\text{plandomain}} := \langle \mathcal{B}_1, \mathcal{B}_2 \rangle \\ &s.t. \quad e_1.b, ..., e_n.b \in \mathcal{B}_1 \quad \forall e_i \in \pi, \quad i = 1, ..., n \\ &s.t. \quad e_j.b \in \mathcal{B}_2 \quad \forall e_j \in \pi \end{aligned} \tag{A-2}$$

$$\forall \tau \exists b \tag{A-3}$$

For the proposed method the plan domain can then be defined using the set of objectiveless affordances $\mathcal{H}_{\text{supportive}}$ and the set of affordances with objectives $\mathcal{H}_{\text{objective}}$ in Equation A-4.

$$D_{\text{plandomain}}(\text{BOS-Graph}) = \langle \{b : b \in \mathcal{H}_{\text{supportive}}.b\}, \{b : b \in \mathcal{H}_{\text{objective}}.b\} \rangle \tag{A-4}$$

Bibliography

[1] P. Ardón, È. Pairet, K. S. Lohan, S. Ramamoorthy, and R. P. A. Petrick, "Affordances in Robotic Tasks – A Survey," *arXiv:2004.07400 [cs]*, Apr. 2020.

[2] "Spot® - The Agile Mobile Robot," https://www.bostondynamics.com/products/spot.

[3] P. Skoglund, E. Ersmark, E. Palkopoulou, and L. Dalén, "Ancient Wolf Genome Reveals an Early Divergence of Domestic Dog Ancestors and Admixture into High-Latitude Breeds," *Current Biology*, vol. 25, no. 11, pp. 1515–1519, Jun. 2015.

[4] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative Multi-Robot Systems for Search and Rescue: Coordination and Perception," *arXiv:2008.12610 [cs]*, Aug. 2020.

[5] G. D. Cubber, D. Doroftei, K. Rudin, K. Berns, A. Matos, D. Serrano, J. Sanchez, S. Govindaraj, J. Bedkowski, R. Roda, E. Silva, and S. Ourevitch, *Introduction to the Use of Robotic Tools for Search and Rescue*. IntechOpen, Aug. 2017.

[6] P. Narayanan, B. Yeh, E. Holmes, S. Martucci, K. Schmeckpeper, C. Mertz, P. Osteen, and M. Wigness, "An integrated perception pipeline for robot mission execution in unstructured environments," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, T. Pham, L. Solomon, and K. Rainey, Eds. Online Only, United States: SPIE, Apr. 2020, p. 54.

[7] "DARPA Subterranean (SubT) Challenge," https://www.darpa.mil/program/darpa-subterranean-challenge.

[8] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 335–346, Jun. 1990.

[9] M. Taddeo and L. Floridi, "Solving the symbol grounding problem: A critical review of fifteen years of research," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 17, no. 4, pp. 419–445, Dec. 2005.

[10] M. Kamermans and T. Schmits, "The History of the Frame Problem," p. 52.

[11] "ANYmal - Autonomous Legged Robot."

[12] "Agility Robotics," https://agilityrobotics.com.

[13] A. Akbari and S. Bernardini, "Informed Autonomous Exploration of Subterranean Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7957–7964, Oct. 2021.

[14] A. Agha, K. Otsu, B. Morrell, D. Fan, R. Thakker, A. Santamaria, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, M. Ginting, K. Ebadi, M. Anderson, T. Pailevanian, E. Terry, M. Wolf, A. Tagliabue, T. Vaquero, M. Palieri, and J. Burdick, *NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge*, Mar. 2021.

[15] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based Subterranean Exploration Path Planning using Aerial and Legged Robots," *Journal of Field Robotics*, Oct. 2020.

[16] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion," *arXiv:2010.09259 [cs]*, Nov. 2020.

[17] TeamCoSTAR, "Autonomous Spot: Long-Range Exploration of Extreme Environments (IROS 2020 Talk)," Oct. 2020.

[18] "Autonomous Resilient Exploration in Subterranean Environments and other High-Risk Settings," https://www.autonomousrobotslab.com/subtplanning.html.

[19] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," Oct. 2019.

[20] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "Scene Graphs: A Survey of Generations and Applications," *arXiv:2104.01111 [cs]*, Mar. 2021.

[21] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," *arXiv:1910.02527 [cs]*, Oct. 2019.

[22] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans," *arXiv:2002.06289 [cs]*, Jun. 2020.

[23] R. E. de Vries, J. M. Tybur, T. V. Pollet, and M. van Vugt, "Evolution, situational affordances, and the HEXACO model of personality," *Evolution and Human Behavior*, vol. 37, no. 5, pp. 407–421, Sep. 2016.

[24] F. Chen, J. Wang, T. Shan, and B. Englot, "Autonomous Exploration Under Uncertainty via Graph Convolutional Networks," p. 16.

[25] C. Stachniss, *Robotic Mapping and Exploration*, Jan. 2009, vol. 55.

[26] I. Lluvia, E. Lazkano, and A. Ansuategi, "Active Mapping and Robot Exploration: A Survey," *Sensors*, vol. 21, no. 7, p. 2445, Apr. 2021.

[27] C. Gomez, A. C. Hernandez, and R. Barber, "Topological Frontier-Based Exploration and Map-Building Using Semantic Information," *Sensors*, vol. 19, no. 20, p. 4595, Oct. 2019.

[28] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object Goal Navigation using Goal-Oriented Semantic Exploration," *arXiv:2007.00643 [cs]*, Jul. 2020.

[29] R. Ashour, T. Taha, J. M. M. Dias, L. Seneviratne, and N. Almoosa, "Exploration for Object Mapping Guided by Environmental Semantics using UAVs," *Remote Sensing*, vol. 12, no. 5, p. 891, Mar. 2020.

[30] C. Wang, D. Zhu, T. Li, M. Q.-H. Meng, and C. W. de Silva, "Efficient Autonomous Robotic Exploration With Semantic Road Map in Indoor Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2989–2996, Jul. 2019.

[31] M. Everett, J. Miller, and J. P. How, "Planning Beyond the Sensing Horizon Using a Learned Context," *arXiv:1908.09171 [cs]*, Jun. 2020.

[32] M. Wang, R. Luo, A. Ö. Önol, and T. Padir, "Affordance-Based Mobile Robot Navigation Among Movable Obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 2734–2740.

[33] T. Gunn and J. Anderson, "Dynamic heterogeneous team formation for robotic urban search and rescue," *Journal of Computer and System Sciences*, vol. 81, no. 3, pp. 553–567, May 2015.

[34] S. Amiri, K. Chandan, and S. Zhang, "Reasoning With Scene Graphs for Robot Planning Under Partial Observability," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5560–5567, Apr. 2022.

[35] C. Galindo and A. Saffiotti, "Inferring robot goals from violations of semantic knowledge," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1131–1143, Oct. 2013.

[36] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti, "TASKOGRAPHY: Evaluating robot task planning over large 3D scene graphs," *ok*, p. 20.

[37] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational Graphs for Robot Navigation in Structured Indoor Environments," *arXiv:2202.12197 [cs]*, Feb. 2022.

[38] M. Endsley, "Endsley, M.R.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal 37(1), 32-64," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, pp. 32–64, Mar. 1995.

[39] ——, "Situation awareness analysis and measurement, chapter theoretical underpinnings of situation awareness," *A Critical Review*, pp. 3–33, Jan. 2000.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[41] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "VQA: Visual Question Answering," Oct. 2016.

[42] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural Motifs: Scene Graph Parsing with Global Context," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 5831–5840.

[43] D. Rosen, K. Doherty, A. Espinoza, and J. Leonard, "Advances in Inference and Representation for Simultaneous Localization and Mapping," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, May 2021.

[44] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans," in *Robotics: Science and Systems XVI.* Robotics: Science and Systems Foundation, Jul. 2020.

[45] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, *Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs*, Jan. 2021.

[46] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A Real-time Spatial Perception Engine for 3D Scene Graph Construction and Optimization," *arXiv:2201.13360 [cs]*, Jan. 2022.

[47] J. J. Gibson, "The theory of affordances," *Hilldale, USA*, vol. 1, no. 2, pp. 67–82, 1977.

[48] P. Ardón, È. Pairet, K. S. Lohan, S. Ramamoorthy, and R. P. A. Petrick, "Affordances in Robotic Tasks – A Survey," *arXiv:2004.07400 [cs]*, Apr. 2020.

[49] E. Senft, M. Hagenow, K. Welsh, R. Radwin, M. Zinn, M. Gleicher, and B. Mutlu, "Task-Level Authoring for Remote Robot Teleoperation," *Frontiers in Robotics and AI*, vol. 8, 2021.

[50] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager, "CoSTAR: Instructing collaborative robots with behavior trees and vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 564–571.

[51] "Py Trees," splintered-reality, Aug. 2022.

[52] "BehaviorTree/BehaviorTree.CPP: Behavior Trees Library in C++. Batteries included." https://github.com/BehaviorTree/BehaviorTree.CPP.

[53] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of Behavior Trees in robotics and AI," *Robotics and Autonomous Systems*, vol. 154, p. 104096, Aug. 2022.

[54] "Editing Autowalk missions," https://support.bostondynamics.com/s/article/Editing-Autowalk-missions.

[55] C. Galindo, J.-A. Fernandez-Madrigal, and J. Gonzalez, "Improving efficiency in mobile robot task planning through world abstraction," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 677–690, Aug. 2004.

[56] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated Task and Motion Planning," *arXiv:2010.01083 [cs]*, Oct. 2020.

[57] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning," *arXiv:1802.08705 [cs]*, Mar. 2020.

[58] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 1470–1477.

[59] L. Steenstra, "PDDL-Based Task Planning of Survey Missions for Autonomous Underwater Vehicles: A generic planning system, taking into account location uncertainty and environmental properties," 2019.

[60] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, second edition ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press, 2018.

[61] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," p. 34.

[62] A. Schoen, C. Henrichs, M. Strohkirch, and B. Mutlu, "Authr: A Task Authoring Environment for Human-Robot Teams," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. Virtual Event USA: ACM, Oct. 2020, pp. 1194–1208.

[63] M. Colledanchise and P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," *arXiv:1709.00084 [cs]*, Jul. 2018.

[64] "Door Manipulation with the Spot Arm," https://support.bostondynamics.com/s/article/Door-manipulation-with-the-Spot-Arm.

[65] G. Sepulveda, J. C. Niebles, and A. Soto, "A Deep Learning Based Behavioral Approach to Indoor Autonomous Navigation," Mar. 2018.

[66] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 6541–6548.

[67] "Creating robust Autowalk missions," https://support.bostondynamics.com/s/article/Creating-a-robust-Autowalk-mission-framework.

[68] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Studies in Computational Intelligence*, May 2015, vol. 604, pp. 31–51.

[69] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, Oct. 2013.

[70] P. Quin, D. D. K. Nguyen, T. L. Vu, A. Alempijevic, and G. Paul, "Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration," *Frontiers in Robotics and AI*, vol. 8, 2021.

[71] "GraphNav Map Structure — Spot 3.2.0 documentation," https://dev.bostondynamics.com/docs/concepts/autonomy/graphnav_map_structure.

[72] R. Stern, M. Goldenberg, A. Saffidine, and A. Felner, "Heuristic search for one-to-many shortest path queries," *Annals of Mathematics and Artificial Intelligence*, vol. 89, no. 12, pp. 1175–1214, Dec. 2021.

[73] S. Koenig and M. Likhachev, "D*lite," in *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, Jul. 2002, pp. 476–483.