Active Inference for Graph Exploration and Searching in Unknown Environments

An Application to Mobile Robots

Willem Meijers

Master of Science Thesis

Delft Center for Systems and Control

Active Inference for Graph Exploration and Searching in Unknown Environments

An Application to Mobile Robots

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Willem Meijers

June 12, 2022

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

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

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Active Inference for Graph
Exploration and Searching in
Unknown Environments

by

Willem Meijers

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: June 12, 2022

Supervisor(s):

Dr. Ir. J. Sijs

Ir. C. Pezzato

Reader(s):

Prof. Dr. Ir. M. Wisse

Prof. Dr. Ir. B.H.K. de Schutter

Abstract

The autonomy of mobile robots has been greatly improved in recent decades. For these robots, the field of search and rescue is of particular interest. This thesis introduces a new method to let a mobile robot (Spot by Boston Dynamics) explore and search for victims in unknown environments. Existing methods include coverage, which aims to fully cover the environment as efficiently as possible. Exploration methods place more emphasis on gaining knowledge of the environment quickly, but do not actively search for victims. A new method based on active inference is introduced with the aim of combining exploration and exploitation behaviour within one framework. The active inference model is based on a graph representation of the environment, formulated as a POMDP. The framework is built up by incrementally more difficult cases. The first case allows a mobile robot to navigate a known graph to search for victims. The second case assumes a partially unknown graph. Uncertainty about the existence of unvisited nodes is included in the predictions. The final case adjusts the model to the agent's point-cloud and camera sensors. The framework is then used in a simulation environment, showing how it can be implemented in real-world scenarios. To do so, the active inference framework is combined with techniques from information gain exploration. This thesis shows that active inference can be used in large unknown environments to carry out search and exploration.

Glossary

List of Acronyms

BCD	Boustrophedon Cellular Decomposition
EFE	Expected Free Energy
KL	Kullback-Leibler
POMDP	Partially Observable Markov Decision Process
SIG	State Information Gain
SLAM	Simultaneous Localisation and Mapping
TSP	Traveling Salesman Problem
UT	Utility

Table of Contents

List of Figures

List of Tables

Acknowledgements

First I would like to thank my supervisors Joris Sijs and Corrado Pezzato. You have helped to raise the level of this thesis to a higher level. I always felt that I had the freedom to pursue my own ideas. I greatly appreciate the time you both took to help. I had a lot of fun working out new concepts and discussing possible techniques with you during the last year.

Then, I would like to thank my friends. This thesis marks the end of my time as a student. All the adventures we had and the great memories we made truly make me look back with a huge smile. That we may have many more in the future!

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.

Delft, University of Technology — Willem Meijers
June 12, 2022

“We shall not cease from exploration, and the end of all our exploring will be to arrive where we started and know the place for the first time.”

— T.S. Eliot

Chapter 1

Introduction

This introduction chapter starts with a historical overview on autonomous robots. Next, the Snow Project by TNO is introduced, from which the main research motivation will be derived. After this, some state-of-the-art techniques on coverage and exploration are given, as well as a brief introduction to active inference. From these findings, the research questions are formulated. The chapter closes with an overview of the further content of this report.

1-1 Autonomous Robots, From Fiction to Reality

On the 25th of January 1921, just over 100 years ago, the play Rossumovi Univerzální Roboti (R.U.R.), by the Zcech writer Karel Čapek, premiered [1]. The play was very influential at the time. The English version, called Rossum's Universal Robots, resulted in the first ever entry of the word "robot" into the English language. After this, the media depcited robots mainly as humanoid and intelligent machines, though merely existing in science-fiction novels and movies. In the following years, fiction slowly turned into reality with the introduction of real robots on assembly lines. Something very common today, illustrated by the vast number of robotic arms encountered in car factories. Assembly line robots, however, are fixed in place and are often designed for one specific task. To help or replace humans in other environments or industries, greater mobility of a robot was desired. In 1966 Stanford Research Institute (SRI) launched Shakey, which not only was one of the first mobile robots, but also had the ability to perceive and reason about its surroundings [2]. The researchers looked at new areas of study, such as planning and route finding. Life megazine called Shakey the first electronic person. Since then, great improvements have been made. Around the year 2000, MIT researchers used mobile robot to do coverage over fields, eventually leading to the algorithms used in today's robotic vacuum cleaners. In 2004 Boston Dynamics introduced BigDog, a four-legged mobile robot that, for its autonomy, is better suited to work in areas designed for humans. Lately, the field of self-driving cars has accounted for important improvements in the autonomy of systems, and especially the last 10 years the field of artificial intelligence has seen big changes by the take-off of deep learning and artificial neural networks. The most recent humanoid robots can, through the use of advanced control techniques, mimic the agility of humans to a large extent [3]. Perhaps one could say that the ‘roboti’ as introduced by Čapek, are no longer science fiction after all. However, there is still much to gain and learn about the operation of mobile robots in human environments. One of the fields where robotic support can be of great benefit is search and rescue missions. They are often conducted in dangerous and difficult-to-manoeuvre environments, while the stakes of quickly finding and assessing survivors are high. The Dutch research organisation TNO started a project that tackles modern challenges in autonomous systems. The project is called Snow, and the knowledge gained by this project is demonstrated in a search and rescue setting.

Project Snow

Snow is a TNO project that studies autonomous systems operating in the open world. It addresses research challenges in situational awareness, self-awareness, and situational planning. The agent used is the Spot robot from Boston Dynamics [4]. The task of Spot is to autonomously assist a fire brigade during a rescue mission. Spot is tasked with finding survivors in a villa. The robot has to locate family members and medically assess them. To do so, both the data collected by the agent and knowledge that can help conduct the mission more efficiently are used. Examples of knowledge are the villa map, what to expect inside, and targets and preferences given by the fire brigade.

Within this use case, one of Spot’s tasks is to thoroughly search small to medium-sized rooms. These rooms are represented as unknown, obstacle-cluttered environments. Preferably, the search can be adjusted to the operator’s desires. Sometimes a quick search is desired, while, at other moments, a thorough search is more suitable. The main challenge is to search intuitively, basically mimicking the decisions a real firefighter would make while searching such a room. Methods that greedily adjust the robot’s trajectory to enlarge the chance of finding survivors are therefor prefered. Now, the main motivation for this research can be defined. The aim is to find an answer to the following question:

How can we let Spot intuitively search an unknown room for victims?

1-2 State of the Art

To find an answer to the stated motivation, there are three research areas of interest. The first; **coverage**, is concerned with fully covering an area. This is often done in a time or energy efficient way. A clear example is field coverage with a ploughing machine or room coverage with a robotic vacuum cleaner. Another research area; **exploration** is less concerned with complete coverage, yet tries to explore areas as quickly as possible. Exploration is generally of better use for search and rescue, as time is of the essence and the environment is often (partially) unknown. The third area adds exploiting behaviour. There is a clear goal of finding the victim. Techniques that reward behaviour, which directly increase the chances of doing so, are preferred. In this research, a new method to include this exploiting behaviour will be reviewed. The method used is based on **active inference**. In this section, a state-of-the-art overview of all three fields is given.

Coverage

The mobile robot coverage problem is often encountered in forestry, agriculture, and cleaning applications. The problem is given by an environment in which a path needs to be found such that every accessible area within the environment has been visited at least once. This problem is known as the Traveling Salesman Problem (TSP). The TSP is NP-hard and therefore not solvable in polynomial time. One way to solve the TSP is to find the spanning tree of the graph. The spanning tree is a tree that contains every node in the original graph. Gabriely and Rimon [5] showed how such a spanning tree can be used for both offline and online coverage. Graham and Hell [6] described multiple algorithms to solve spanning-tree problems. For the TSP a graph representation that represents the environment must be found. This graph has to ensure full coverage by its nodes. Research in other fields, such as wireless network coverage, describes how full coverage graphs can be designed. Eledlebi et al. [7] found a method to create and adjust a graph online, so that it covers all parts of the environment. Other methods for online coverage have been introduced by the use of reinforcement learning, both for static [8, 9] and dynamic [10] environments. In addition to graph representations, there are also methods for grid-world representations. Boustrophedon coverage is a popular method; it is based on a repeating pattern of back-and-forth motions through the environment. Choset et al. [11] introduced the Boustrophedon Cellular Decomposition (BCD). This method divides an environment into smaller areas that can be fully covered by boustrophedon motions. Viet et al. [12] build on this method by introducing a method called BA^*. It combines BCD with the popular A^* search algorithm of Hart et al. [13]. This combination makes that BCD can be executed in unknown environments, online. Once a Boustrophedon area is completed, the agent uses A^* to navigate to the closest undiscovered area. Coverage is a good solution if time is not of the essence. However, in search and rescue, the goal is often to find victims in the shortest time. Therefore, another set of techniques is more interesting.

Exploration

Exploration aims to cover new areas and gathers the most information from unknown spaces. It often aims to create a model of the environment faster and is less concerned with full coverage. Many exploration methods are combined with Simultaneous Localisation and Mapping (SLAM), as they are generally carried out in unknown environments. Spot already uses an internal localisation method. Therefore, localisation is not taken into account in this research. One set of methods are the frontier-based methods. Frontiers are locations on the edges of known space and form the boundary between explored and unexplored areas. A simple exploration method was introduced by Yamauchi et al. [14]. In their method, the agent simply moves to the closest frontier possible. Improvements can be made using entropy-based methods. These methods select the best frontier based on how much new information can be gathered. Thrun et al. [15] introduced a method in which the environment is represented by grid cells. They steer the agent to the frontiers from which the most new grid cells can be covered. González-Baños and Latombe [16] used a similar method, but weighed the information gain against the cost of getting to a frontier. This is done, for example, by adjusting for the time needed to reach a frontier. These methods often take longer to fully cover an area compared to coverage. However, the chance of finding a victim quickly increases as it covers larger areas in the early stages. In search and rescue, there are often clues as to where to expect a victim in the room. This kind of knowledge is not taken into account in methods that are aimed at exploration. Reinforcement learning is often used with autonomous robots. Within reinforcement learning, reward functions can be adjusted to obtain the desired behaviour [17]. However, for this research, active inference will be used to include exploiting behaviour.

Active Inference

Active inference comes from a neuroscientific model on the workings of the brain [18]. In recent years, it has been applied to numerous applications such as: decision making under uncertainty [19], control for the mountain-car benchmark problem [20], control for robotic manipulators [21], the exploration-exploitation dillema [22] and many more [23]. The fundamental principle of the technique is the minimisation of a quantity called the Expected Free Energy (EFE) [24], which can take multiple forms. Within this report, it is given by:

$$\text{EFE} = \text{State Information Gain} + \text{Utility} \tag{1-1}$$

The EFE consists of two segments. One segment aims to minimise uncertainty over states (state information gain). The other rewards certain states or observations (utility). This combination of both information gain (exploration) and the possibility to reward certain observations or states (exploitation) is promising for the search and rescue application. Active inference has been used in 'toy' examples like a mouse in a T-maze [25] and planning and navigation in small mazes [26, 27]. However, this is still on a significantly smaller scale than is needed for the desired application of search and rescue in unknown environments.

1-3 Research Questions

The main target in this research is to find out how Spot can explore an unknown area, search for victims, and exploit knowledge that can lead to the location of the victim. Based on the techniques noted in this introduction, the following research questions have been formulated:

- **How can Information Gain Exploration be Applied to Spot?**
  The aim of this question is to find out what capabilities the agent has. What sensors does it have and how can we use them to perform information gain exploration. The information seeking behaviour can then be used to compare or extend the active inference framework.

- **How can we use active inference for decision making during search and rescue missions?**
  The aim of this question is to gain fundamental knowledge on active inference. It attempts to show how this technique is implemented in practise. It tries to find a translation from existing active inference modelling techniques to graph representations of rooms or buildings. It also tries to show how the important states for search and rescue missions can be formulated.

- **How can we use active inference with unknown environments?**
  The aim of this question is to build on the answers to the previous question. It takes into account the uncertainty of unknown environments. How can a model be created for environments that are yet unknown, and how can we plan over areas we still have to explore?

- **(How) does active inference relate to information gain exploration?**
  This question aims to find the added value of active inference. Can we get the same information-seeking behaviour from active inference as from information gain exploration if we take the utility out of the question?

1-4 Report Lay-Out

Chapter 2 provides an overview of all the necessary formulations, techniques, and concepts used in this thesis. It discusses the agent and its sensors and introduces different world representations that can be used. Furthermore, it will show how entropy-based exploration can be applied to Spot to give some insight into the agent's capabilities and the fundamentals of exploration. This also introduces some techniques that will be useful later on. Chapter 3 is about the active inference framework. A theoretical background on active inference will be given. The framework is built up in incrementally more difficult cases. Every step contains one or multiple examples to give more insight and to confirm that the model leads to the desired behaviour from the agent. The first case shows how active inference can be used for navigation and searching over graphs. The second case adds uncertainty over the existence of nodes and the graph's structure. The final case takes Spot's camera and point-cloud sensors into account. Chapter 4 combines the two previous chapters and shows how the active inference framework can be implemented in real-world missions. It uses information on Spot's capabilities and some techniques from information gain exploration to create a graph representation of the world. Decision making on this graph will then be carried out by active inference. Chapter 5 concludes the report and summarises the previous chapters. It also answers the research questions. In the end, some recommendations are given for possible future work.

Chapter 2

Exploration with Spot

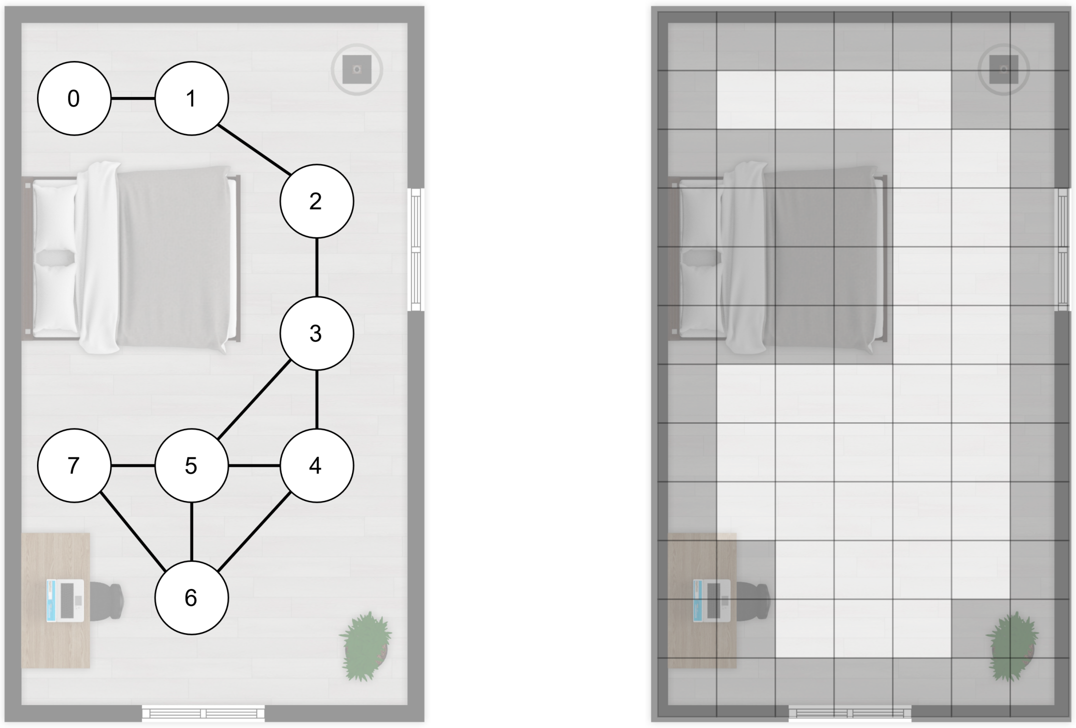
This chapter contains an overview on all the necessary background information for this thesis. The chapter starts with a description of the agent used for this research, the Spot robotic dog from Boston Dynamics. The second paragraph shows how environments can be represented both by grid and graph representations. Finally, entropy-based exploration methods are discussed, and it is shown how such methods could be implemented on Spot.

2-1 Representing the Environment

In this report, two different representations are used. The grid representation is used for obstacle avoidance, map building, node placement, and entropy calculations. The graph representation is used to navigate the world, represent world states, and to do decision making with active inference. Both representations are discussed in this section.

2-1-1 Grid Representation

The first representation is the occupancy grid or gridmap. The representation is given by a map of evenly sized squares (tiles). For simple occupancy grids, every tile has a binary value attached, representing free (0) or occupied (1) spaces. An example of such a grid is given in Figure 2-1b. Another possibility is the belief grid where every tile c can have a value $p(c) \in [0, 1]$. The variable $p(c)$ represents how strongly the agent thinks that a tile is free or occupied. The grid can be used for path planning with popular methods such as A^*. The agent is restricted to only move to adjecant, free tiles. Sometimes, additional tiles are mapped as occupied. This is to account for inaccuracies in the agent's dynamics or the map itself. Doing this increases the chance that a path in free space is feasible at all times. In this thesis 2.5D grid maps are used as well. In 2.5D maps, all tiles are labelled by terrain height. Labelling is done based on terrain map data obtained by Spot, but could also be generated, for use with other agents, from LiDaR sensors or stereo imagery [28]. Another alternative is the use of 3D representations where the tiles are replaced by voxels in 3D space.



(a) Graph representation **(b)** Grid representation

Figure 2-1: Different representations of a room.

2-1-2 Graph Representation

The other representation used is the graph. A graph $\mathcal{G}$ has a set $\mathcal{V} = \{1, \ldots, n_n\}$ of nodes (or vertices). The nodes can be connected to each other through a set of links $\mathcal{E}$. The nodes of the graph represent areas in the free space of the world. Links between nodes represent accessible (free) paths. Links can be weighted by the distance between nodes or the time it takes to traverse them, which is useful for finding the shortest path. A graph can be directed, in which case (some) links can only be traversed in one direction. If all links can be traversed in both directions, a graph is called undirected. In the remainder of this report, there will only be simple graphs. Simple graphs are graphs that are unweighted, undirected, and do not contain any self-loops (links node to itself). In this case, a graph can be fully described by the pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For the mission carried out, on even terrain, with an agile robot, the assumption that links can always be traversed in both directions is fair. In addition, new nodes will be placed on the edges of the agent's sensing radius. The radius is constant at every location. Therefore, the distances between the nodes are consistent enough to ignore weights. Figure 2-1a shows a graph representation of a room.

Besides the visual representation, a graph can be represented by its adjecancy matrix G_A. The adjecancy matrix represents the graph's structure by a matrix of booleans. Both columns and rows represent nodes in the graph. Their corresponding index is 0, if the two nodes are not connected, or 1 if they are connected. The adjecancy matrix corresponding to Figure 2-1a is given by:

$$G_A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{2-1}$$

A graph structure that will often be mentioned is the line graph. A line graph contains a node for each link in the graph and a link joining those nodes if two links share a common node. Figure 2-2 shows the layout of the linegraph.

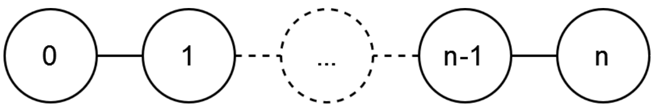


Figure 2-2: General structure of a line graph

The adjecancy matrix of a line graph consists of two, off-diagonal, identity matrices. It indicates that for every given node one can move one node further or one node back (except for the edges). The adjecancy matrix of a line graph can be represented by:

$$G_A^{n_n \times n_n} = \begin{bmatrix} \mathbf{0} & I \\ 0 & \mathbf{0}^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} & 0 \\ I & \mathbf{0}^T \end{bmatrix} \tag{2-2}$$

Note that in the graph shown in Figure 2-1a, nodes 0 to 3 represent a line graph. This can also be recognised by the off-diagonal entries in the corresponding adjecancy matrix. Most of the graphs used in this research are connected, meaning that from any node in the graph, any other node in the graph can be reached. However, for some sections, a dummy node will be added which is part of the graph but not connected. A dummy node is represented by an additional row and column of zeros in the adjecancy matric. It increases the graph's dimensions, while the additional node is not connected to any other nodes nor to itself. Its usefulness will become clear in Section 3-3. When dummy nodes are used, this is always clearly stated, and they are always represented by the last node in the graph (thus the final column and row in the adjecancy matrix).

In graphs, a path is a sequence of visited nodes, from an origin node i to a goal node j, in which no node can be visited twice. A path has a length l, which for a simple graph is equal to $n_v - 1$, with n_v the nodes visited along the path. The diameter of a graph is defined as:

$$\text{diam}(\mathcal{G}) = \max_{i,j} \text{dist}(i,j) \tag{2-3}$$

With $\text{dist}(i,j)$ being the length of the shortest path between nodes i and j. The diameter shows the minimum path length needed to go from any node to any other node in the graph. For planning, if the plan length is at least the diameter of the graph, every node in the graph can always be included in the plan.

Centralities

Graph centralities tell something about the importance of nodes in different aspects. In the remainder of the report, the different centralities are used in discussions and explanations. First, the degree centrality. To calculate the degree centrality, first the degree is introduced. The degree of a node is equal to the number of links connected to it. The vector representing all node degrees can be calculated using:

$$\mathbf{d} = G_A \mathbb{1} \tag{2-4}$$

Here $\mathbb{1}$ is the column vector of all ones. The degree centrality is the normalised degree of a node. The degree centrality of node x is given by:

$$C_d(x) = \frac{\mathbf{d}(x)}{||\mathbf{d}||_1} \tag{2-5}$$

The degree centrality indicates the importance of a node based on how many nodes it is linked to. It is an important characteristic because it tells how many other nodes can be directly reached, but also how many free directions there are to look in during the search mission. Another centrality is the betweenness centrality [29]. For node x it is given by:

$$C_b(x) = \sum_{x \neq i \neq j} \frac{\sigma_{ij}(x)}{\sigma_{ij}} \tag{2-6}$$

With i and j any other nodes in the graph. The variable σ_{ij} is the total number of shortest paths between i and j. The variable $\sigma_{ij}(x)$ is the number of shortest paths that run through node x. A node with a high betweenness centrality needs to be crossed relatively often to reach different parts of the graph. These are the nodes that the agent will visit, but are also the nodes that the victim is more likely to pass when moving. Another centrality is the closeness centrality [30]:

$$C_c(x) = \frac{n_n}{\sum_j \text{dist}(x,j)} \tag{2-7}$$

With n_n the number of nodes in the graph. It shows a relative score for any node based on how many nodes are 'nearby'. Creating the graph is done by using sensor inputs from the agent. The next section will explain what Spot as an agent can do, what sensors it can use, and how the graphs can be built.

2-2 About Spot

Figure 2-3: The used agent for this research, Spot by Boston Dynamics.

Spot is an agile mobile robot made by Boston Dynamics [4]. It is instrumental in rough terrain and smaller environments. Its mobility makes it an ideal agent to explore small to medium-sized rooms with obstacles inside. Spot has built-in stereo cameras that it uses to create a point cloud of its surroundings. It has 360° perception and built-in obstacle avoidance. Its sensors have a radius of roughly 2 metres. For project Snow, Spot is additionally equipped with pan-tilt stereo cameras on top and a microphone. For the Snow use case, the autonomy aspect of Spot is the main concern. Especially its obstacle avoidance, mapping and path planning capabilities. If Spot is operating in an unknown environment, it can build a map while operating. To do so, it does not make use of autonomous localisation and mapping (SLAM) techniques. During execution, a graph is created in which every visited location is represented by a node. Every node is linked to a snapshot containing data for that location. For the simulations in this report, an adjusted snapshot is used. The adjusted snapshot consists of the agent's coordinates, the obtained terrain map, and a set of scores, which will be explained in Section 3-4. The nodes are connected by links that describe how to go from waypoint to waypoint. In rough terrain, Spot also needs to know what kind of movement is required; for example, edges that contain stairs require different movements than flat terrain. When a node has been visited, it is possible to plan a movement to a new node, as long as it is accessible and lays within Spot's sensing range. This is an easy and straightforward way of exploring which is part of Spot's API. During missions, three representations are used. Two global representations: the graph (for decision making) and the global occupancy grid (for node placement). The terrain map is a local representation and is unique for every node in the graph. It is used to extend the occupancy map and as an indicator of where possible victims may be located. Together with the front-facing camera, it is one of the two sensor inputs used by Spot. Further explanations and examples of the various representations are given in Chapter 4.

Sensor Inputs

The exploration frameworks described in this report are based on two inputs from the agent. The 2.5D terrain map generated by the robot and the front-facing cameras. The terrain map is a map of 2×2 metres and consists of 128×128 tiles. Each tile represents the value of the highest point at the corresponding coordinate. The terrain map can give a first guess on possible victims in the agent's direct surroundings. The terrain map is not accurate enough to distinguish people from other objects in a room, but it can help to differentiate between areas of greater and lesser interest. Figure 2-4b shows a rendering of a room. The red square indicates the sensor radius of the agent. Figure 2-4a shows the corresponding terrain map.

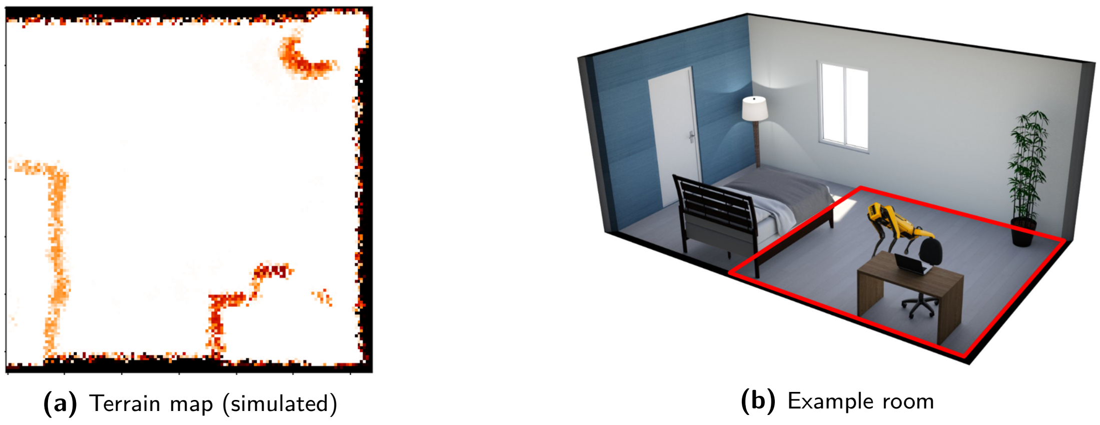

(a) Terrain map (simulated) **(b)** Example room

Figure 2-4: Example room with the corresponding terrain map.

Camera

Besides the terrain map, Spot uses front-facing cameras. These are used, in combination with image recognition software, to identify objects. For this research, the assumption is made that a victim will always be correctly identified as long as:

- The victim (at least one grid tile) is within the field of view of the agent. Which in this research is always within a 90 degree angle of the current orientation of the agent.
- The victim (at least one grid-tile) is within the sensing range of the agent. In this research this is always 2 metres.

Now that all the tools are present to create a map of the world, navigation and node placement are the last problems to solve. Basic frontier-based methods would explore the room by moving the agent to the closest unexplored frontier. A frontier is defined as any node that is unvisited and known. Unvisited means that the agent has not been there before. Known means that the agent is certain that that node will be part of the final graph. Candidate frontiers will be sampled on the edges of the agent's sensing range. The best candidate frontier can be selected on the basis of the distance or time needed to get there. Improvements can be made by greedily selecting the best frontier based on the information gain it delivers. This is the idea behind entropy-based exploration methods, which will be discussed in the next section.

2-3 Entropy Based Exploration

Entropy in information theory was introduced by C.E. Shannon in 1948 and is also referred to as Shannon entropy [31]. Entropy is the average level of uncertainty of a random variable. A random variable with low entropy has low uncertainty and variables with high entropy have high uncertainty. Mathematically, it is given by:

$$H(X) = -\sum_x p(x) \log p(x) \tag{2-8}$$

Here, X is the random variable and the sum denotes the sum over all possible values of X. There is also the relative entropy, also known as the Kullback-Leibler (KL) divergence. It contains the entropy of $p(x)$ relative to another distribution $q(x)$ and is a measure of how much two distributions differ. It is given by:

$$KL(p(x) \| q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{2-9}$$

The KL-divergence will be used multiple times later on. For now, let us continue with the entropy as given in Equation 2-8. In information gain exploration, the goal is to increase certainty about the environment. Using a grid representation, a tile (or cell) c can be occupied or empty. An empty tile would correspond to $p(c) = 0$ and an occupied tile to $p(c) = 1$. This makes that $p(c)$ is a distribution over the random variable c. With full knowledge that a tile is occupied or empty, the corresponding entropy would be zero. Anywhere in between gives an indication on how strong the belief is on the status of the tile. A tile with $p(c) = 0.5$ has maximum entropy and indicates that the chance that the tile is occupied or empty is equal. The goal is to minimise the entropy of the room by selecting the frontiers that do so the most. Making the assumption that tiles in a grid are independent of one another, the entropy of the environment is given by the summation of the entropy of every tile [32]. Since there are only two states: occupied $p(c)$ and free $(1 - p(c))$, the summation from Equation 2-8 can be removed. The total entropy of the map m is then given by:

$$H(m) = -\sum_{c \in m} p(c) \log p(c) + (1 - p(c)) \log(1 - p(c)) \tag{2-10}$$

The next challenge is to find the frontier that reduces the entropy of the map as much as possible.

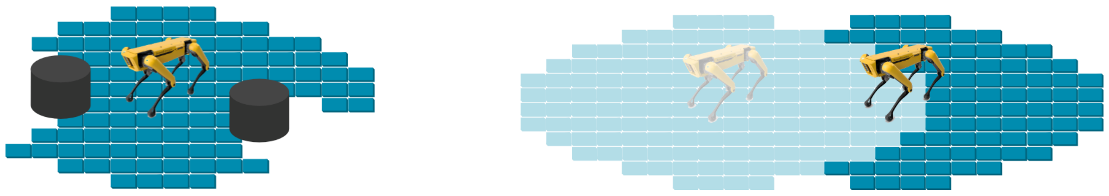
2-3-1 Information Gain Exploration with Spot

Now a method will be designed that uses previously stated theoretic background in a way compatible with Spot's requirements and sensors. The challenge is to find the best frontier out of a selection of candidate frontiers. For now, full certainty over observed tiles is assumed, meaning that $p(c)$ will always be 1 or 0 after the observation has taken place. This means that any observed tile will always have minimal entropy. Any unobserved tiles will have the same entropy at the start. This means that the entropy reduction is proportional to the number of unknown tiles covered by the agent's sensors from the candidate frontier. Later, in Chapter 4, trust values will be added to tiles that can be based on a sensor model. Then, $p(c)$ can take any value even after the observations have taken place. Figure 2-5 illustrates how the sensing radius of the agent is transformed to a representation by tiles.

(a) Spot's sensor radius **(b)** Sensor radius represented by grid

Figure 2-5: Sensor radius as grid

The difference in potential entropy reduction between candidate frontiers is given by tiles that cannot be observed, or that are already lower in entropy than the initial value. There are two reasons why this may be the case. The first is if the tiles are blocked by obstacles. Any tiles laying beyond the obstacle cannot be detected, and thus will not count to entropy reduction. For this, a best-case scenario is used. If the presence of an obstacle is not yet known, it is assumed that it is not there. Entropy reduction of tiles can also be lower, for example, because they have been observed by the agent before. Both situations are illustrated in Figure 2-6.



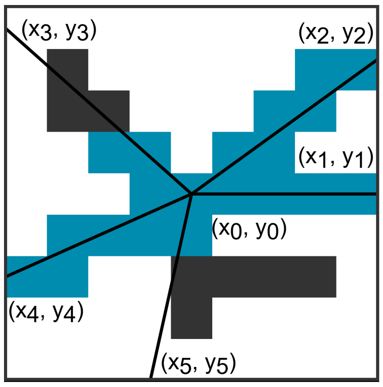
(a) View blocked by obstacles **(b)** Reduced entropy from earlier observation

Figure 2-6: Suboptimal entropy reduction

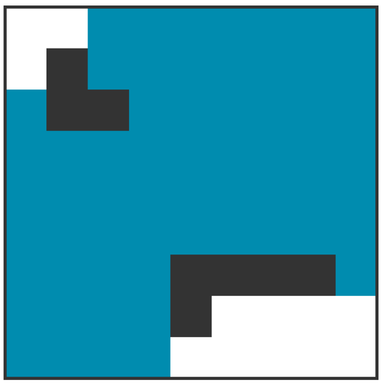
To predict how many nodes will be visible from a given candidate frontier, a technique called Ray-Tracing is used.

2-3-2 Ray-Tracing

The terrain map is represented by a grid world. Let us take the origin where the agent is located, and denote it by (x_0, y_0). To generate the field of view from the origin, Bresenhem's algorithm [33] can be used. It is a very simple and fast way to find the corresponding tiles of a line from (x_0, y_0) to (x_r, y_r) with the subscript r indicating any coordinates at the end of the sensing range. The algorithm will be repeated for multiple angles. The algorithm will break and continue to the next angle once an object is 'hit'. Tiles that have been crossed by previous lines can be excluded. Figure 2-7 clarifies the use of the algorithm.



(a) The algorithm finds the tiles (blue) corresponding to 5 different angles (black lines)



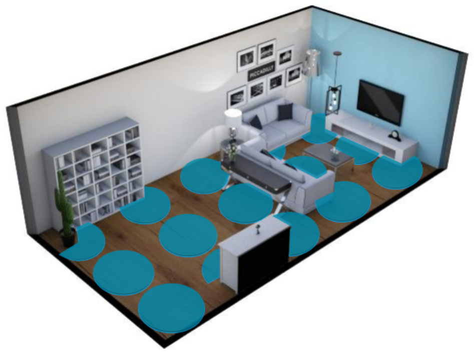
(b) After a full scan of the environment, the blue area is returned by Bresenham's algorithm

Figure 2-7: Bresenham's algorithm

All grid tiles that are partly crossed by a line are included in the entropy prediction. If a line is blocked by an obstacle, the remaining tiles will not be counted. A set of (x, y) coordinates, corresponding to tiles, can be found by casting lines in a complete circle around the agent. The best-case scenario is taken, which implies that unless a ray is certainly blocked, the assumption is made that it will reach the entire space within its maximum radius. The sum of the covered tiles is called the potential of the origin tile. The found potentials can be combined in a potential field.

2-3-3 Potential Field

For a given room, every tile has a potential. This potential is the possible entropy reduction from that tile if the agent were to be located there. By finding the potential for every tile in the room, a potential field can be generated. Figure 2-8 shows the potential field of an example room. The inverse potential is used so that a terrain-like structure arises, where the lowest point in the terrain is the location the agent wants to be in most, since it yields the highest entropy gain. By representing it in this way, one can model the robot as if it were a ball rolling into the valley of the terrain. This rolling motion then represents a path that will yield the highest information gain. Note that this figure shows the potential for every tile if no parts of the room have been explored yet. In practise, it is not possible to generate this map for an unknown room.


(a) Render of room

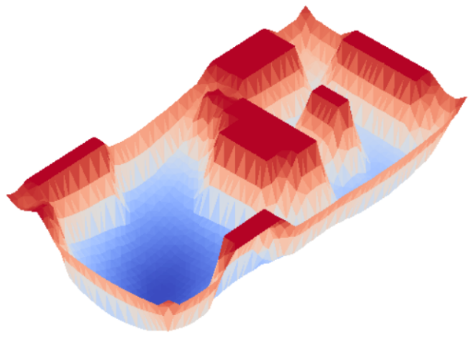

(b) Potential field of room

Figure 2-8: Potential field representation of room

Figure 2-9 shows how locally generated potential fields are used for navigation. The figures on the left show the floor plan of the room. White spaces have already been explored and represent tiles with minimal entropy. The potential fields on the right represent the entire room. Note that for unexplored areas, the potential is maximal since there is no knowledge of the presence of any obstacles yet. Once exploration takes place, areas where the agent (represented by the green dot) has already been get elevated. Any new obstacles encountered become visible because the areas right next to them have very low potential. By taking the negative gradient of the potential field, the direction which yields the highest information gain can be obtained. Following this direction creates a trajectory through the room that can be used for exploration. Note that in the last situation the map becomes largely flat since most areas are explored. The only areas with potential left are not accessible by the agent. In this situation, the agent gets stuck. The potential differences become small, and elevation changes are merely existing. In these cases, one could use a path finding algorithm, such as A^* to go to the closest area with potential to continue the search.

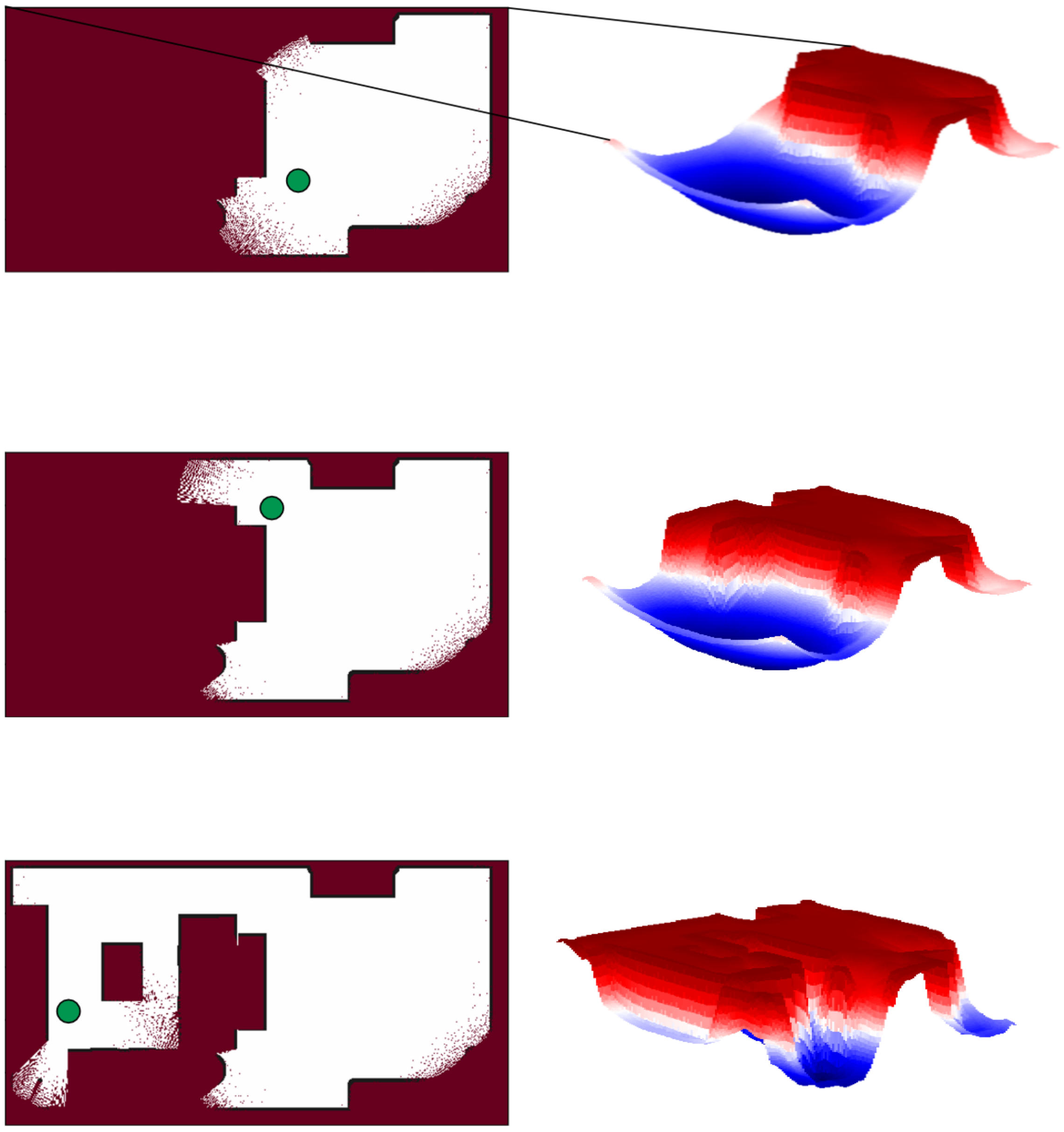


Figure 2-9: Different stages during room exploration using the potential field to navigate. Left images show the floor plan as created by the agent. The agent is represented by the circle. Right images show the corresponding potential field at that moment.

2-4 Summmary

This chapter introduced some important background information that will be used in future chapters. First, the grid and graph representations have been introduced. The grid representation is used for node placement and potential calculations. The graph representation is used for decision making within the active inference framework of Chapter 3. The concept of entropy has been explained, and it is shown how, using Bresenham's algorithm, the number of unobserved tiles within the agent's field of view can be found. Later in Chapter 4, this method will be used again to create a graph online, which will then be used with the active inference framework. The next chapter introduces this framework.

Chapter 3

Active Inference for Search and Exploration

In this chapter a description of the active inference framework for exploration is given. Section 3-1 starts with an introduction to active inference. Then Section 3-2 will show how active inference can be used to explore and search over graphs. Section 3-3 extends this framework to (partially) unknown graphs, and Section 3-4 translates the framework to be compatible with the sensory inputs generated by Spot. Section 3-5 gives a large-scale example of the developed framework. It demonstrates the framework applied to a search mission in an unknown room.

3-1 Background on Active Inference

Homeostasis is the *self-regulating process by which biological systems maintain stability while adjusting to changing external conditions.* Several brain models are based on this neuroscientific concept. One such model is the free energy principle of the brain [18]. When this principle is used to come up with desired actions to take, we are talking about active inference. This section will look at the free energy principle and how it can be used to find the best actions for an agent to take. The aim is to find an active inference implementation for a mobile robot that navigates through an unknown environment. The focus is on active inference using discrete state space and time. Especially implementations using Partially Observable Markov Decision Process (POMDP) as internal models are considered. All calculations are done using the PYMDP package for Python [34]. The notation of the theoretical background in this section is primarily based on Solopchuk's introduction to active inference [35]. The modelling as POMDPs is based on the step by step tutorial by Smith et al. [24] which also contains a clear introduction to the theoretical principles behind active inference.

3-1-1 The Free Energy Principle

The goal of the brain is to ensure that certain states, such as body temperature or energy levels, remain close to their reference values. Deviations from these references produce unexpected sensory inputs. These unexpected inputs can be seen as surprises. Therefore, the function of the brain can be viewed as minimising surprise. Since it is only possible to infer these deviations from sensory measurements, be it through nerves or digital sensors, the brain aims to minimise the surprise of sensory observations instead. The real world is described by the generative process $R(s^*, o)$. The true hidden states are described by s^*. The agent is unable to know the true hidden states. However, the generative process 'generates' observations which the agent can use to infer the true hidden states s^* to find the inferred states s. Inferring the hidden states requires a model of the generative process, the generative model. The same model can then be used to find actions. Actions are chosen so that the hidden states change in a way that results in the agent getting preferred observations. A sequence of actions is called a plan π. Figure 3-1 illustrates this interaction between the agent and the environment.

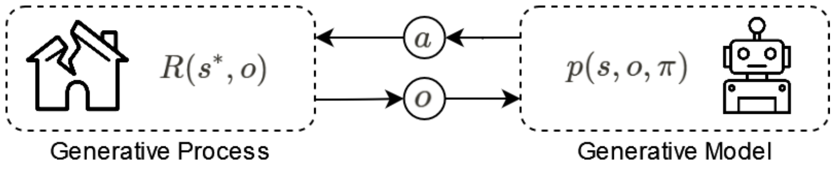


Figure 3-1: Illustration of the interaction between the generative process (hidden states) and generative model (agent's belief)

Inference over the states can be done in an optimal way by Bayes' rule:

$$p(s|o) = \frac{p(o|s)p(s)}{p(o)} \tag{3-1}$$

Thus, the hidden states, given observations $p(s|o)$ are inferred using the prior belief $p(s)$, the likelihood $p(o|s)$ and the model evidence $p(o)$. This last term is called model evidence because the better the model, the higher the probability of the observed data. Now, the surprisal, which has to be minimised, is defined as the negative log of the model evidence:

$$-\log p(o) \tag{3-2}$$

Thus, observation with low probability yields high surprisal and vice versa. In addition, minimising the surprisal is equivalent to maximising the model evidence. However, it is often challenging to compute $p(o)$ directly, since it requires summing out all the hidden states s from the joint distribution.

Free Energy

However, an approximation can be used that is easier to work with. To do so, the surprisal will be rewritten:

$$-\log p(o) = -\log \sum_s p(o, s) = -\log \sum_s q(s) \frac{p(o, s)}{q(s)} \tag{3-3}$$

Here, the surprisal is multiplied by 1, which introduces the dummy distribution $q(s)$. This distribution is an approximation on the posterior, and its use will become clear later on. Next, the upper bound on the surprisal is taken:

$$-\log \sum_s q(s) \frac{p(o,s)}{q(s)} \leq -\sum_s q(s) \log \frac{p(o,s)}{q(s)} = F \tag{3-4}$$

The right-hand side of this equation forms an upper bound on the surprisal and is called the Free Energy. Minimising Free Energy indirectly minimises the surprisal. Let us now further rewrite to a form that gives us some more insight. By the use of:

$$p(a,b) = p(b|a)(a) \tag{3-5}$$

F can be rewritten as:

$$F = \sum_s q(s) \log \frac{q(s)}{p(s|o)p(o)} = \sum_s q(s) \log \frac{q(s)}{p(s|o)} - \log p(o) \tag{3-6}$$

Next, the Kullback-Leibler divergence (KL) is introduced. This is the statistical distance between two distributions $p(a)$ and $q(a)$. If both distributions are equal, the KL-divergence is zero. The KL-divergence is defined as:

$$KL(q(a)\|p(a)) = \sum_a q(a) \log \frac{q(a)}{p(a)} \tag{3-7}$$

Now, further rewriting of the free energy results in the following.

$$F = KL(q(s)\|p(s|o)) - \log p(o) \tag{3-8}$$

Thus, the free energy is equal to the Kullback-Leibler divergence between $q(s)$ and $p(s|o)$ plus the surprisal. This means that when the KL-term becomes zero, thus, when the introduced dummy distribution matches $p(s|o)$, the Free Energy is equal to the surprisal. Therefore, by adjusting $q(s)$, a stricter bound on the suprisal can be established, while at the same time the free energy gets minimised. The next question is how this minimisation of F takes place. The goal is to minimise F over future time steps; therefore, the switch will be made to another quantity, namely the Expected Free Energy (EFE).

Expected Free Energy

Frist, the free energy of Equation 3-4 will be rewritten to:

$$F = \sum_s q(s) \log \frac{q(s)}{p(s)p(o|s)} \tag{3-9}$$

The (passive) free energy F can be made active by introducing actions. Actions can change the true hidden states of the world. By choosing the right actions, $q(s)$ can be altered so that it minimises free energy. Actions are presented as plans π. A plan is a sequence of actions for multiple time steps in the future. For these time steps, the separate free energy contributions can be summed, and together they form the expected free energy. By including the plans, the free energy can be transformed to the expected free energy G:

$$G = \sum_s q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(s_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}, s_\mathcal{T}|\pi\right)} = \sum_s q\left(s_\mathcal{T}|\pi\right) \sum_o p\left(o_\mathcal{T}|s_\mathcal{T}\right) \log \frac{q\left(s_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}, s_\mathcal{T}|\pi\right)} \tag{3-10}$$

Here $q(s_\mathcal{T}|\pi)$ is the approximation of the posterior given a plan. How this plan influences this approximation is part of the model and will be discussed later. Another part of the model is $p(o_\mathcal{T}|s_\mathcal{T})$. This includes observations that are expected in the future, given the states in the future. This part is the same for every plan. Now, further rewriting yields the following:

$$G = \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(s_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}, s_\mathcal{T}|\pi\right)} \tag{3-11}$$

This, with the use of Equation 3-5 can be further rewritten to:

$$G = \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(s_\mathcal{T}|\pi\right)}{p\left(s_\mathcal{T}|o_\mathcal{T}, \pi\right) p\left(o_\mathcal{T}\right)} \tag{3-12}$$

This introduces two other terms. The distribution $p(o_\mathcal{T})$ is the prior preference over future outcomes. It is something specified by the designer and encodes a distribution over future observations that is preferred. Now, further rewriting eliminates the true posterior $p(s_\mathcal{T}|o_\mathcal{T}, \pi)$. To do so, the log will be split:

$$G = \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(s_\mathcal{T}|\pi\right)}{p\left(s_\mathcal{T}|o_\mathcal{T}, \pi\right)} - \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log p\left(o_\mathcal{T}\right) \tag{3-13}$$

Now, by using Bayes' rule (Equation 3-1), this can be rewritten to:

$$G = \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(o_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}|s_\mathcal{T}, \pi\right)} - \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log p\left(o_\mathcal{T}\right) \tag{3-14}$$

By combining the logs and splitting them, the EFE can be rewritten in a different manner:

$$G = \sum_{o,s} p\left(o_\mathcal{T}|s_\mathcal{T}\right) q\left(s_\mathcal{T}|\pi\right) \log \frac{q\left(o_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}\right) p\left(o_\mathcal{T}|s_\mathcal{T}, \pi\right)} \tag{3-15}$$

$$= \sum_{o,s} q\left(s_\mathcal{T}|\pi\right) p\left(o_\mathcal{T}|s_\mathcal{T}\right) \log \frac{q\left(o_\mathcal{T}|\pi\right)}{p\left(o_\mathcal{T}\right)} - \sum_s q\left(s_\mathcal{T}|\pi\right) \sum_o p\left(o_\mathcal{T}|s_\mathcal{T}\right) \log p\left(o_\mathcal{T}|s_\mathcal{T}\right) \tag{3-16}$$

Now, by using the KL-divergence and the Shannon entropy H from Section 2-3, the EFE takes its final form:

$$G = \underbrace{KL\left(q\left(o_\mathcal{T}|\pi\right) \| p\left(o_\mathcal{T}\right)\right)}_{\text{reward seeking}} + \underbrace{\sum_s q\left(s_\mathcal{T}|\pi\right) H\left[p\left(o_\mathcal{T}|s_\mathcal{T}\right)\right]}_{\text{information seeking}} \tag{3-17}$$

This form gives more insight into how EFE calculations can adjust the agent's behaviour. The reward seeking part is the KL-divergence between the 'desired' observations $p(o_\mathcal{T})$ and the expected observations under a plan. So policies that yield expected observations that are wanted are rewarded. The information-seeking part depicts the uncertainty between the states and observations $p(o_\tau|s_\tau)$. Plans that yield lower uncertainty over states are thus rewarded. There are many forms in which the EFE can be given, multiple of them are given by Friston et al. [36]. Within the PYMDP package, the Expected Free Energy (EFE) is given in the following form:

$$-\,\text{EFE} = \text{State Information Gain} + \text{Utility} \tag{3-18}$$

Note that PYMDP returns the negative EFE. In the above formula **State Information Gain** (SIG) represents information-seeking behaviour and **Utility** (UT) represents the reward-seeking behaviour. SIG and UT contributions will often be mentioned during examples in the remainder of this report. For more information on how these are defined and calculated, see the PYMDP documentation [34].

The goal of active inference is to find the plan π that has the lowest corresponding EFE. Generally, this is done by calculating the EFE for all plans, after which the best plan is chosen. Now the remaining question is how the terms from Equation 3-17: $q(o_\tau|\pi)$, $q(s_\tau|\pi)$ and $p(o_\tau|s_\tau)$ can be modelled and how $p(o_\tau)$ should be defined. The generative model will be constructed conform a POMDP formulation.

3-1-2 Creating the Generative Model

Table 3-1 introduces all the necessary and important dimensions used in the different components of the model. Table 3-2 gives an overview of all the elements of the model and their related EFE formulation. The formulations are based on the POMDP formulation described by Smith et al. [24]. And they comply with the requirements of the PYMDP solvers [34]. Next, several aspects of Table 3-2 will be discussed in more detail.

Table 3-1: Symbols regarding the dimensions of the model and their description.

Symbol	Description
$\mathbb{I}$	the set of state-factors i that are part of the model
m^i	number of mutually exclusive values in state factor i
$\mathbb{J}$	the set of outcome modalities j that are part of the model
w^j	number of possible observations within outcome modality j
v^i	total number of actions corresponding to state factor i
p	plan-length. Number of actions (or steps) in π.

Table 3-2: Description of the model symbols and their relation to the EFE equation (Eq. 3-17).

Symbol	Eq. 3-17	Description
s^i	s_τ	state-factor i, $s^i \in \mathbb{R}^{m^i}$, $i \in \mathbb{I}$
o^j	o_τ	outcome modality j, $o^j \in \mathbb{R}^{w^j}$, $j \in \mathbb{J}$
a^i	$\pi(i, \tau)$	actions corresponding to state-factor i. $a^i \in \mathbb{N}^{v^i}$
π	π	plan, set of actions over p steps in the future
$A\{o^j\}$	$p(o_\tau \| s_\tau)$	mapping from hidden states to observations
$B\{s^i\}$	$q(s_\tau \| \pi)$	state transition under a given plan, $B\{s^i\} \in \mathbb{R}^{m^i \times m^i \times v^i}$
$C\{o^j\}$	$p(o_\tau)$	encodes desired distribution over observations, $C\{o^j\} \in \mathbb{R}^{w^j}$
$D\{s^i\}$	$q(s_{t=0})$	encodes intial prior beliefs over the states, $D\{s^i\} \in \mathbb{R}^{m^i}$

State Factor

Every model consists out of one or more state factors s^i. The superscript labels which state factor is regarded. For example, later the state of the robot (or agent) s^r and the state of the victim s^v will be introduced. The state factor at a given time τ is a column vector $\in \mathbb{R}^{m^j}$ with m^j a number of mutually exclusive values. The number of states m can be different for different state factors.

Outcome Modality

Every model exists out of one or multiple outcome modalities o^j. Like with state factors, the superscript labels the modality. The outcome modalities consist of w^j observations. For example, the outcome modality regarding the victim o^v can have the observation *victim observed* or *victim not observed*, making the modality consist of two observations ($w^j = 2$). The outcome modality at a given time τ is a column vector $\in \mathbb{R}^{w^j}$.

Plans and Actions

For every state factor s^i there is a set of v^i possible actions. For example, the state of the robot s^r could change because the agent can move left and right. In that case, there are two actions $v^i = 2$. The plan π contains a set of actions from $\tau = t_0$ to $\tau = t_p$ with p the length of the plan. The plan π contains the corresponding action for each state factor at every time step between t_0 and t_p.

A-tensor (likelihood tensor)

The A-tensor encodes the probability of making an observation given the current states. It is defined as:

$$A\{o^j\}(o^j, s^{\mathbb{I}}) \tag{3-19}$$

Here $A\{o^j\}$ means the A-tensor corresponding to the outcome modality o^j. All the state factors that are part of the model are represented by $s^{\mathbb{I}}$. The part between parentheses shows the corresponding entries of the tensor. For example, $A\{o^v\}(o^v, s^r, s^v)$ consists of multiple matrices (layers) in which the rows show the possible observations corresponding to o^v. The columns show the possible states of the robot s^r and one matrix is defined for every possible location of the victim s^v.

B-tensor (state-update tensor)

The state-update tensor encodes how for every state factor s^i the states change depending on a given action a from τ to $\tau + 1$. It is given by:

$$B\{s^i\}(s^i_{\tau+1}, s^i_\tau, a) \tag{3-20}$$

Here $B\{s^i\} \in \mathbb{R}^{m^i \times m^i \times v^i}$ is a tensor. It consists of v^i layers (one layer for every possible action in s^i) where each layer is a matrix. The columns correspond to the current state. The rows correspond to the state in the next step.

C-vector (prior preferences)

For every outcome modality, there is a distribution over the outcomes that are prefered $C\{o^j\} \in \mathbb{R}^{w^j}$. This vector represents the traditional reward as known from reinforcement learning. For example, the outcome modality o^v for the victim with two observations (observed / not observed) could encode a strong preference to observe the victim using $C\{o^v\} = [1\ 0]^T$.

D-vector (initial prior)

The D-vector $D\{s^i\} \in \mathbb{R}^{m^i}$ encodes the initial belief over the states for every state factor s^i. If there is no prior knowledge on the state, the uniform distribution is used.

Note that parenthesis are used throughout this whole chapter to indicate vector and tensor entries, and thus do not show variable dependencies of functions. Furthermore, most vectors are shown as row vectors, which is consistent with how they are entered into the PYMDP solvers. In theoretic notation, they are often defined as column vectors.

Table 3-3: Overview of the symbols, tensors, matrices and vectors used in Chapter 3.

Symbol	**Explanation**
n_n	Number of nodes in a graph.
n_d	Number of orientations (directions) the robot can take.
s^r	State of the robot, encodes the current node the robot is in.
s^d	Direction of the robot, encodes the current orientation of the robot.
s^v	State of the victim, encodes the current node (and direction) the victim is in.
o^v	Victim outcome modality, possible observations are 'victim observed', 'victim observed in neighbouring node' and 'victim not observed'.
Θ	Node upon arrival Θ_A and best (Θ_1) to worst (Θ_{n_d}) directions.
o^N	Direction outcome modality (North / Θ_A), Possible observations are 'victim observed' and terrain map scores.
o^E	Direction outcome modality (East / Θ_1).
o^S	Direction outcome modality (South / Θ_2).
o^W	Direction outcome modality (West / Θ_3).
$A\{o^v\}$	Likelihood tensor; maps states to expected victim observations.
$B\{s^r\}$	State-update tensor for the robot's location. Links actions to how the robot traverses along the edges of the graph. $B\{s^r\} \in \mathbb{R}^{n_n \times n_n \times n_n}$
$B\{s^d\}$	State-update tensor for the robot's direction. Links actions to how the robot changes orientation. $B\{s^d\} \in \mathbb{R}^{n_d \times n_d \times n_d}$
$B\{s^v\}$	State-update tensor for the victim's location. Shows how the victim traverses along the edges of the graph (independent on actions). $B\{s^v\} \in \mathbb{R}^{n_n n_d \times n_n n_d \times 1}$
$C\{o^v\}$	Prior preferences on the victim observations. Encodes that observing the victim is rewarded.
$D\{s^r\}$	Initial belief on the robot's location. Vector of zeros $\in \mathbb{R}^{n_n}$ with single entry of 1 for the intial position.
$D\{s^d\}$	Initial belief on the robot's orientation. Vector of zeros $\in \mathbb{R}^{n_d}$ with single entry of 1 for the initial direction.
$D\{s^v\}$	Initial belief on the victim's location. Uniformly distributed vector $\in \mathrm{R}^{n_n n_d}$ to encode full uncertainty on victim's location.
ϕ	Vector indicating how strong the belief is in a node's existence. $\phi \in \mathbb{R}^{n_n}$
ϕ_F	Vector indicating how strong the belief is in a node to be accessible from the current frontier. $\phi_F \in \mathbb{R}^{n_n}$
G_A	Adjecancy matrix of a graph.
K	Adjecancy matrix of the known part of a graph.
Q	Adjecancy matrix encoding how frontiers link the known and unknown parts of a graph.
L	Adjecancy matrix of a line-graph. Representing the unknown parts of a graph.
n_s	Number of possible scores to be obtained from the terrain map.
T	Vector encoding the magnitude of the possible scores that can be obtained from the terrain map. T $\in \mathbb{R}^{n_s}$
S_v	Distribution over expected score if the victim is present. $S_v \in \mathbb{R}^{n_s}$
S_e	Distribution over expected scores if the victim is not present. $S_e \in \mathbb{R}^{n_s}$
ξ	Expected total score for a node, for a room without victim.

3-2 Active Inference on Graphs

The previous section gave an introduction to active inference and showed how the generative model can be formulated by a set of tensors and vectors. In the remainder of this chapter, the models that can be used to carry out search and rescue missions will be constructed. A graph representation as introduced in Section 2-1-2 will be used. Each section starts with a theoretical framework and will contain one or more examples that are used to clarify the model and confirm the expected behaviour by the robot. Every section extends the framework, and unless stated otherwise, all assumptions, tensors, and vectors as defined in earlier sections remain. Table 3-3 on page 27 gives an overview of the important symbols, tensors, and vectors. For this first section, a general active inference framework to navigate and search over graphs will be introduced.

3-2-1 Navigation over Graphs

Let us start with a simple graph of size n_n. The assumption is made that the agent will not deviate while traversing the graph's edges. Therefore, it will always end up precisely at the node it is instructed to move to. If the agent is commanded to move to a node that is not a neighbour of the agent's current node, it will stay in place. The first state factor that is introduced is s^r; the state of the robot or agent. It encodes the current location of the agent. It is defined as a vector whose dimensions correspond to the number of nodes n_n in the graph. The agent's starting location c_0 is assumed to be known. The initial belief on the agent's location can now be defined as:

$$D\{s^r\} = \begin{bmatrix}\alpha_0 & \alpha_1 & \dots & \alpha_n\end{bmatrix} \tag{3-21}$$

Where α_i is 0 for $i \neq c_0$, with c_0 being the agent's starting node, and $\alpha_{c_0} = 1$. We assume full certainty over the agent's location, thus only the B-tensor has to be defined for s^r. If action a_i is defined so that it is the action that lets the agent move from its current node to node i, the different layers of the B-tensor are given by:

$$B\{s^r\}(s^r_{\tau+1}, s^r_\tau, a_i) = G_A \odot \bar{\beta} + I - I \odot G_{Ai} \tag{3-22}$$

$$\bar{\beta} = \begin{bmatrix}\beta_0 & \beta_1 & \dots & \beta_{n_n}\end{bmatrix}^\mathsf{T}, \qquad \beta_i = 1, \quad \beta_{j\neq i} = 0$$

In the remainder of this report, the symbol $\odot$ is used to indicate element-wise multiplication. In the above equation G_A is the adjacency matrix of the graph. G_{Ai} is the i-th row of G_A. This equation selects the nodes connected to node i. The identity matrix is added so that for nodes from which the action cannot take place, the agent will stay at its current node. The added terms are then removed for nodes for which they are unnecessary. Note that by following this method, a total of n_n actions have to be defined. This also means that the B-tensor will consist of n_n layers. This method gives better insight, since actions can easily be translated to what is happening on the graph. However, the model can be generated with fewer actions. The minimum number of actions needed equals the maximum degree over all the nodes of the graph:

$$n_a = \max(G_A \mathbb{1}) \tag{3-23}$$

This smaller B-tensor is constructed by taking the i-th nonzero entry for every column in the adjacency matrix, with i the corresponding action, thus layer in the tensor. If a column has fewer than i nonzero entries, it will be replaced by the identity entry for that column. This means that the action does not change the state, since it cannot be carried out from that location. Algorithm 1 describes this process. The benefit of this approach is that the number of layers (or actions) and, therefore, the number of possible plans can be significantly reduced. This is especially true for larger graphs and cluttered areas. The downside is that it makes the model less intuitive, since actions are no longer directly linked to the corresponding states. Therefore, it may be beneficial to use the earlier formulation from Equation 3-22 for analysis, illustrations, and debugging.

Algorithm 1: translates the adjacency matrix G_A to the B-tensor, with minimal dimensions

```
   Input  : adjecancy matrix $G_A$ and tensor of zeros $B$
   Output: B-tensor $B$
1  for $a$ in the range of $n_a$ do
2      for columns $c$ in $G_A$ with index $c_i$ do
3          if nonzero entries in $c > a$ then
4              select $e_a$ as the $a$-th nonzero entry of the column
5              B($e_a$, $c_i$, $a$) = 1
6          else
7              B($c_i$, $c_i$, $a$) = 1
8          end
9      end
10 end
```

3-2-2 Search over Graph

Within this report, the emphasis is on active inference for search and rescue missions. Therefore, the victim will be added as a state factor s^v to the model. Where $s^v \in \mathbb{R}^{n_n}$ since the victim can be located at any of the nodes. One of the objectives is to reduce the uncertainty about the location of the victim. Since the victim's location is unknown at the start, it will be uniformly distributed over all the nodes in the graph. The initial belief on the victim's location is given by:

$$D\{s^v\} = \begin{bmatrix} \frac{1}{n_n} & \frac{1}{n_n} & \dots & \frac{1}{n_n} \end{bmatrix}^{1\times n_n} = (\mathbb{1}^{n_n})^{\mathsf{T}} \frac{1}{n_n} \tag{3-24}$$

With $\mathbb{1}$ being the column vector of all ones. The victim will be allowed to traverse along the graph edges, just like the agent does. The victim can also stay in place, and the probabilities that the victim will move or remain can differ from each other. The state-update tensor (B-tensor) for the victim can be constructed by:

$$B\{s^v\}(s^v_{\tau+1}, s^v_\tau) = (1-\alpha_v)\tilde{G_A} + \alpha_v I \tag{3-25}$$

Here, α_v represents the probability that the victim stays in place. The matrix $\tilde{G_A}$ is the adjusted version of the adjecancy matrix such that all columns sum up to 1. Note that $B\{s^v\}$ is just a matrix, since there are no separate actions describing the victim's movement, there is only one layer. Next, the outcome modality o^v, which defines the observations regarding the victim, is introduced. For the likelihood tensor (A-tensor), there are several possibilities. For this section, a model will be used in which the victim will certainly be observed if it is at the same node as the agent. If the victim is in a neighbouring node of the agent, there is a 50% chance of observing it. This percentage is arbitrary and is chosen to illustrate the different influences on the EFE calculations. If the victim is at any of the other nodes, the victim will not be observed. The several layers of the A-tensor can then be constructed using:

Algorithm 2: Forms the A-tensor such that the agent can observe victims either in the current node (0), another node (1), or not at all (2).

```
   Input  : tensor of zeros $A$
   Output: A-tensor $A$
1  for all victim locations $l_v$ do
2      for all agent locations $l_a$ do
3          if $l_a == l_v$ then
4              A(0, $l_a$, $l_v$) = 1
5          end
6          if $l_v$ in neighbours $l_a$ then
7              A(1, $l_a$, $l_v$) = 1/2
8          end
9          A(2, $l_a$, $l_v$) = 1 - sum(A(:, $l_a$, $l_v$))
10     end
11 end
```

Last is the C-vector. Depending on what the preferred behaviour is, rewards can be given for observing the agent at the current node or a neighbouring node. This will be illustrated in the following example.

Example: Search and Navigate

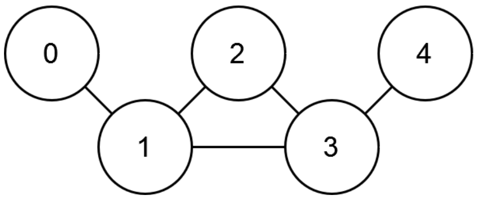


Figure 3-2: Example graph

Figure 3-2 shows a graph of 5 nodes. The generative model for this graph will be created, and the active inference problem will be solved. The agent will start at node 0, and the victim is located at node 4. The adjacency matrix of the graph is given by:

$$G_A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3-26}$$

The initial belief in the agent's location s^r and on the victim's location s^v are:

$$D\{s^r\} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-27}$$

$$D\{s^v\} = \begin{bmatrix} .2 & .2 & .2 & .2 & .2 \end{bmatrix} \tag{3-28}$$

Now, the assumption is made that the victim stays in place. Therefore, $B\{s^v\}$ will be the identity matrix I. The B-tensor for the actions of the agent is derived using Equation 3-22. It consists of five layers; an example layer, corresponding to action a_1: *move to node 1*, is given below:

$$B\{s^r\}(s^r_{\tau+1}, s^r_\tau, a_1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-29}$$

The A-tensor is constructed using Algorithm 2 and consists of 5 layers, one layer for every possible location of the victim. The layer corresponding to the victim being in node 4 is given below as an example. The columns indicate the possible locations of the agent:

$$A\{o^v\}(o^v, s^r, s^{v=4}) = \begin{array}{c} \begin{array}{ccccc} n_0 & n_1 & n_2 & n_3 & n_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & .5 & 0 \\ 1 & 1 & 1 & .5 & 0 \end{bmatrix} \end{array} \begin{array}{l} \text{victim observed} \\ \text{victim in neighbour} \\ \text{victim not observed} \end{array} \tag{3-30}$$

Next, the C-vector will be defined so that ending up in the same node as the victim will be rewarded:

$$C\{o^v\} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} \text{victim observed} \\ \text{victim in neighbour} \\ \text{victim not observed} \end{array} \tag{3-31}$$

Note that this can be changed according to personal preferences. Other possibilities are, for example, to also reward observing the victim is a neighbouring node. Using $C = \begin{bmatrix} 2 & 1 & 0 \end{bmatrix}^\mathsf{T}$ rewards both observations, although the preference to observe the victim at the same node as the agent is higher.

For this example, a plan length of $p = 1$ is used. Table 3-4 shows the possible actions and their corresponding State Information Gain (SIG) and Utility (UT) for each step during the simulation. A step consists of 1 action or movement by the agent. Note that the values shown in the table are of the negeative EFE. This is conform the values as returned by the solver. The expected observations for the possible actions are shown in the table as well. The separate actions at the same step indicate alternative plans. Only the action with the best (lowest) EFE contribution is executed. Table 3-5 shows how the belief in the location of the victim changes after each action.

Table 3-4: The EFE contribution for given actions starting from a certain node. Actions with a minimal EFE contribution have been left out. Expected observations by conducting the given action are displayed on the right-hand side.

step	node	action	SIG	UT	-EFE	vic.	neigh. vic.	no vic.
t0	0	1	0.56	-1.41	-0.85	0.14	0.29	0.57
t1	1	**2**	0.73	-1.3	**-0.57**	0.25	0.125	0.625
		3	0.56	-1.3	-0.74	0.25	0.375	0.375
t2	2	**3**	0.5	-0.85	**-0.35**	0.2	0.4	0.4
		1	0.14	-1.55	-1.41	0	0.56	0.94
t3	3	4	0	-0.55	-0.55	1	0	0

Table 3-5: The belief on the victim's location for every time step during execution.

step	n0	n1	n2	n3	n4
t0	0	0.14	0.29	0.29	0.29
t1	0	0	0.25	0.25	0.5
t2	0	0	0	0.2	0.8
t3	0	0	0	0	1

Some interesting aspects are that, for earlier actions, the main contribution is given by the state information gain; for later actions, where the certainty in finding the victim is larger, the utility becomes of greater importance. An interesting moment is when the agent is in node 1 and has to decide between nodes 2 and 3. In this case, it chooses to go to node 2 because of the higher state information gain. One may have expected that the highest state information gain would be from the node with the highest degree centrality. This is node 3 since from node 3 the agent can see both nodes 2 and 4. To gain a better understanding of the EFE calculations, this step will be discussed in more detail. Utility contributions are only given if the victim is observed in the same node as the robot (conform the C-vector). Since both nodes 2 and 3 have the same probability that the victim is located there, the utility values are equal. Now, for state information, let us look at the six possible situations:

1. Move to node 2
   (a) Victim observed: there is full certainty over the victim's location, maximum state information gain.
   (b) Victim in a neighbour: there is full certainty over the victim's location, since the victim is not in node 1, it has to be in node 3.
   (c) No victim: victim is in node 3 or 4. With node 4 being four times as likely.
2. Move to node 3
   (a) Victim observed: full certainty, thus maximum information gain.
   (b) Victim in a neighbour: the victim can be in node 2 or 4 with node 4 being twice as likely.
   (c) No victim: the victim can be in node 2 or 4, with node 4 being twice as likely.

Looking at the different scenarios, the main difference between going to node 2 or node 3 is when the victim is observed in a neighbouring node. From node 3 this does not yield any new information. From node 2 however, it gives full certainty over the state of the victim. In reality, the agent would know in which neighbouring node it observes the victim. To do so, directionality must be taken into account. This is discussed in Section 3-4. First, an expansion to the framework will be introduced for unknown graphs.

3-3 Active Inference with Frontiers

In this section, the model will be extended to unknown parts of the graph. For a search and rescue mission, the environment will often be unknown. his means that the adjecancy matrix has to be created during execution. This implies that for planning, EFE calculations need to be made over unknown parts of the graph. First, the model will be adjusted to include frontiers by adding uncertainty on the existence of nodes. In addition, assumptions have to be made on the structure of the unvisited part of the graph. After the framework is explained, some examples will illustrate the implications of this new model and how it affects the agent's behaviour.

3-3-1 Generative Model with one Frontier

Missions will be carried out in unknown environments; therefore, the shape of the graph is not known beforehand. However, it is reasonable to assume that there is some foreknowledge on the final size of the graph. To incorporate this into the model, a new state factor is introduced; the total size of the graph s^T. The new state factor contains the belief on the graph size after exploration has been completed. The initial prior $D\{s^T\}$ is an assumption and should be defined based on expectations or previous experiences. For this report, a normal distribution will be used, where the mean μ is the expected total graph size, and the standard deviation σ can be used to alter the trust in this expectation. The entries of $D\{s^T\}$ can then be calculated for any node x in the graph, using the probability density function of the normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{3-32}$$

The D-vector will be extended consecutively. When the probability of a graph size drops below a certain threshold α_T, no new nodes will be added. At this point, the obtained $D\{s^T\}$ will be normalised. In the remainder of this report, the threshold $\alpha = 0.01$ will be used. The total graph size can be translated to the probability that the graph exists out of at least x nodes. This is done using the cumulative normal distribution:

$$\Phi(x) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \tag{3-33}$$

Or, in practise, $D\{s^T\}$ can be translated directly. The probability that the graph contains at least x nodes can be obtained by:

$$\phi(x) = 1 - \sum_{i=0}^{x-1} D\{s_T\}(i) \tag{3-34}$$

The corresponding values for every node x in the graph are stored in the vector $\phi \in R^{n_n}$. The vector ϕ is needed to define the D-vector for the victim's location. The initial belief that the victim is located at node x can now be found by:

$$D\{s^v\}(x) = \frac{\phi(x)}{\mu} \tag{3-35}$$

During execution, the belief on the existence of a node can change. Nodes that had reduced belief at the beginning can be discovered eventually, which gives certainty on its existence.

When this happens the belief has to be updated. During execution, the belief on the victim's location changes due to inference. This is done using a fixed-point iteration algorithm that represents Bayesian inference:

$$p(s|o) = \frac{p(o|s)}{p(o)} p(s) \tag{3-36}$$

Note that the updated version depends on both the model $p(o|s)$ and the prior $p(s)$, or the inferred states in the previous step. When new nodes are discovered, ϕ changes, and the initial prior as calculated in Equation 3-35 no longer holds. When this happens, the current belief is updated by an additional inference step. This inference step uses current knowledge on ϕ, together with all past observations, to come up with the best updated belief. When nodes that were uncertain to exist are added to the graph, the mean on the total graph size μ changes slightly. To account for this change, μ in Equation 3-35 will be replaced by the adjusted expected graph size $\tilde{\mu}$. Here $\tilde{\mu}$ is the sum over all values of ϕ.

To construct the B-tensor an assumption is needed on the expected graph structure. There are several options, such as a best-case scenario, where the unknown parts can be modelled as a fully connected graph. A worst-case scenario could be a tree graph in which one has to traverse back to the origin before being able to visit a new node. In the remainder of this report, the unknown parts of the graph are modelled as a line graph. The line graph incorporates that the agent has to traverse links twice if it wants to return to its starting location. It is also close to what is to be expected in buildings, which are often clutered and where rooms are often linked by small halways. The adjecancy matrix can now be partitioned into:

$$G_A = \begin{bmatrix} K & Q \\ Q^\intercal & L \end{bmatrix} \tag{3-37}$$

Here, K is the known portion of the graph. Q connects the known area to the unknown graph and is just a matrix of zeros instead for the frontier nodes, which will be the links between the known and unknown parts. L has the general shape of a line graph. The adjecancy matrix of a line graph of size n_L is given by:

$$L^{n_L \times n_L} = \begin{bmatrix} \mathbf{0} & I \\ 0 & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & 0 \\ I & \mathbf{0} \end{bmatrix} \tag{3-38}$$

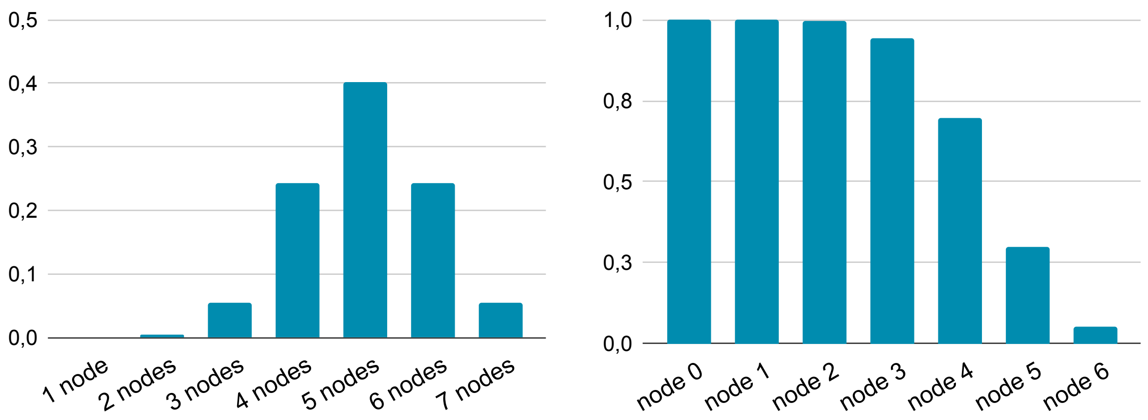
Modeling Choices

The above implementation is just one of many possibilities to incorporate the uncertainty of nodes being part of the graph. A total of three possible implementations have been explored, all with their own pros and cons. The current implementation has been chosen because it is able to achieve the desired effect, which is discounting utility and state information gain on nodes that are not certain to be part of the graph. It does so without changing the base model, keeping it simple. Calculations and updates for *phi* and $D\{s^v\}$ only have to take place when frontiers are visited. During this step, the model already needs to be adjusted since the graph's structure changes. Furthermore, since the only adjustment is in the victim's belief, it is easy to integrate with other expansions. This will become clear when the terrain map is added to the model.

Another possibility is to incorporate the belief on the total graph size directly into the model. This would mean adding another observation such that the number of newly observed nodes can be inferred over. Next, one could incorporate the possibility to observe the victim into the belief that a node is part of a graph. The A tensor for the observation would then be such that if the total graph size is less than the node number in question, the observation would be *no victim* (since the node does not exist). The main downside of this approach is that it further complicates the model. It also means that a new assumption is needed about the number of new nodes that can be observed from a frontier.

In both solutions mentioned, the agent is still able to 'visit' the nodes that are unknown in the EFE predictions. However, in reality, nodes that do not exist cannot be visited. This can be incorporated into the model in several ways. One is to alter the B-tensor and make it non-deterministic. Another is to add actions as state factors and observations. In this way, it is possible to infer over actions taking place. If a node does not exist, the agent will, therefore, not be able to perform the action. It would then also be needed to infer over the agent's location. This approach would add multiple state-factors and observation. However, it seems to be the most accurate representation of the generative process. It is also more widely applicable since it uses only parts of the model that are part of graph navigation, so no victim state is needed to add uncertainty on nodes.

Example: Unknown Graph with one Frontier



(a) Initial belief on the graph's size **(b)** probability that a given node is part of the final graph

Figure 3-3: Belief in the final graph size and node existence distributions.

For this example, the graph will be unknown. The assumption is made that the graph consists of 5 nodes. To model this, $\mu = 5$ and $\sigma = 1$ are used. This yields the distributions as shown in Figure 3-3. This corresponds to:

$$D\{s_T\} = \begin{bmatrix} 0 & 0 & | & \underbrace{.05 \quad .24 \quad .40 \quad .24 \quad .05}_{\text{unknown}} \end{bmatrix} \tag{3-39}$$

$$\phi(x) = \begin{bmatrix} 1 & 1 & | & 1 & .94 & .70 & .30 & .05 \end{bmatrix} \tag{3-40}$$

Now, the initial belief for the victim to be located at a node is:

$$D\{s^v\} = \begin{bmatrix} .2 & .2 & | & .2 & .19 & .14 & .06 & .01 \end{bmatrix} \tag{3-41}$$

The initial version of the graph is shown in Figure 3-4. It consists of six nodes, the structure is not yet known, and the probability that the victim is located at a node (or the node existing) is represented by the saturation.

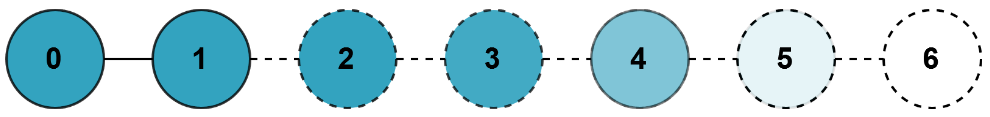


Figure 3-4: Unstructured graph of 7 nodes, with reducing belief on the victim's pressence.

For the model, the graph is assumed to be a line graph. The plan starting at node 0 and taking all necessary actions to end up at node 6 will be discussed. EFE contributions for each separate action are given in Table 3-6. Nodes with full certainty of existence have equal EFE contributions. For nodes with uncertainty, the EFE increases, and thus are less preferred.

Table 3-6: EFE contributions of the separate actions

Action	0 - 1	1 - 2	2 - 3	3 - 4	4 - 5	5 - 6
SIG	0.67	0.68	0.66	0.59	0.39	0.17
UT	-1.35	-1.35	-1.36	-1.41	-1.49	-1.54
-EFE	-0.68	-0.67	-0.70	-0.82	-1.10	-1.37

Example: Partially Known Graph with one Frontier

Now, the situation where the graph is partially known will be discussed. Let us look at the graph illustrated in Figure 3-5.

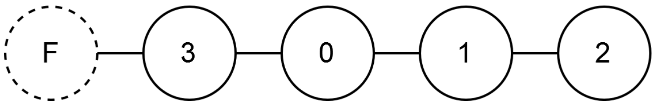


Figure 3-5: Partially known graph with one frontier

The agent starts at node 0. Now, depending on the belief on the size of the graph, the EFE changes, and thereby the decision to move either left or right from node 0. Table 3-7 shows the negative EFE for six situations, both for plans of length 2 ($p = 2$) and length 3 ($p = 3$). The difference in EFE between the two possibilities is given by Δ. The colour of the cells indicates the direction the agent prefers (yellow for left and green for right). The saturation of the cells indicates how strong this preference is. For plans of length 2 the agent will go to the left for larger expected graph sizes. This is mainly due to the higher information gain from going to the left if the graph consists of more than 5 nodes. This is because node 5 can be observed from node 4. The higher the trust in the total graph size estimate, the higher the preference. For plans of length 3 the agent always prefers to go to the left. The degree of preference depends on the estimated graph size.

Table 3-7: Table showing the different negative EFE results for different policies. Green colours show that the agent would prefer to move to the right. Orange indicates that the agent prefers to go to the left. The saturation shows how large the difference is.

	plan	**case 1**	**case 2**	**case 3**	**case 4**	**case 5**	**case 6**
μ		5	8	5	8	5	8
σ		0.5	0.5	1	1	2	2
p=2	1 - 2	-1.35	-1.77	-1.36	-1.77	-1.4	-1.77
	3 - 4	-1.4	-1.72	-1.52	-1.73	-1.68	-1.75
Δ		0.05	-0.05	0.16	-0.04	0.28	-0.02
p=3	1 - 2 - 1	-2.9	-3.32	-2.91	-3.32	-2.95	-3.32
	3 - 4 - 5	-2.65	-2.58	-2.62	-2.59	-2.72	-2.66
Δ		-0.25	-0.74	-0.29	-0.73	-0.23	-0.66

3-3-2 Generative Model with Multiple Frontiers

In the previous paragraph, only one frontier node was present. However, during exploration, generally there will be multiple nodes connected to unexplored areas of the world. The assumption is made that every frontier has equal probability of an undiscovered node being accessible exclusively from that frontier. When constructing plans, the probability of a node existing and being accesible from a frontier both need to be taken into account. Let us first introduce the probability that any undiscovered node is accessible from a given frontier p_F. Accessible here means that, to reach this undiscovered node, the agent needs to pass the corresponding frontier. This probability is the inverse of the total number of frontier nodes n_F. For example, if there are two frontiers. The chance that any undiscovered node is accessible from a certain frontier is then $p_F = n_F^{-1} = 0.5$. Now, by using the binomial distribution, the probability distribution over the number of nodes being accessible from a frontier can be found. The bionomial distribution is given by:

$$p(x = i) = \mathrm{nCr} \cdot p_F^i \cdot (1 - p_F)^{n_u - i} \tag{3-42}$$

Here, $p(x = i)$ is the probability that exactly i nodes are accessible from a frontier if there are n_u undiscovered nodes left. With nCr being the total number of possible combinations, which can be calculated by:

$$\mathrm{nCr} = (n_u)!/(i!(n_u - i)!) \tag{3-43}$$

Now, the probability of a frontier giving access to at least i nodes can be found using:

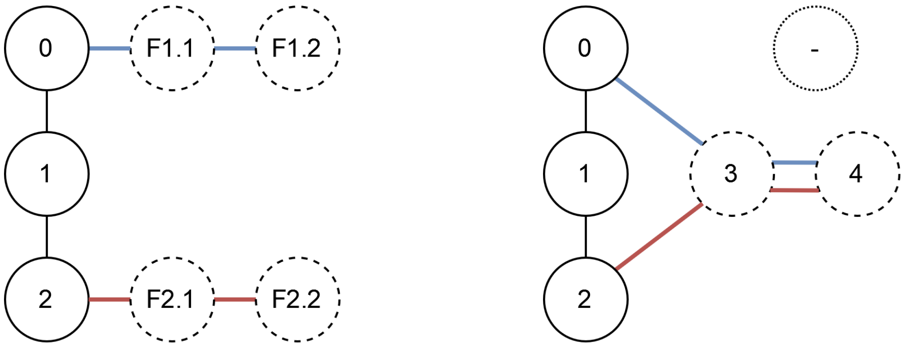
$$p(x \geq i) = 1 - \sum_{j=0}^{i-1} p(x = j) \tag{3-44}$$

Now, for every possible combination of n_u nodes left and with n_k known nodes already in the graph, and n_M being the maximum number of nodes left in the graph. The probability that a node is accessible from a frontier is given by:

$$\phi_F(i + n_k) = \sum_{n_u=1}^{n_M} p(x \geq i) \cdot D\{s^T\}(n_u + n_k) \tag{3-45}$$

This equation takes the probability that if n_u undiscovered nodes are left in the graph, that at least i of these nodes are accessible from the frontier. This is multiplied by the chance that the graph actually has n_u nodes left, which is $D\{s^T\}(n_u+n_k)$. If for every possible graph size these probabilities are summed together, the final probability that node $(i + n_k)$ is accessible from a frontier is found.

To take all possible graph constructions into account, every frontier has to be connected to n_u nodes, which would add many additional nodes to the graph. Instead of doing so, all frontiers will be linked to the same first node in the line graph that represents the unknown part of the graph. Their belief on the victim's location is adjusted as described above. This makes that the total belief on the victim's location no longer sums up to 1. Belief values corresponding to the removed nodes need to be stored to keep the distribution over the victims' location proper. This is done by adding a dummy node. Also, a constraint has to be added for the selection of admissible plans, since the agent can only exit the unknown part of the graph from the frontier from which it was entered. This concept is clarified by Figure 3-6, which shows how 2 undiscovered nodes can be represented in the graph.



(a) Adding the total possible number of nodes to every frontier.

(b) Using different paths for the same nodes.

Figure 3-6: Illustration of different ways to include two undiscovered nodes into the graph.

In Figure 3-6a for every frontier, there are two possible nodes that can be linked to it. Since equal probability is used that any node is accessible from any of the frontiers, any frontier node $F_{i.j}$ would have the same effect as any other frontier node $F_{q.j}$ on the EFE calculations. Therefore, they can be modelled as the same node, as shown in Figure 3-6b. However, it would not be possible to go from node 0 to node 2 through node 3. The different path colours show mutually exclusive paths which cannot be switched between. This constraint will be added while finding the possible plans. Note that the total belief of all nodes removed in the translation from 3-6a to 3-6b is stored in the dummy node. This makes that all beliefs still sum up to 1 and that the beliefs for the undiscovered nodes remain correct.

Example: Multiple Frontiers

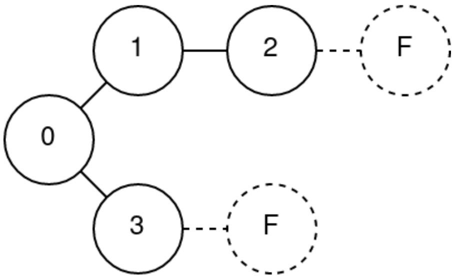


Figure 3-7: Example graph with two frontiers

The described framework will be illustrated by an example. Figure 3-7 shows the same graph as in the previous example, but with two frontier nodes now present. The expected total graph size μ is taken to be 6, with a standard deviation σ of 0.75. This results in the probability of a node being part of the graph of:

$$\phi = \begin{bmatrix} 1 & 1 & 1 & 1 & | & .99 & .77 & .23 & .02 \end{bmatrix} \tag{3-46}$$

There are two frontiers. Now, for undiscovered nodes, the probabilities that that node is accessible from a given frontier are calculated:

$$\phi_F = \begin{bmatrix} 1 & 1 & 1 & 1 & | & .71 & .25 & .03 & .0 \end{bmatrix} \tag{3-47}$$

This can be translated to the probability of the victim being located in a node considering one frontier:

$$D\{s^v\} = \begin{bmatrix} \underbrace{.17 \quad .17 \quad .17 \quad .17}_{\text{known}} & | & \underbrace{.12 \quad .04 \quad .01 \quad 0}_{\text{unknown}} & \underbrace{| \quad .17}_{\text{dummy}} \end{bmatrix} \tag{3-48}$$

Note the addition of the dummy node, which houses the total belief that the victim's locatation is accesible through the other frontier. Since there are two nodes, the belief for the dummy node is equal to the sum over the unknown part, which represents the other frontier. Now, the graph is modelled as shown in Figure 3-8.

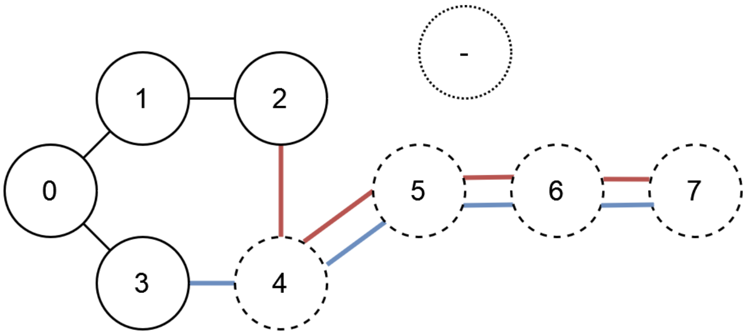


Figure 3-8: Graph with two frontiers. Within plans it is forbidden to switch from one coloured path to another.

With the corresponding adjecancy matrix being:

$$A_G = \begin{bmatrix} K & Q & \mathbf{0} \\ Q^\intercal & L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 0 \end{bmatrix} = \left[\begin{array}{cccc|cccc|c} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{3-49}$$

Now, active inference will be used to find the best plans. Plans of length $p = 4$ will be used. At each step, going to either of the two frontiers will be compared. Furthermore, the beliefs on the victim's location will be shown at every step. The results are given in Table 3-8. Note how initially going to node 1 is preferred over going to node 3. This is because node 2 is certain to exist from the beginning. The EFE difference reduces once the belief in the existence of future nodes reduces. Once node 4 is entered, the graph ends and node 3 is the only frontier left. Note how the dummy node in this case becomes 0. This concludes the section on frontier nodes. The next section will extend the model and shows how Spot's sensors can be used within the active inference framework.

Table 3-8: The EFE for different possible plans at every time step. The belief in the victim's location is shown on the right for every node in the graph.

step	**node**	**policy**	**-EFE**	n0	n1	n2	n3	n4	n5	n6	n7	d
t_0	0	1 - 2 - 4 - 5	-3.23	0	0.13	0.25	0.13	0.18	0.06	0	0	0.25
		3 - 4 - 5 - 6	-4.08									
t_1	1	2 - 4 - 5 - 6	-3.7	0	0	0.17	0.17	0.24	0.08	0.01	0	0.34
		0 - 3 - 4 - 5	-3.72									
t_2	2	4 - 5 - 6 - 7	-4.02	0	0	0	0.32	0.32	0.17	0.01	0	0.18
		1 - 0 - 3 - 5	-4.1									
t_3	4	2 - 1 - 0 - 3	-4.73	0	0	0	0.65	0	0.33	0.02	0	0

3-4 Active Inference with the Terrain Map

The terrain map was introduced in Section 2-2. This section will show how the terrain map can be included as input to the generative model. The terrain map cannot help to identify victims, but can rule out the victim's presence and can point the agent in the right direction. The terrain map is combined with the agent's front-facing cameras. To include both sensor inputs into the model, the agent's orientation needs to be added as a state-factor. This is done by adding a total of n_d directions that cover multiple areas. These will be equally divided into pie-shaped sections, where the agent forms the centre. In this report only examples and simulation will be given using 4 direction ($n_d = 4$). In addition, a total of n_s scores are added. Scores translate the terrain map to the likelihood of the victim's presence. Figure 3-9 below shows two examples for different n_d.

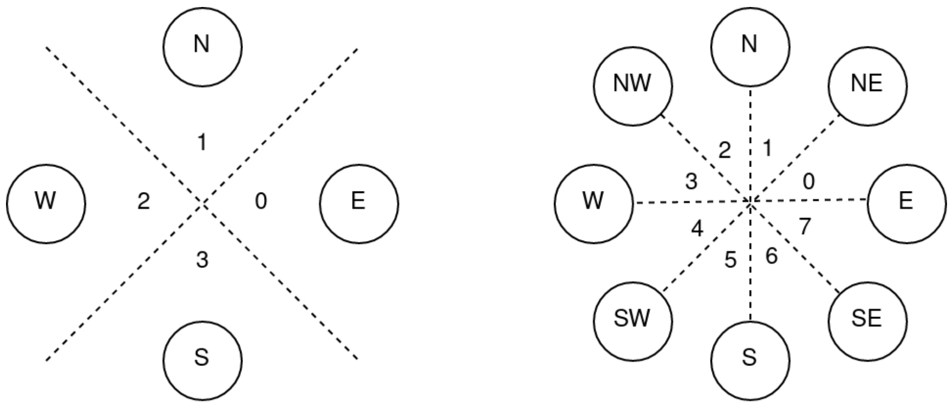


Figure 3-9: Two examples where the centre represents the agent's location. The area is split into several (numbered) areas. Letters represent the different orientations.

A direction that receives a high score from the terrain map is more likely to contain a victim than a low score. The framework is based on a linear scaling of the scores, which means that a score of 4 is twice as likely to contain the victim as a score of 2, and four times as likely compared to a score of 1. The vector of possible scores T is defined as:

$$T = \begin{bmatrix} 0 & s_0 & s_1 & \dots & s_{n_s} \end{bmatrix}^\mathsf{T} \tag{3-50}$$

To construct the A-tensor, the expected observations need to be defined both for the case that the victim is present and for when it is not present. The expected score obtained from the terrain map in a given direction, if a victim is located in the corresponding section of that terrain map, is given by S_v:

$$S_v = \frac{T}{||T||_1} \tag{3-51}$$

When a victim is not present, a score can still be obtained. To find the expected distribution over scores, when a victim is not present, an assumption on the general clutterdness of a room is needed. This assumption is made by means of ξ. This is the average score obtained within a node and has a value between 0 and $n_d T(n_s)$. With $T(n_s)$ the maximum score and n_d the number of directions that the agent can take within a node. For example, if there are four directions $n_d = 4$ and the maximum score that can be obtained in a given direction is $T(n_s) = 2$, the maximum score that can be obtained from a node is 8. Next, a uniform distribution over the scores needs to be found, such that, on average, the value of ξ is achieved in a node. The uniform distribution is used since it is very hard to predict what distribution over scores will be achieved. Furthermore, wrong predictions make for unweighted scalings in the belief on the victim's location. Using this assumption, ξ can also be related to the total number of expected onbstacles in the world O_T by:

$$\xi = \frac{||T||_1 O_T}{n_s n_n} \tag{3-52}$$

This takes the expected score per obstacle, times the number of expected obstacles, divided by the number of nodes. This yields the expected score per node. Now, the expected distribution over scores for locations where the victim is not present is given by:

$$S_e = \begin{bmatrix} 1 - ||X||_1 & X \end{bmatrix}^\intercal, \qquad X = \mathbb{1}^{1\times n_s} \frac{\xi}{n_d ||T||_1} \tag{3-53}$$

Here, X represents the uniform distribution that ensures that the expected total score obtained from a node is equal to ξ. Now, these expectations can be included in the generative model. To do so, a total of n_d new outcome modalities are added, so that for every direction, a score can be observed. The number of observations per outcome modality is $n_s + 1$, one observation per possible score, plus the observation to identify the victim. Now, the A-tensor for a given outcome modality (or direction) j is defined as:

$$A\{o^j\}(o^j, s^{v=i}) = \begin{bmatrix} q_0 & q_1 & \dots & q_{n_d} \\ 0 & 0 & \dots & 0 \end{bmatrix}, \qquad q_{j=i} = S_v, \quad q_{j\neq i} = S_e \tag{3-54}$$

Note that a row of zeros is added at the end. These zeros correspond to observing the victim at the location. It is not possible to do so by simply obtaining the terrain map, but this will be a needed observation later on when the camera input is included.

Example: Inference Using the Terrain Map

Let us take a graph consisting of only one node. There are a total of $n_d = 4$ directions (North, East, South and West). The following possible scores are defined: no victim (0), somewhat likely to find a victim (1) and likely to find a victim (2). When a total score in a node of $\xi = 2$ is assumed based on the room, the following vectors will be obtained.

$$T = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}^\intercal, \qquad S_v = \begin{bmatrix} 0 & .33 & .66 \end{bmatrix}^\intercal \tag{3-55}$$

$$S_e = \begin{bmatrix} .67 & .17 & .17 \end{bmatrix}^\intercal \tag{3-56}$$

The A-tensor, for observing the area north of the agent, is defined as:

$$A\{o^N\}(o^N, s^v) = \begin{array}{c} \begin{array}{cccc} N & E & S & W \end{array} \\ \begin{bmatrix} 0 & .67 & .67 & .67 \\ .33 & .17 & .17 & .17 \\ .66 & .17 & .17 & .17 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} \\ \text{score 0} \\ \text{score 1} \\ \text{score 2} \\ \text{victim} \end{array} \tag{3-57}$$

This matrix shows what the agent expects to observe to its north, for different locations of the victim (columns). Now, starting from a uniform distribution on the victim's location, the scores change the beliefs in the following manner. Directions with a score of 0 will result in complete certainty that the victim is not there. Directions with a score of 2 have twice the belief compared to directions with a score of 1. This is shown in Figure 3-10

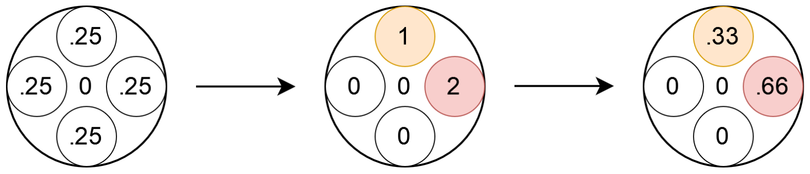


Figure 3-10: The initial belief on the victim's location (L), the observed scores after obtaining the terrain map (M) and the updated belief (R).

Now, another node will be added to the graph. In the initial situation the belief on the victim being in any direction will be 0.125. Table 3-9 shows how the belief for all eight directions changes based on different observations made at node 0.

Table 3-9: Shows the updated beliefs on the victim's location after the terrain map scores are observed. Most left columns show the scores obtained in every direction. Most right column shows the total belief of the victim being located in node 1.

N	**E**	**S**	**W**	**0N**	**0E**	**0S**	**0W**	**1**
1	1	0	0	0.25	0.25	0	0	0.5
2	0	0	0	0.5	0	0	0	0.5
2	1	1	0	0.33	0.17	0.17	0	0.33
1	0	0	0	0.33	0	0	0	0.67

Note that observations with a score of 2 have twice the belief of those with a score of 1. An expected score per node of $\xi = 2$ was defined. For the third row of observations, a total score of 4 is found; thus, the probability of the victim being located in this node increases (to 4/6), and it being located in node 1 decreases (to 2/6). The same effect can be seen for the last observation, but then the other way around. Thus, ξ also indicated the exact score obtained from the terrain map for which the probability that a victim is within a node does not change.

3-4-1 Agent Orientation

The previous example shows how the beliefs on the location of the victim are updated using the terrain map. To identify the victim, the agent's front facing cameras are needed. Therefore, the agent's current orientation has to be included as a state factor. The state factor s^o for orientation is added, which has a total of n_d states. One state for every possible direction. The A-tensors remain largely unchanged, since regardless of which direction the agent is looking at, for all other directions the terrain map is still obtained; thus scores will still be found. The direction that the agent is looking in does not obtain a score. It only returns if the victim is observed or not. Now, the A-tensor can be defined by:

$$A\{o^{di}\}(o^{di}, s^{vj}, s^o) = \begin{bmatrix} q_0 & q_1 & \dots & q_{n_d} \end{bmatrix} \tag{3-58}$$

$$q_{z=j=i} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}, \qquad q_{z=i\neq j} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}, \qquad q_{z\neq i=j} = \begin{bmatrix} S_v \\ 0 \end{bmatrix}, \qquad q_{z\neq i\neq j} = \begin{bmatrix} S_e \\ 0 \end{bmatrix},$$

There are four situations; i represents the outcome modality, j the location of the victim and z the orientation of the agent. The first $z = j = i$ indicates that the agent is orientated in the direction where the victim is located and that it is considering the outcome modality of that same direction. In that case, the victim is observed. The second case $z = i \neq j$ is the same; however, the victim is not located in that direction. Thus, no victim will be observed, which is the same as observing the score 0 from the terrain map, since both rule out the presence of the victim. The third situation $z = i \neq j$ means that the victim is located in the same direction as the corresponding outcome modality; however, the agent is orientated in another direction (so the cameras point elsewhere). In this case, S_v is expected as was the case without taking orientation into account. The last situation $z \neq i \neq j$ means that the direction corresponding to the outcome modality does not host the victim, and the agent is orientated in another direction. This is represented by S_e. To be able to change directions, a new set of n_d actions is added. Every action corresponds to moving to a certain direction. The B-tensor then has the following form:

$$B\{s^o\}(s_{\tau+1}, s_\tau, a_i) = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n_d} \end{bmatrix}, \qquad \beta_i = \mathbb{1}^{1\times n_d}, \qquad \beta_{j\neq i} = \mathbf{0}^{1\times n_d} \tag{3-59}$$

Thus, for the action: *change to direction* i, the matrix (layer) corresponding to action a_i consists of zeros and a row of ones in row i. That is, the agent will always end up in the orientation corresponding to the action taken, regardless of the initial orientation.

Example: Orientation

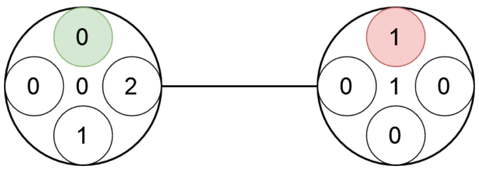


Figure 3-11: Example graph, green indicates the agent's intial position, red the location of the victim. Numbers indicate scores.

Now, let us look at the example given in Figure 3-11. The agent will start at node 0, looking north, and the victim is located at node 1 north. The EFE contributions and expected observations for different plans are given in Table 3-10. The expected observations are given as the expected score. For the direction in which the agent is pointed, a percentage is displayed, showing the probability of observing the victim. Table 3-11 shows the updated belief after the actions have taken place. The first observation will be made located at node $0N$, this observation forms the initial belief.

Table 3-10: Shows the EFE for different plans of length 1 and 2. The right halve shows the expected observations where decimal numbers indicate the expected scores, and percentages show the probability of observing the victim.

length	plan	SIG	UT	-EFE	exp. N	exp. E	exp. S	exp. W
p=1	0E	0.84	-6.57	-5.73	0.5	40%	0.73	0.5
	0S	0.76	-6.77	-6.01	0.5	0.97	20%	0.5
	1N	0.63	-6.87	-6.24	10%	0.62	0.62	0.62
	1E	0.63	-6.87	-6.24	0.62	10%	0.62	0.62
p=2	0E - 0S	1.6	-13.35	-11.75				
	1E - 1S	1.27	-13.75	-12.48				

Note how both utility and state information gain are higher for moving in directions with higher scores. This happens because the chance on a reward is greater and the probability of getting full certainty on the victim's state is larger as well. Important is to note that there is no difference between the actions moving to node 1N or 1E, or to any of the directions in node 1 for that matter. This will be discussed in the next section.

Table 3-11: Shows the belief in the victim's location after an action has taken place. Action 0N is the initial action and thus also the prior belief for all the other actions.

action	0N	0E	0S	0W	1N	1E	1S	1W
0N	0	0.2	0.4	0	0.1	0.1	0.1	0.1
0E	0	0	0.33	0	0.17	0.17	0.17	0.17
0S	0	0.5	0	0	0.125	0.125	0.125	0.125
1N	0	0	0	0	1	0	0	0
1E	0	0.5	0.25	0	0.25	0	0	0

3-4-2 Terrain Map for Unvisited Nodes

In the previous example, there is no difference in EFE for plans $1E - 1N$ and $1S - 1W$. This happens because the chance of obtaining a certain score, or the victim being present, is equal for every direction in node 1. However, once node 1 has been visited and the terrain map has been obtained, when deciding to look in another direction, this will always be the direction with the highest score obtained. Therefore, it makes sense to look at policies such as $0S - 1E - \Theta_1$ where Θ_1 encodes looking in the best direction once the terrain map of node 1 has been obtained. To do so, the expected oberservations need to be ordered from best to worst. This can be done by the binomial distribution, which has been used before in Section 3-3-2. Here, p_S is the probability of obtaining any score, which is equal to any entry of X from Equation 3-53.

$$p(x = i) = \mathrm{nCr} \cdot p_S^i \cdot (1 - p_S)^{n_d - i} \tag{3-60}$$

Here $p(x = i)$ is the probability that if the agent obtains the terrain map for n_d directions, that i of these directions contain a certain score. The chance that at least i directions contain a certain score can be calculated using:

$$p(x \geq i) = 1 - \sum_{j=0}^{i-1} p(x = j) \tag{3-61}$$

Now, the chance that the best direction contains the best score is represented by $p(x \geq 1)$. The chance that the second-best direction contains the best score is represented by $p(x \geq 2)$, and so on. If a direction does not obtain the best score, the probability that this direction contains the next best score can be calculated by:

$$p^{\text{2nd}} = (1 - p(x \geq i))\tilde{p}(x \geq i) \tag{3-62}$$

That is, the probability that the direction does not obtain the best score multiplied by the adjusted probability that it will obtain another score. This adjusted probability $\tilde{p}(x \geq i)$ uses an adjusted $\tilde{p}_S = p_S/(1 - p_S)$ since the total possible number of scores has decreased. In this way, all probabilities to obtain the different scores can be calculated incrementally and directions can be ordered from best to worst.

Upon arriving at a node, the agent will be pointed in some direction. This direction will be indicated by Θ_A for arrival. Now, a total of $n_d - 1$ directions must be ordered from best to worst. Next, the ordering has to be integrated into the expected observations given the states (A-tensor). Since the initial belief will be higher for the victim to be located in the best location compared to the worst. The belief will be divided on the basis of the expected scores and can be found by:

$$D\{s^v\}_u = \begin{bmatrix} \frac{1}{n_d} & \frac{(S_\Theta T)^\intercal}{\xi} \end{bmatrix} \tag{3-63}$$

Here, $D\{s^v\}_u$ represents the belief in the location of the victims for an unvisited node. The first entry is just the uniform belief, which corresponds to the arival direction. The matrix S_Θ is the matrix that describes the distribution of the scores from the best direction to the worst direction. The column vector $S_\Theta T$ represents the expected scores for the ordered directions. This process is clarified in the next example.

Example: Terrain Map with Unvisited Nodes

This example uses the same situation as the previous example, but takes order for unvisited nodes into account. The distribution over scores ordered from best to worst is:

$$S_\Theta = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{ccc|l} s_0 & s_1 & s_2 & \\ 0.30 & 0.28 & 0.42 & \text{best direction} \\ 0.74 & 0.19 & 0.07 & \text{second direction} \\ 0.96 & 0.03 & 0.00 & \text{worst direction} \end{array} \tag{3-64}$$

From this, the expected scores for the ordered directions can be found:

$$S_\Theta T = \begin{bmatrix} 1.125 \\ 0.333 \\ 0.042 \end{bmatrix} \begin{array}{l} \text{best direction} \\ \text{second direction} \\ \text{worst direction} \end{array} \tag{3-65}$$

Note that with an expected score at arrival of $\xi/n_d = 0.5$, the total expected score of the node sums up to 2 (ξ). The beliefs for the ordered directions can be found by $S_\Theta T/\xi$, which yields the initial beliefs on the victim's location:

$$D\{s^v\} = \begin{array}{cccccccc} \mathit{0N} & \mathit{0E} & \mathit{0S} & \mathit{0W} & \mathit{1\Theta_A} & \mathit{1\Theta_1} & \mathit{1\Theta_2} & \mathit{1\Theta_3} \\ [.125 & .125 & .125 & .125 & .125 & .28 & .08 & .01] \end{array} \tag{3-66}$$

Next, the expected observations will be changed, in case the victim is not present, accordingly:

$$S_{e\Theta_1} = \begin{bmatrix} .25 & .375 & .375 \end{bmatrix} \tag{3-67}$$

$$S_{e\Theta_2} = \begin{bmatrix} .78 & .11 & .11 \end{bmatrix} \tag{3-68}$$

$$S_{e\Theta_3} = \begin{bmatrix} .97 & .01 & .01 \end{bmatrix} \tag{3-69}$$

Now, let us again look at the EFE and beliefs for different plans. The results are given in Table 3-12.

Table 3-12: Shows the EFE for different plans of length 1 and 2. The second node is now ordered from best (Θ_1) to worst (Θ_3). Delta Δ shows the difference in EFE compared to the situation where the second node is not being ordered.

length	plan	SIG	UT	-EFE	Δ	exp. Θ_A	exp. Θ_1	exp. Θ_2	exp. Θ_3
p=1	$1\Theta_A$	0.51	-6.87	-6.36	-0.12	10%	1.25	0.33	0.04
p=2	$1\Theta_A$ - $1\Theta_1$	1.27	-13.62	-12.35	0.13	0.62	23%	0.42	0.05
	$1\Theta_A$ - $1\Theta_2$	0.95	-13.78	-12.83	-0.35	0.62	1.25	7%	0.05
	$1\Theta_A$ - $1\Theta_3$	0.82	-13.84	-13.02	-0.57	0.62	1.25	0.42	1%

Notice that the utility for moving to node $1\Theta_A$ corresponds to any of the directions of node 1 that have been seen before in Table 3-10. The state information gain is lower since knowledge on the distribution over scores, and thus the belief on the victim's location, is already taken into account. Now, after obtaining the terrain map, different plans can be considered. Note that in this case, the EFE for looking in the best direction is lower compared to the two-step policies of the previous table. However, the second and third best directions are higher; thus worse. This also becomes clear by looking at the chance to observe the victim, which is more than twice as high for looking into the best direction, compared to any given direction when ordering of the scores is not taken into account.

3-4-3 Terrain Map with Frontiers

This implementation of the terrain map can be easily translated into the earlier work on uncertainty over nodes. To do so, adjustments are needed for the reduced belief in s^v. This is done by replacing the expected scores by the adjusted scores $S_\Theta \phi_F(x)$ with ϕ_F as introduced by Equation 3-45 and x being the relevant uncertain node. This concludes the active inference framework for graphs. The next section will conclude the framework in a larger-scale example. The next chapter will describe a simulation environment in which an agent explores an unknown room. It will show how the graph can be constructed online, and active inference will be used to find the best plans to traverse this graph.

3-5 Example: Search of a Room

In this section, the active inference framework will be put to the test by a larger scale problem. The agent has to conduct a search mission in a room. Figure 3-12 shows a rendering of the room in question.

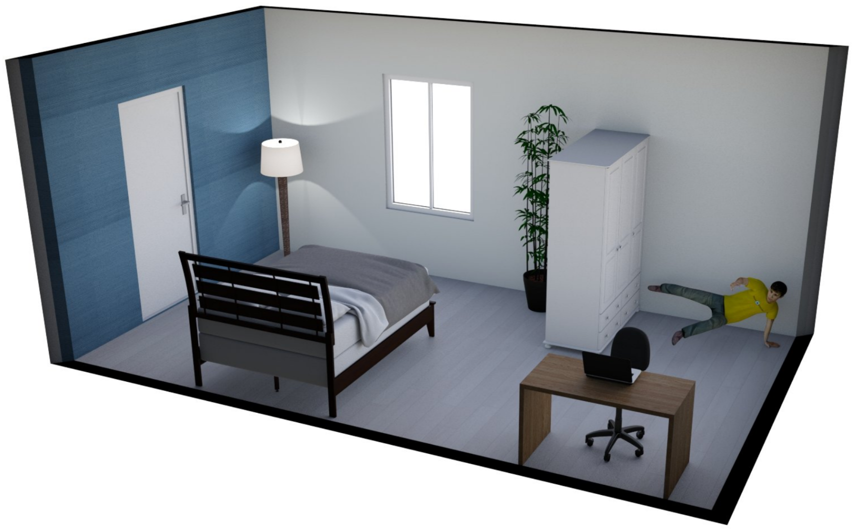


Figure 3-12: Render of an example room which will be explored using the active inference framework.

For this room, the final graph will be defined beforehand, as well as the scores that will be obtained. The agent itself is not aware of the graph's size, structure or any scores beforehand. For this example, there is a total of 8 nodes, which will also be μ. The standard deviation σ is chosen to be 0.75. This gives a total possible number of 10 nodes in the graph. There are three possible scores and four directions n_d per node. This gives us a total of 10 agent states and 40 victim states. Later the influence of the expected score ξ on the EFE calculations will be discussed. For now, $\xi = 2$ is used, which is also the average for the final graph. Possible scores are (0), (1) and (2). The necessary vectors are as follows:

$$T = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}^\intercal, \qquad S_v = \begin{bmatrix} 0 & .33 & .66 \end{bmatrix}^\intercal \tag{3-70}$$

$$S_e = \begin{bmatrix} .67 & .17 & .17 \end{bmatrix}^\intercal$$

This is equal to the example in the previous section. The corresponding expected scores are thus:

$$S_\Theta T = \begin{bmatrix} 1.125 \\ 0.333 \\ 0.042 \end{bmatrix} \begin{matrix} \text{best direction} \\ \text{second direction} \\ \text{worst direction} \end{matrix} \tag{3-71}$$

Now, with the chosen mean and standard deviation on the final graph size, the belief on the victim's location can be found. First, let us find the probability that a node exists:

$$\phi = \begin{bmatrix} 1 & 1 & 1 & | & 1 & 1 & 1 & .98 & .77 & .23 & .02 \end{bmatrix} \tag{3-72}$$

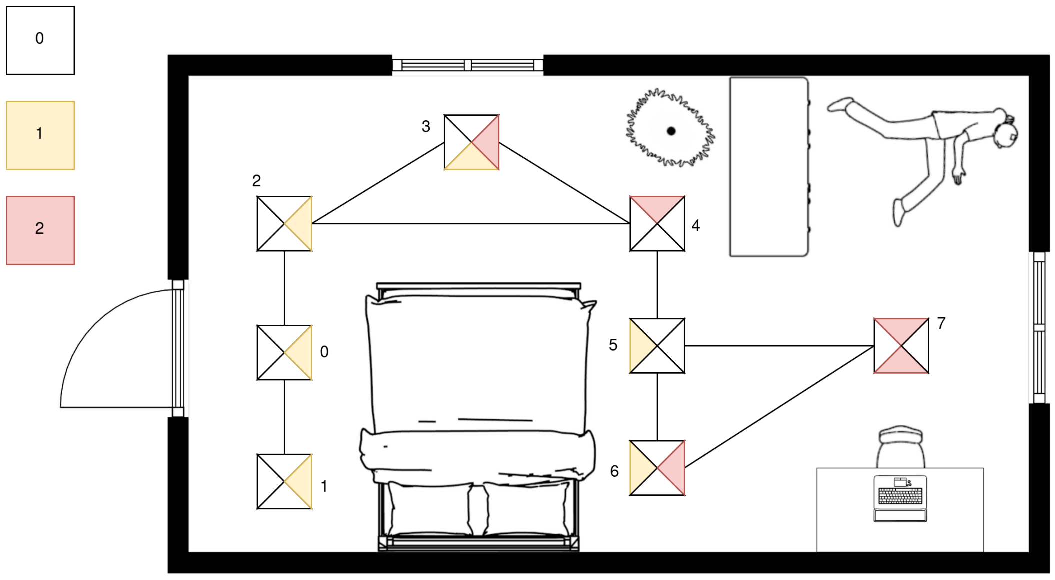


Figure 3-13: Floor plan of an example room. The final graph indicated including the scores that will be observed by the agent once visiting a node for the first time. The victim is located in node 7N.

The initial graph will be what the agent can observe from node 0, thus there will already be two frontiers (node 1 and node 2). The probability of a node being accessible from a frontier then becomes:

$$\phi_F = \begin{bmatrix} 1 & 1 & 1 & | & .96 & .80 & .49 & .19 & .04 & .00 & .00 \end{bmatrix} \tag{3-73}$$

Taking orientation into account, this results in the following initial belief on the victim's location:

$$D^*\{s^v\} = \begin{array}{c} \begin{array}{cccccccccc} \mathrm{n}_0 & \mathrm{n}_1 & \mathrm{n}_2 & \mathrm{n}_3 & \mathrm{n}_4 & \mathrm{n}_5 & \mathrm{n}_6 & \mathrm{n}_7 & \mathrm{n}_8 & \mathrm{n}_9 \end{array} \\ \begin{bmatrix} .03 & .03 & .03 & .03 & .02 & .02 & .01 & .00 & .00 & .00 \\ .03 & .07 & .07 & .07 & .06 & .03 & .01 & .00 & .00 & .00 \\ .03 & .02 & .02 & .02 & .02 & .01 & .00 & .00 & .00 & .00 \\ .03 & .00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \end{bmatrix} \begin{array}{l} \mathrm{N} / \Theta_A \\ \mathrm{E} / \Theta_1 \\ \mathrm{S} / \Theta_2 \\ \mathrm{W} / \Theta_3 \end{array} \end{array} \tag{3-74}$$

Here $D^*\{s_v\}$ indicates a special matrix form of $D\{s_v\}$ where the entries are ordered in columns by node, which is used for clarity. Note that this already includes the frontiers. The dummy node is not included but would have a value of 0.31. Next, active inference will be used to solve the problem. There are moments for which the EFE for two plans are equal (for example, in the beginning, where there are two frontiers to choose from). A preference for node 2 is given. After that, the plans will be ordered by node number, from low to high, starting at the first action of the plan (thus, plan 2-3-4 has preference over 2-4-3 or 2-4-5).

3-5-1 Plan Construction

In previous sections, a small number of pre-specified plans were compared. In this and next sections, all possible plans are taken into account. Within this framework, some techniques have been introduced that need corresponding rules. The following four rules must be followed during the construction of candidate plans:

- **Duplicate steps are not allowed**.
  During utility calculations, locations that have high victim probability will return high utility for the whole plan. To account for this effect, and since in reality looking into the same direction twice within a plan would not add any new value, every action within a plan has to be unique.
- **An unknown part of the graph can only be exited via the frontier it was entered**.
  This is conform the rules introduced in Section 3-3. Since the same nodes are used for different paths, paths cannot be switched halfway.
- **A new node is always entered in the same direction (Θ_A)**.
  Since upon arrival in a new node, the direction are not yet ordered, a node is always entered in direction Θ_A. After a new node has been entered once within a plan, the ordering is known, and the agent can take any of the other orientations.
- **Only transitions conform the graph are possible**.
  This is needed since actions that are not possible to take will lead to the agent staying in place, which can have a positive effect on the EFE if that location yields high utility. This rule also greatly reduces the number of possible candidate plans.

3-5-2 Total Algorithm

Now let us have a look at what the total active inference approach looks like and what steps must be taken. The total algorithm is an iteration over the following steps:

1. **Graph update** (or initialisation)
   Consists of adding the new nodes to the graph, updating the known nodes, frontier nodes, and unknown nodes. Updates the victim's belief by changes due to added or removed frontiers, and changes in belief on total graph size. Updates the B-tensor conform to the new adjecancy matrix.
2. **Observations**
   The inference step that changes the belief in victim's location based on new terrain map scores and camera verification.
3. **EFE calculations**
   Construction of all possible plans and calculation of the corresponding EFE values. Takes the plan with the lowest EFE and returns the first action of the plan as the next action to be taken.
4. **Taking action**
   Updates the belief on the agent's location and orientation based on the new actions found by EFE calculations. Only the first action of a plan is taken.
5. **Repeat**
   Itterate over the steps above untill the finishing condition has been reached. For this simulation, this happens when the victim is found.

3-5-3 Results

All the results from this example are on page 56. The best plans found during each execution step are given in Table 3-13. The arrival direction Θ_A is based on the geography of the graph, and the agent's previous location. For example, in the first step the agent moves from node 0 to node 1. If this were done by walking forward (which is the assumption here), the agent would end up at node 1 pointing north. During the state updates, this is regarded as the arrival direction. If the direction in which the agent will arrive is not yet known, it is stated as direction Θ_A in the table. The order in which the actions of a plan are executed does not influence the EFE. For all plans, the ordering is adjusted such that looking into directions with a score of 2 get priority. Then moving towards new nodes, and after that any other actions.

In general the agent tends to go to new frontiers. This is not the case if a score of 2 is observed, then the agent stays within the node to verify that direction first. Table 3-14 shows a colour map with the belief in the location of the victim in every direction during each execution step. Some notable moments are the arrival in node 2 (step 1). At that moment there are three active frontiers (nodes 1, 3 and 4). Plans that just contain frontier nodes are prioritized, since they have no uncertainty of existing. Upon arriving at node 3 there are only 2 frontiers left (nodes 1 and 4). However, since the existence of node 5 is not yet known, this is the only moment where the agent prioritises going back to already visited nodes to look into directions that are not yet confirmed. Upon arrival at node 4 there is certainty on the existence of node 5 and plans going towards this node get prioritised again. Upon arriving at node 7 the only frontier left is node 1. Notice that the belief of the dummy node becomes 0 at the moment that there is only 1 frontier left, since it is no longer of need.

The behaviour is influenced by how many objects are expected in a room. In very crowded rooms, the agent tends to skip objects that have a low score. Table 3-15 shows that for a lower value of ξ the agent will only continue after all directions within every node have been ruled out. Instead of using ξ as a prediction on how cluttered the room is, one can use it, together with the number of possible scores T, as parameters that changes the trade-off between exploring (obtaining more terrain maps from new nodes) and exploiting (ruling out or confirming victim's pressence).

Table 3-13: Overview of best found policies and corresponding expected free energy for every step during execution.

Step	Node	Best Policy	-EFE
0	0E	2N $\rightarrow 2\Theta_1 \rightarrow 3\Theta_A \rightarrow 3\Theta_1$	-26,50
1	2N	3E $\rightarrow 3\Theta_1 \rightarrow 4\Theta_A \rightarrow 4\Theta_1$	-26,39
2	3E	4E $\rightarrow 4\Theta_1 \rightarrow 2E \rightarrow 3S$	-26,35
3	4E	4N $\rightarrow 5S \rightarrow 5\Theta_1 \rightarrow 6\Theta_A$	-25,86
4	5S	6S $\rightarrow 6\Theta_1 \rightarrow 7\Theta_A \rightarrow 7\Theta_1$	-26,28
5	6S	6E $\rightarrow 7\Theta_A \rightarrow 7\Theta_1 \rightarrow 6W$	-26,2
6	6E	7E $\rightarrow 7\Theta_1 \rightarrow 7\Theta_2 \rightarrow 6W$	-26,82
7	7E	7N $\rightarrow 7S \rightarrow 5W \rightarrow 6W$	-25,24
8	7N	victim found	

Table 3-14: Colormap where the saturation of the colour indicates how strong the belief is in the victim's location. Columns indicate nodes and directions (N,E,S,W). Every row indicates a time step during the simulation.

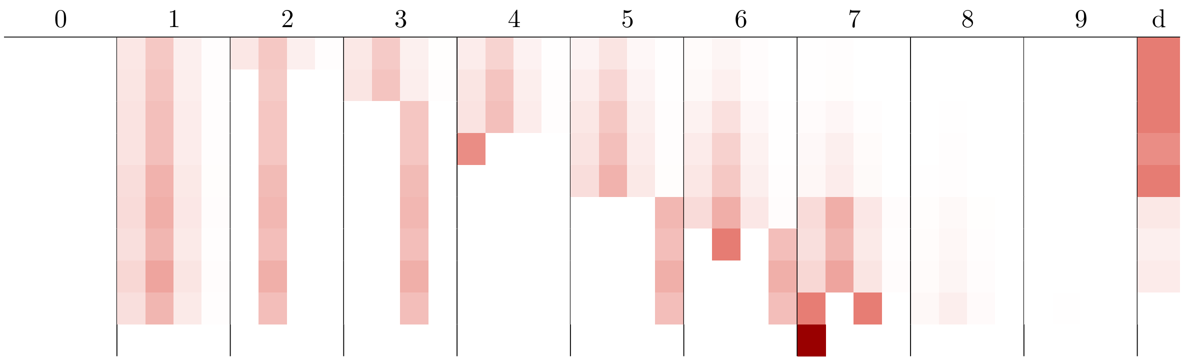


Table 3-15: Different plans taken by different expected scores per tile. Note that for the final path taken with $\xi = 1$ directions are collected per node.

ξ	Final Path Taken	Steps
1	0E $\rightarrow 2NE \rightarrow 3ES \rightarrow 4EN \rightarrow 5SW \rightarrow 6SEW \rightarrow 7EN$	13
2	0E $\rightarrow 2N \rightarrow 3E \rightarrow 4E \rightarrow 4N \rightarrow 5S \rightarrow 6S \rightarrow 6E \rightarrow 7E \rightarrow 7N$	9

Chapter 4

Active Inference with Spot

In this chapter, the active inference framework obtained in Chapter 3 will be applied to Spot. The first section contains an overview of the algorithm and the several techniques needed to construct a graph online. The chapter is finalised with a simulation of a search and rescue mission in a room.

4-1 Algorithm Overview

This Chapter focuses on a simulation conducted in a fictive room. A render of the room is shown in Figure 4-1. The victim is located next to the coach on the floor. The agent will start in the lower left corner (opposite the chess board).

Figure 4-2 shows a schematic overview of the algorithm. For simulation environments, the blue parts are used (and the green part is not used). For real-world implementations, the green part substitutes for the blue part. White boxes are always used. The orange box shows the active inference algorithm which has been explained in Chapter 3. The boxes ‘map image’ and ‘real world’ are the sources of the algorithm.

The focus of this section is to describe the general workings of the seperate boxes. Often straightforward and simple solutions will be used. The aim is not to find the best implementation but merely to describe the framework. Sometimes, possible solutions or literature that could be used for improved implementation are mentioned.

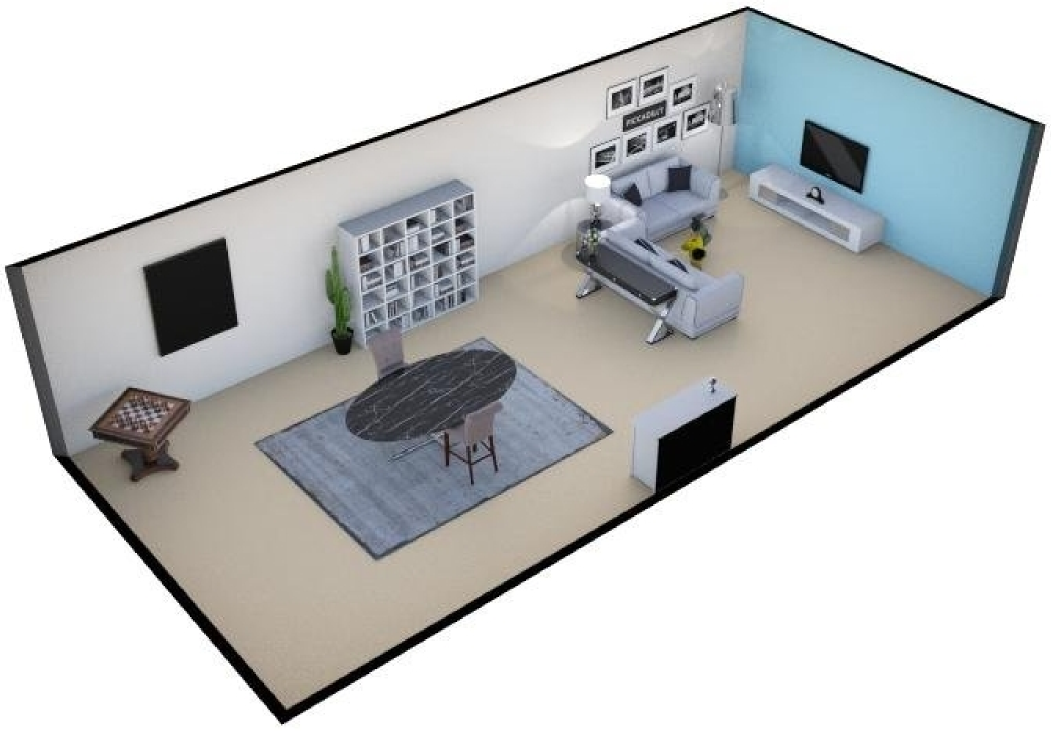


Figure 4-1: Render of the room used for the search and rescue mission.

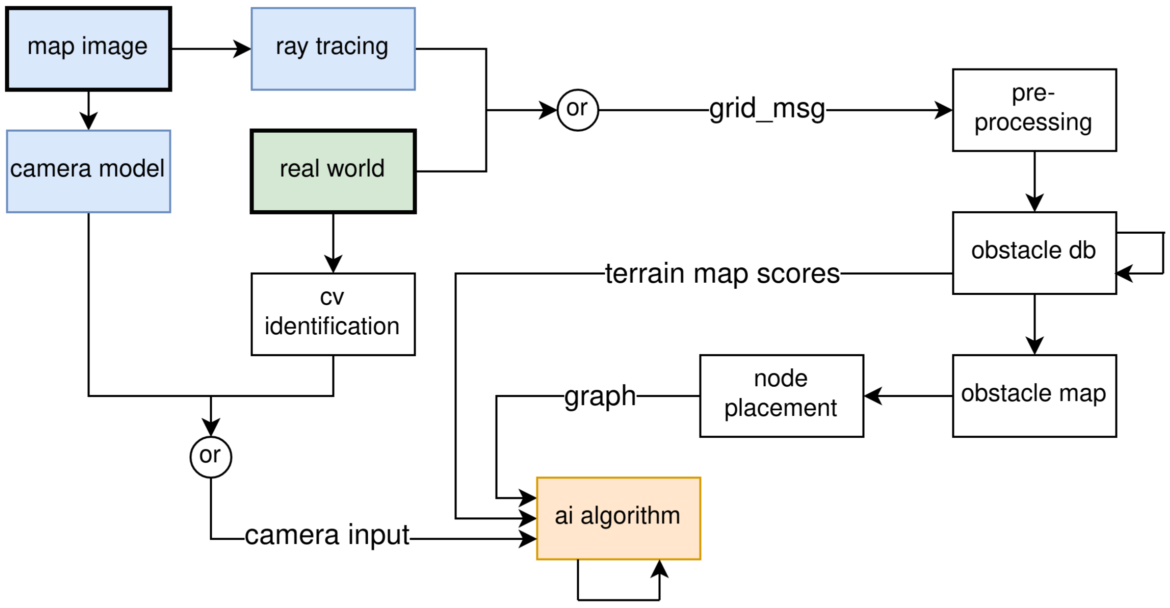


Figure 4-2: Overview of the final algortihm for simulations (blue boxes) and real implementations (green boxes).

4-1-1 Map Image

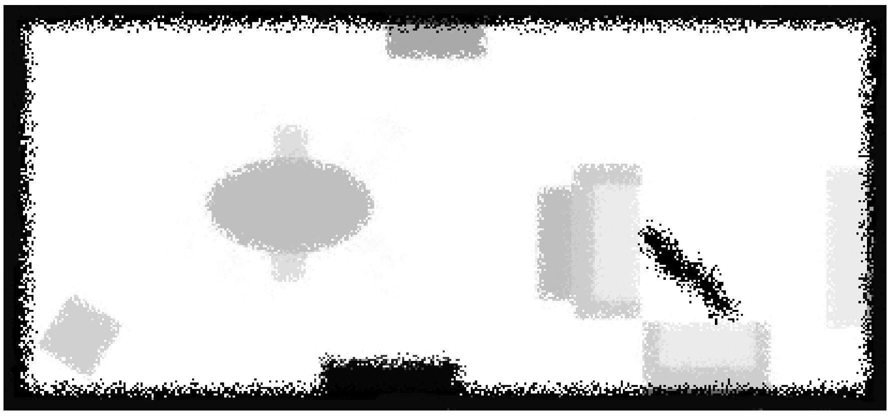


Figure 4-3: Map image of the room (mirrored compared to render).

In Chapter 2 the two inputs that Spot can use during execution have been discussed. One is the camera input. The camera input is a binary value and only indicates if a person is located within the agent's current field of view. The other is the terrain map, which is formed from a grid message. The grid message is a ROS-message (Robotic Operating System) given in the form of a list:

$$G_{msg} = \begin{bmatrix} x_0 & y_0 & z_0 & x_1 & y_1 & z_1 \ldots x_q & y_q & z_q \end{bmatrix} \tag{4-1}$$

With q the total number of tiles in the terrain map, which in Spot's situation is equal to 128^2 at every node. For the simulation, a source had to be found that contains the necessary sensor information. The decision was made to use an image that will be referred to as the 'map image'. The map image is a JPEG greyscale image of the room. The coordinates of the pixels of the image correspond to the x and y coordinates of G_{msg}. The greyscale indicates the height or z-value. The greyscale of the grid image is scaled between 1 at maximum height $h_{\max}$ and 255 at minimum height $h_{\min}$. The location of the victim corresponds to a greyscale value of 0. This makes that all necesary sensor inputs can be derived from just one file. The translation from greyscale to object height can be made using:

$$h_{gs} = 255 - \frac{(h_{max} - h_{min})}{255 - 1} h \tag{4-2}$$

With h_{gs} being the greyscale value corresponding to height h. A quick example of the scaling is given in Table 4-1 where $h_{\max} = 150$ and $h_{\min} = 0$ are used. For the example room shown in Figure 4-1 the corresponding map image is given in Figure 4-3. For this image, a diffusion layer has been used to represent noise.

Table 4-1: Translation from greyscale to height

greyscale	0	1	62	128	184	255
relative ($h_{\max}$)	victim's location	1	3/4	1/2	1/4	0
absolute (cm)		150	112.5	75	37.5	0

4-1-2 Ray Tracing and Grid Message

The terrain map as discused in Section 2-2, is created from the grid message obtained at a new node. This grid message needs to be recreated from the map image. This is done by ray-tracing and Bresenham's method. Both have been introduced in Section 2-3-2. Here the method will be adjusted conform a 2.5D grid representation. The algortihm stores the height values derived by the map image's greyscale values into a list conform Equation 4-1. A value will only be added to the message if it is above a certain threshold. Figure 4-4 illustrates this idea. The dotted line is the field of view beyond obstacle B. To acccount for noise in the graph image, and to ensure that obstacles are represented by more than one tile (or pixel), the actual threshold laggs behind by a couple of tiles. This is illustrated by the threshold line in the figure. For the simulation in this chapter a threshold lagg of 5 tiles has been used. Now, every object beyond B that lays above the threshold will be added to the grid message. These parts are represented by the blue areas. Once obstacle D is 'hit' the threshold will be updated to a slightly steeper incline, such that the new threshold is conform the coloured corner of obstacle D.

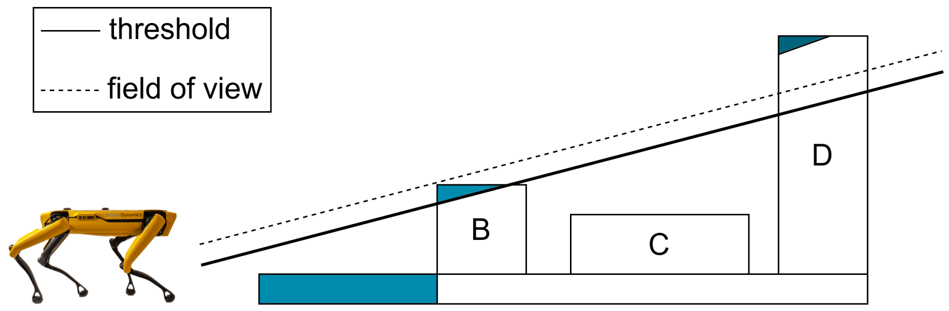


Figure 4-4: Illustration of parts added to the grid message (blue) with raytracing in 2.5D

After these values are taken all around the agent, conform the ray tracing method as explained in Section 2-3-2, the grid message is passed on to the pre-processing module after which it is stored in the obstacle database.

4-1-3 Obstacle Database

The obtained grid message is first transformed into a 2D array structure, with columns x, y, z representing the coordinates and height of the terrain map. Next, a belief value (b) is added to every row. The belief value represents how strong the trust in the measurement is. This belief value is conform a sensor model of the agent. Then, the current itteration step (i) or time is added as a separate column. Next, the coordinates will be translated to the global coordinates of the obstacle map. The initial coordinates of the agent at the start are the origin $(0, 0)$. By translating the coordinates a global database of all the obstacles in the map is obtained, including the belief in the obstacles themselves. In the simulation, any duplicates are removed. In real situations there are often contradicting values. This can happen due to drift, noise, sensor accuracy, or because the world has changed. In this case one could use the belief values and time indicator to select, or merge, the correct heights at every obstacle.

The final database has the form:

$$O_{db} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{b} & \mathbf{i} \end{bmatrix} \tag{4-3}$$

Now from this database the obstacle map O_m is constructed. The obstacle map is a 2D binary array representing a grid-tile world, for which hold:

$$O_m(x, y) = \begin{cases} 1, & \text{if } z(x, y) > \alpha_o \\ 0, & \text{otherwise} \end{cases} \tag{4-4}$$

With $z(x, y)$ indicating the z value in O_{db} corresponding to coordinates (x, y). The threshold is used to exclude small obstacles that can easily be walked over. In the simulation this threshold is set to 15cm.

4-1-4 Node Placement

The graph is created from the obstacle map. To do so, nodes are added based on information gain as conform Section 2-3. Figure 4-5 on page 62 shows an overview of the node placement algorithm. Node placement is only activated upon arriving in a node for the first time. Before node placement takes place, any already present, but not yet visited nodes, within a certain radius of the current node, are removed. This is done since those nodes have been placed at times when less information about their surroundings was available, thus might no longer be at the best location. Nodes that have been visited before will always remain part of the graph. After this, potential nodes are sampled at a radius r from the current node. It is also possible to sample nodes in a band of a certain width. Then for every node the euclidean distance to any other fixed node in the graph is calculated. If the distance is too short the node is neglected. This distance threshold is set to 100cm in this simulation. This is an easy first selection for nodes, whose potential gain would be very low since they would be very close to already discovered areas. After this, the potential gains for all the nodes in the list of potential frontiers are calculated. This is done in accordance with the information gain calculations as described in Section 2-3. Next, every node that is not conform a minimum potential gain requirement will be disregarded. For this simulation, only nodes that have a potential information gain of 55% of the maximum amount, are added. Next, the node with the highest information gain is added to the graph. The algorithm is repeated for the remainder of the list, taking the newly added node into account in the distance selection. The information gain of the remaining nodes does not need to be calculated again. Once there are no nodes left to be added, the node placement is finished.

Now the distance matrix between any nodes in the graph is updated. In this simulation, nodes that are within 2 metres of ane another can be linked. Bresenham's is used to find all the tiles in the obstacle map, linking the two nodes together. If none of these tiles contains an obstacle consistent with the obstacle map O_m then the nodes are connected and a link is placed. The newly updated graph will be passed on to the active inference algorithm.

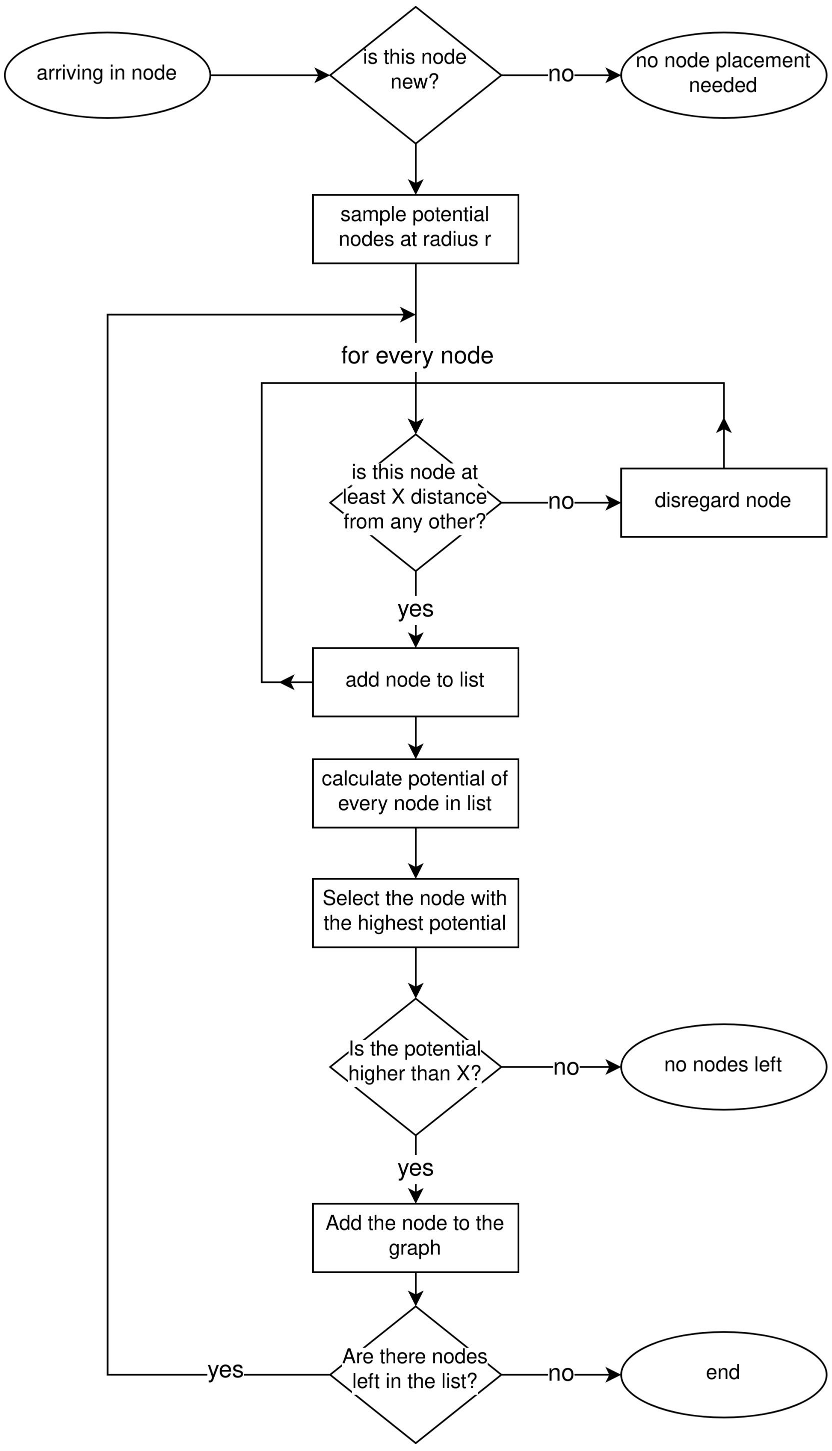

arriving in node
is this node new?
no
no node placement needed
sample potential nodes at radius r
for every node
is this node at least X distance from any other?
no
disregard node
yes
add node to list
calculate potential of every node in list
Select the node with the highest potential
Is the potential higher than X?
no
no nodes left
yes
Add the node to the graph
Are there nodes left in the list?
yes
no
end

Figure 4-5: Flow chart of the node placement algorithm.

4-1-5 Terrain Map Scores

From the obstacle database, the terrain map scores must be obtained for the active inference algorithm. A popular method by Zhou and Tuzel [37] describes how end-to-end learning can be used for feature detection and identification of objects using point clouds. These types of algorithms could be useful to determine the scores for the active inference framework. For this thesis, a simplified model is used. A lying victim is assumed. Any obstacle which has a height between 20 and 80 centimetres counts as 1 point per grid-tile. Now any direction which contains less than 60 points gets a score of 0. Any direction that has between 60 and 90 points gets a score of 1. Any direction with more than 90 points gets a score of 2. An example of how points are converted to direction scores is given in Table 4-2 on page 66 in the next section. This translation from terrain map to score is arbitrary and, in real scenarios, should be replaced by more meaningful methods such as the one mentioned above.

4-1-6 Victim Identification

In the simulation, as soon as the victim is within the field of view of the agent, thus if Bresenham's method encounters a greyscale of 0, the observation 'victim found' will be returned. If the victim is not within the field of view but is registered on the terrain map, an obstacle height of 45cm is returned. For real implementations computer vision to detect humans should be used to return the probability of a human being present within the agent's current field of view. Possible methods are for example by the use of cascades [38] or Histograms of Oriented Gradients (HOG) [39].

4-2 Simulation

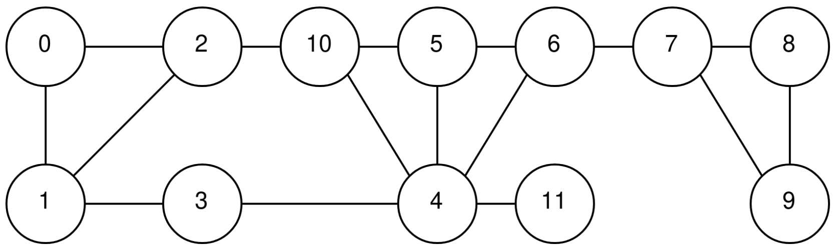


Figure 4-6: Final graph created when room is fully searched.

The total algorithm will be illustrated by a simulation run. The room as given in Figure 4-1 will be searched. During node-placement the numbering of nodes can change. To be consistent within the results of selected plans, a fixed final graph will be used. This graph is given in Figure 4-6 and represents the final graph as found by the entire room has been covered.

Figure 4-7 illustrates how the map building process takes place. First, the terrain map is created based on the grid message, either from Spot itself or generated from the map image. Based on the location of the agent and the number of directions, the scores are calculated. These will be used as input to the active inference algorithm such that inference on the victim's location can take place. The terrain map will be translated into the occupancy map. The occupancy map is then stitched to the global map. Node placement will take place on the global map by doing information gain calculations and taking node placement constraints into account. Then the active inference model is updated to this new graph and the EFE calculations can take place.

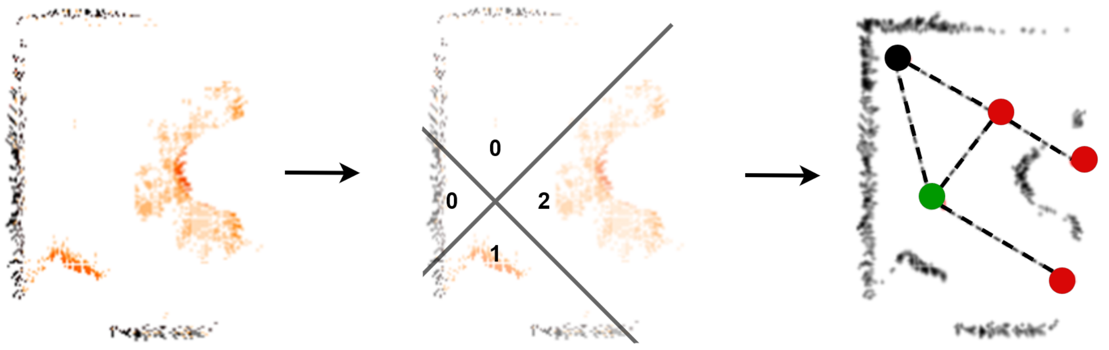


Figure 4-7: Steps taken at a new node: translate grid_msg to terrain map (L), find corresponding terrain map scores (M), translate terrain map to obstacle map, stitch to global map and place new nodes (R). Visited nodes are marked black, the current location of the agent is green, unvisited nodes are marked red.

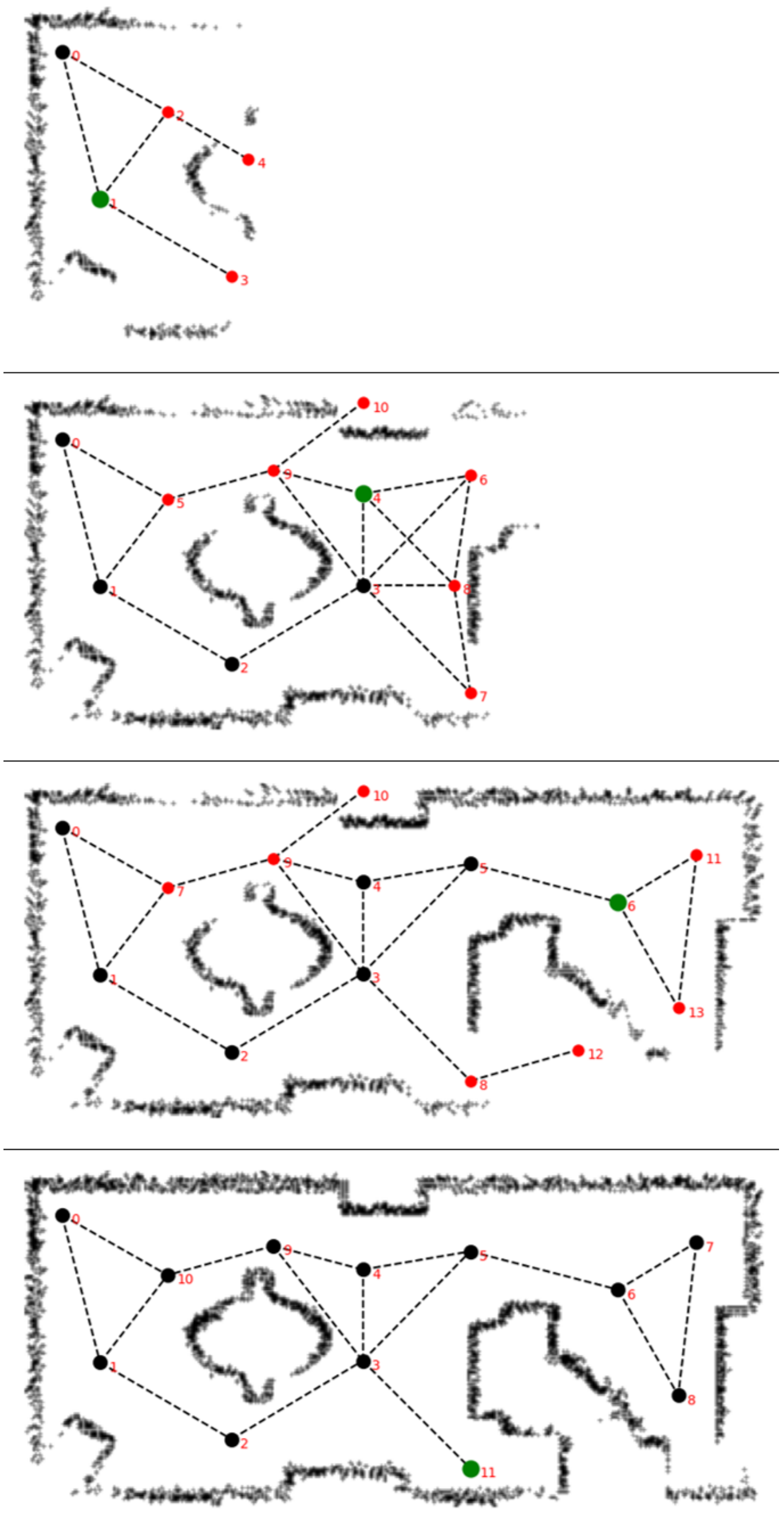


Figure 4-8: Shows the global map with nodes for different moments during execution. Black nodes are visited, green node is the current location, red nodes are frontiers (when linked to at least one black node).

Figure 4-8 shows how the global map changes during execution. The green nodes indicates the current location of the agent. The red nodes indicate unvisited nodes or candidate nodes. The black nodes are visited nodes. Red nodes can be removed, replaced and renumbered. Black and green nodes are fixed in place and their numbering does not change anymore. Together they form the known part of the adjecancy matrix. Node placement is done within the current frame of the global map. Therefore sometimes nodes are placed that eventually are not feasible such as node 4 in the first step and node 10 in the second and third step. These nodes however will not be regarded as frontiers. Only unvisited nodes that are directly linked to a visited node will be passed on as frontiers to the active inference model. Sometimes feasible nodes are removed because they no longer are conform the constraints of the node placement process. This can happen because another node that is more promising is too close or because the information gain from that node is not sufficient anymore. This is for example the case with node 8 at the second step. The victim will be found in the third image. The last image shows how the map would look like if exploration continued untill all nodes are visited.

The active inference simulation is similar to as shown in Section 3-5. For the final graph size a mean of $\mu = 14$ is used, with a standard deviation of $\sigma = 2$. The expected score per node is $\xi = 2$ which makes the A-tensors conform the ones used in Section 3-5. Eventually the victim will be found in node 7. At that time there are 12 nodes part of the graph, 7 visited nodes and 5 frontiers. Table 4-3 shows the belief on the victim's location during execution. For the results, only the beliefs over the first eight nodes are shown. Node 2 illustrates the belief of any of the frontiers, since they all have equal belief. The belief of the dummy is given by the actual value, since it is much higher than the seperate beliefs, especially in the beginning. A total of 18 nodes are modelled for this example, most of them are not shown in this colourmap. Table 4-4 shows the best policies and their corresponding EFE at every time step. An overview of the scores obtained by the terrain map at every node is given by Table 4-2. Every set of two columns shows the number of tiles that could possibly be a victim (L) and the corresponding score (R).

Table 4-2: Table that shows the number of tiles (t) between 20cm and 80cm for the seperate directions. The corresponding scores (s) are shown next to the number of tiles. The location of the victim is indicated by v.

node	E_t	E_s	N_t	N_s	W_t	W_s	S_t	S_s
0	37	0	2	0	0	0	2	0
1	122	2	2	0	1	0	88	1
3	5	0	102	2	69	1	0	0
4	115	2	27	0	113	2	0	0
5	105	2	22	0	104	2	81	1
6	38	0	9	0	80	1	101	2
7	121	2	3	0	104	2	v	2

Table 4-3: Colourmap where the saturation of the colour indicates how strong the belief is in the victim's location. Columns indicate nodes and directions (N,E,S,W). Every row indicates a time step during the simulation.

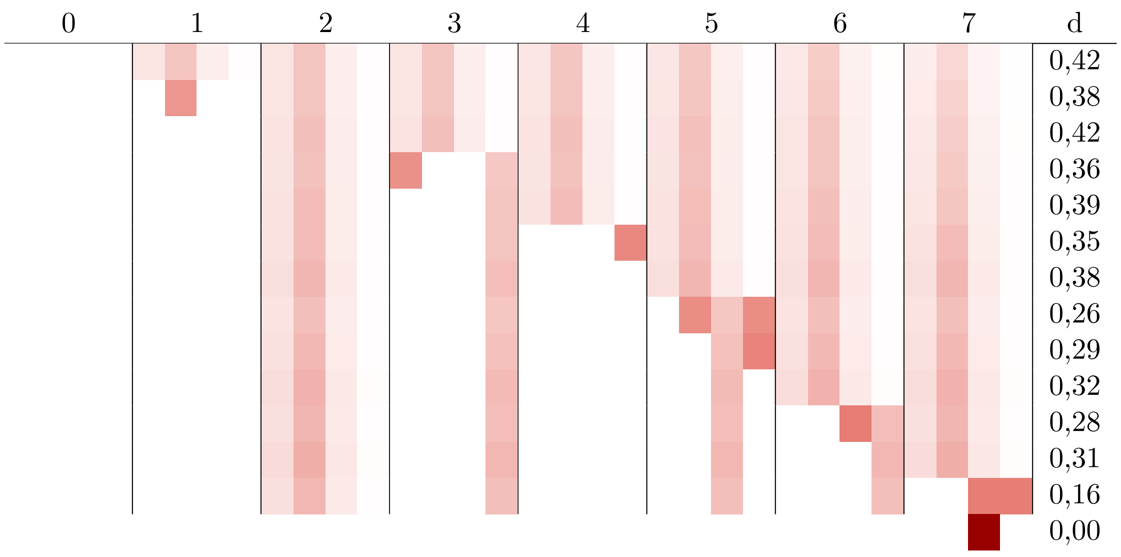


Table 4-4: Overview of best found policies and corresponding expected free energy for every step during execution.

Step	Node	Best Policy	-EFE
0	0E	1S $\rightarrow$ 1$\Theta_1$$\rightarrow$ 2Θ_A $\rightarrow$ 2Θ_1	-27,05
1	1S	1E $\rightarrow$ 2Θ_A $\rightarrow$ 3Θ_A $\rightarrow$ 3Θ_1	-26,97
2	1E	3E $\rightarrow$ 3Θ_1 $\rightarrow$ 4Θ_A $\rightarrow$ 4Θ_1	-26,99
3	3E	3N $\rightarrow$ 4E $\rightarrow$ 4Θ_1 $\rightarrow$ 5Θ_A	-26,65
4	3N	4E $\rightarrow$ 4Θ_1 $\rightarrow$ 5Θ_A $\rightarrow$ 5Θ_1	-26,96
5	4E	4W $\rightarrow$ 5Θ_A $\rightarrow$ 5Θ_1 $\rightarrow$ 6Θ_A	-26,87
6	4W	5N $\rightarrow$ 5Θ_1 $\rightarrow$ 6Θ_A $\rightarrow$ 6Θ_1	-26,88
7	5N	5E $\rightarrow$ 5W $\rightarrow$ 5S $\rightarrow$ 6Θ_A	-26,33
8	5E	5W $\rightarrow$ 5S $\rightarrow$ 6Θ_A $\rightarrow$ 6Θ_1	-26,64
9	5W	6E $\rightarrow$ 6Θ_1 $\rightarrow$ 7Θ_A $\rightarrow$ 7Θ_1	-26,84
10	6E	6S $\rightarrow$ 7Θ_A $\rightarrow$ 7Θ_1 $\rightarrow$ 6W	-26,59
11	6S	7E $\rightarrow$ 7Θ_1 $\rightarrow$ 6W $\rightarrow$ 5S	-26,83
12	7E	7S $\rightarrow$ 7W $\rightarrow$ 6W $\rightarrow$ 5S	-26,14
13	7S	victim found	

In general the results show the same kind of behaviour as seen in Section 3-5. The EFE contribution clearly improves once more nodes are discoverd. This corresponds to the colourmap, where the beliefs increase once more directions are ruled out. The belief in the dummy node generally decreases since the graph becomes larger and thus there are less undiscovered nodes left to be represented by the dummy. Upon arriving in node 7 the belief on the dummy is very small, note that at this time there are already 12 nodes part of the graph. Since the expected total graph size is $\mu = 14$, it is understandable that the contribution of the dummy node becomes much smaller.

Chapter 5

Conclusions

5-1 Summary

In this work, we have shown how entropy-based exploration and active inference can be combined for decision making in search and rescue applications.

In **Chapter 1** the concepts of coverage, exploration, and active inference were introduced. The main objective was stated *to find a method to let Spot search a room for victims.* The research questions aimed to combine exploratory and exploitative behaviour using active inference.

In **Chapter 2** the agent was introduced. It described what world representations are used and explained the concept of entropy. It showed how by the use of entropy calculations the agent can find a path that quickly covers large areas of unknown space. These techniques were later used in Chapter 4 to do node placement.

In **Chapter 3** active inference was introduced. An active inference framework for graph navigation and decision making was built. The first case showed how the agent and the victim can be represented by the nodes of the graph. It created the general framework to use EFE calculcations to explore and search. In the second case, uncertainty was added about the existence of nodes. By assuming a line-graph shape of unknown areas, EFE calculations could be made on parts of the graph that did not yet exist. The third case added scores based on the terrain map that can help distinguish the level of interest for different directions.

In **Chapter 4** the techniques of Chapter 2 have been combined with the active inference framework of Chapter 3. It described how the input generated by the agent can be used for node placement based on the expected entropy reduction. Then, the planning over the created graph was conducted by the active inference framework. This combination shows that, for an unknown room, active inference can be used to conduct search missions.

5-2 Answers to Research Questions

How can we use Spot to conduct information gain exploration?
By the use of Spot's terrain map we can place nodes in its direct surroundings. With Bresenham's algorithm we can calculate the potential entropy reduction for all of these nodes. By choosing the frontier with the highest potential we can explore the environment using information gain exploration.

How can we use active inference for decision making during search and rescue missions?
A graph representation can be translated into a POMDP generative model. The nodes represent the possible states for both the agent and the victim. By making assumptions on when the agent is able to observe the victim, the best plans can be selected that both reduce the uncertainty on the victim's location and improve the probability of finding the victim.

How can we use active inference with unknown environments?
By making an assumption on the total size of the graph and the structure of the unexplored parts of the graph, we can make plans over unexplored areas. By adjusting the probability of the victim being located in uncertain nodes includes this uncertainty in utility and state information gain calculations.

(How) does active inference relate to information gain exploration?
Since the states for the active inference framework (nodes) and the states for information gain exploration (grid tiles) are different, information gain exploration cannot be made part of the EFE formulation directly. However, node placement can still be based on information gain. In this way, the same behaviour on frontier selection can still be achieved. Active inference adds the possibility to include reward seeking behaviour, by the utility contribution of the EFE.

5-3 Discussion and Future Work

This part aims to give an overview of the problems within the current method, some ideas for improvements, and introduces explored ideas that have not been included in the final report.

5-3-1 Execution Problems

There are still some problems that come to light when using the framework with the exploration of rooms. One is the update for the belief adjustment ϕ_F. Which starts to give problems for graphs that are becoming unlikely large comapred to the initial guess on the total graph size. A solution might be to adjust the graph's size expectations during execution, such that these very unlikely discoveries cannot take place. There also is the problem that EFE contributions for actions for which we know in reality will not yield any rewards, still contribute. Going to a node in which it is likely to find the victim will then return very good EFE values for just staying in that node for all the steps in the plan. Also, the return of the best plan sometimes seems unlogical, this is for there is no penalty on movements. For example the plan $0N \rightarrow 0E \rightarrow 1N \rightarrow 1E$ has the same EFE as $0N \rightarrow 1N \rightarrow 0E \rightarrow 1N$. The first contains only one movement and two orientation changes. The other contains three movements and two orientation changes.

5-3-2 Undiscovered Nodes

The way uncertainty over undiscovered nodes is implemented in the framework is that the SIG and UT contributions of these nodes within plans are reduced. This is done by adjusting the probability of the victim. This means that the state that yields the reward (s^v) is needed to include the uncertainty on the graph. One can question if this method holds if there are multiple states with different rewards and uncertainties. A different model where we add the state of the agent and the possible actions as observations, and add the possible actions as a state factor as well, would allow us to infer over actions. This makes for a model in which we directly infer if we can move to a node based on its existence. This is closer to reality and makes for an active inference framework for graph navigation that is independent on different states or rewards. In the paper by Smith et al. [24] there is an example in which inference takes place on actions.

5-3-3 Model Expansion

The active inference graph navigation framework can be further expanded. We already introduced the terrain map and the camera inputs. However, more outcome modalities can be linked to the existing framework. For example, in the Snow usecase Spot is also equiped with microfones and a speaker. Deciding to take an action that lets the agent stay in place and uses its speaker to find any clues on the victim's pressence can be modeled by an expected return to the microphones if the victim is within a certain distance (links in graph) from the agent's location. Another expansion could be to take the overlap between terrain maps into account. Often terrain map information gathered at one node can also change the beliefs on neighbouring nodes that overlap with that terrain map. This would also make that nodes with small overlap automatically have higher state information gain contributions. This way, if there is no utility contribution we could mimic information gain exploration, entirely within the active inference framework.

5-3-4 Graph Characteristics

It may be interesting to see how certain graph characteristics relate to decisions made using the active inference framework. For example, situations with moving victims could change the importance of nodes with high betweenness centrality. The ability for the agent to record audio from possible victims could influence the importance of nodes with high closeness centrality. And using overlap or the ability to see victims in neighbouring nodes would expect to steer the agent to nodes with high degree centrality.

5-3-5 Scalability

In this report we only used plan lengths up to four steps. This is due to the exponential scaling of possible plans with the number of steps in a plan. Since in active inference we have to calculate through all the possible plans this places a limit on possible plan lengths. However, the nature of the search and rescue problems would not need us to compare all possible plans. Some plan selection criteria were already introduced. Further reducing the number of possible plans can greatly improve the execution speed.

5-3-6 Real World Implementation

The report is primarily aimed at a specific agent by Boston Dynamics. The aim was to create a framework that would translate to real-world implementations as easily as possible. Sadly however this has not been tested and making the framework work on the real agent would be a very rewarding next step.

Bibliography

[1] Wikipedia Contributors. R.u.r., Nov 2019. available at: https://en.wikipedia.org/wiki/R.U.R.

[2] Shakey the robot, Aug 2020. available at: https://www.sri.com/hoi/shakey-the-robot.

[3] Atlas - boston dynamics.
available at: https://www.bostondynamics.com/atlas.

[4] About spot: Boston dynamics.
available at: https://dev.bostondynamics.com/docs/concepts/about_spot.

[5] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. 2:1927–1933 vol.2, 2001.

[6] R.L. Graham and Pavol Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.

[7] Khouloud Eledlebi, Dymitr Ruta, Hanno Hildmann, Fabrice Saffre, Yousof Al-Hammadi, and Abdel F. Isakovic. Coverage and energy analysis of mobile sensor nodes in obstructed noisy indoor environment: A voronoi approach. *CoRR*, abs/2009.04864, 2020.

[8] Olimpiya Saha, Guohua Ren, Javad Heydari, Viswanath Ganapathy, and Mohak Shah. Deep reinforcement learning based online area covering autonomous robot. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 21–25, 2021.

[9] Olimpiya Saha, Viswanath Ganapathy, Javad Heydari, Guohua Ren, and Mohak Shah. Efficient coverage path planning in initially unknown environments using graph representation. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 1003–1010, 2021.

[10] Olimpiya Saha, Guohua Ren, Javad Heydari, Viswanath Ganapathy, and Mohak Shah. Online area covering robot in unknown dynamic environments. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 38–42, 2021.

[11] Howie Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Auton. Robots*, 9:247–253, 12 2000.

[12] Hoang Viet, Viet-Hung Dang, Md Laskar, and TaeChoong Chung. Ba: An online complete coverage algorithm for cleaning robots. *Applied Intelligence*, 39, 09 2013.

[13] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Intelligence/sigart Bulletin - SIGART*, 37:28–29, 12 1972.

[14] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005.

[16] Héctor H. González-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *I. J. Robotics Res.*, 21(10-11):829–848, 2002.

[17] Md. Al-Masrur Khan, Md Rashed Jaowad Khan, Abul Tooshil, Niloy Sikder, M. A. Parvez Mahmud, Abbas Z. Kouzani, and Abdullah-Al Nahid. A systematic review on reinforcement learning-based robotics within the last decade. *IEEE Access*, 8:176598–176623, 2020.

[18] Karl Friston. Friston, k.j.: The free-energy principle: a unified brain theory? nat. rev. neurosci. 11, 127-138. *Nature reviews. Neuroscience*, 11:127–38, 02 2010.

[19] Karl Friston, Spyridon Samothrakis, and Read Montague. Active inference and agency: optimal control without cost functions. *Biological cybernetics*, 106(8-9):523—541, October 2012.

[20] Catal, Ozan and Nauta, Johannes and Verbelen, Tim and Simoens, Pieter and Dhoedt, Bart. Bayesian policy selection using active inference. In *Workshop on "Structure Priors in Reinforcement Learning" at ICLR 2019 : proceedings*, page 9, 2019.

[21] Corrado Pezzato, Riccardo Ferrari, and Carlos Hernández Corbato. A novel adaptive controller for robot manipulators based on active inference. *IEEE Robotics and Automation Letters*, 5(2):2973–2980, 2020.

[22] Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas Fitzgerald, and Giovanni Pezzulo. Active inference and epistemic value. *Cognitive Neuroscience*, 6(4):187–214, 2015. PMID: 25689102.

[23] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: A synthesis. *Journal of Mathematical Psychology*, 99:102447, 2020.

[24] Ryan Smith, Karl Friston, and Christopher Whyte. A step-by-step tutorial on active inference and its application to empirical data, 01 2021.

[25] Casper Hesp, Ryan Smith, Thomas Parr, Micah Allen, Karl Friston, and Maxwell Ramstead. Deeply felt affect: The emergence of valence in deep active inference. 12 2019.

[26] Raphael Kaplan and Karl Friston. Planning and navigation as active inference. 12 2017.

[27] Jelle Bruineberg, Erik Rietveld, Thomas Parr, Leendert van Maanen, and Karl J Friston. Free-energy minimization in joint agent-environment systems: A niche construction perspective. *Journal of Theoretical Biology*, 455:161–178, 2018.

[28] Anderson Souza and Luiz Gonçalves. 2.5-dimensional grid mapping from stereo vision for robotic navigation. pages 39–44, 10 2012.

[29] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[30] Wolfgang Sodeur. Bavelas (1950): Communication patterns in task-oriented groups. *Schlusselwerke der Netzwerkforschung*, 2018.

[31] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[32] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. pages 65–72, 06 2005.

[33] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[34] Conor Heins, Beren Millidge, Daphne Demekas, Brennan Klein, Karl Friston, Iain D. Couzin, and Alexander Tschantz. pymdp: A python library for active inference in discrete state spaces. *Journal of Open Source Software*, 7(73):4098, may 2022.

[35] Oleg Solopchuk. Tutorial on active inference. *Published on Medium*, 2018.

[36] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John ODoherty, and Giovanni Pezzulo. Active inference and learning. *Neuroscience Biobehavioral Reviews*, 68:862–879, 2016.

[37] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[38] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[39] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.