Robotised Guard Tours in Security Systems

Johanna Häggström Wedding
Ella Thunborg

LUND UNIVERSITY
Department of Automatic Control

MSc Thesis
TFRT-6237
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2024

Abstract

With the growing market for enhanced security combined with recent advancements in robotic technology, the possibility of integrating these two fields is of considerable interest. While traditional surveillance relies on mounted cameras and scheduled guard tours by humans, the next step would be to use a robot to complement a person in such systems. This project explores the integration of autonomous robotics into security systems by developing and implementing a robotic guard tour using Boston Dynamics' Spot robot. An external thermal camera and computer vision algorithms were used to monitor the environment and respond to anomalies. Utilising Spot's existing capability, the primary objective was to evaluate the robot's ability to dynamically alter its pre-defined inspection route based on real-time sensor input, specifically the detection of anomalies.

Results, gathered at the Mechanical Engineering building at LTH and the Vipan construction site in Lund, show that the robot effectively adjusted its routes in response to detected anomalies, demonstrating enhanced surveillance capabilities. The findings in different settings—LTH's controlled environment versus the dynamically changing conditions at the construction site—proved that the robot could successfully conduct guard tours under both constant and variable conditions. The use of behaviour trees enabled decision-making and route management, suggesting potential improvements for future deployments to achieve more dynamic and autonomous surveillance operations. This thesis highlights the adaptability of autonomous robotic systems in complex and dynamic environments.

Acknowledgements

First, we would like to thank the Department of Automatic Control and the Robotics Lab at LTH for their support during our project. Observing the innovative work being carried out here has been truly inspiring, and we are grateful for the opportunity to conduct this project. We would also like to express our deepest gratitude to our academic supervisors, Björn Olofsson and Mathias Haage. To Björn, for consistently providing clear and actionable feedback on our work, helping us improve significantly, and to Mathias for all the hands-on help with Spot, ensuring we remained on track and completed our project successfully. Lastly, we would like to thank our examiner Karl-Erik Årzén for the valuable input during this project and for pointing us in the right direction when needed.

A special thank you goes to Fredrik at PEAB, who sacrificed numerous lunch breaks so that we could conduct tests with Spot at Vipan. We truly hope you get to sit down during your Friday lunches in the future.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

In 2024, video surveillance plays a crucial role in both public and private settings, with the majority of surveillance being provided by mounted cameras in public spaces such as stores, workplaces, airports, and residential areas. These surveillance systems serve multiple purposes, such as monitoring activities, preventing crimes, and ensuring safety [Mazzeo, 2023, p.1]. Public areas heavily rely on video surveillance for security, where cameras are placed to monitor crowds, detect suspicious behaviour and prevent criminal activities. Today, many customers of security systems combine intrusion alarm access control and video management systems with scheduled guard tours performed by trained security personnel. The purpose of such tours is to verify that no unauthorised persons are at the site and that the premises are in good shape. Examples of verification tasks include asking people for identification, making sure that doors are properly closed/locked, that fences are not tampered with, and that valuables are not stolen.

Recent advancements in video surveillance technologies have been enabled by the integration of machine learning, computer vision, and data analytics [Mazzeo, 2023, p.1]. Mazzeo states that these systems are capable of identifying a range of emergencies, from natural disasters to dangerous situation cased by humans. The real-time surveillance and analysis of recorded video data enable the assessment of security levels and the formulation of preventive strategies. The adoption of smart video surveillance has grown significantly in recent years. This growth can be attributed to a number of factors, including rising crime rates and global security concerns [Mazzeo, 2023, p.2]. The use of artificial intelligence and modern digital technologies has greatly increased the capacity, precision, and smart capabilities of video surveillance systems. One notable trend is the integration of the Internet of Things (IoT) and smart devices, including robots [Mazzeo, 2023, p.2]. Traditionally employed in industrial settings, robots are now being deployed in public spaces [IFR, 2021]. They collaborate with humans, enhancing security and safety. Companies,

employees, and the public benefit from this collaboration as robots assist in various tasks.

Previous Work

Several projects where robots are used for inspection on a construction site or in an industrial environment have been made. For example, the Master Thesis *Building dense reconstructions with SLAM and Spot* evaluated the use of Spot, the quadruped robot designed by Boston Dynamics, as a platform for site inspection [Nilsson, 2022]. The evaluation took place at the construction site Vipan, which is also the site for this project. Furthermore, Nilsson provides useful insights on how to operate on Spot. Politecnico di Torino conducted a Master Thesis, *Autonomous mobile robots: configuration of an automated inspection system,* in collaboration with the company Sprint Reply [De Santis, 2023]. This project used Spot to perform a routine inspection tour for a site in the oil and gas industry. The robot successfully reads the values of meters in the plant while following a prerecorded path, stopping on designated points to complete a reading. De Santis gives information on the use of autonomous robots in industrial environments, as well as how to execute a route on Spot without deviation from the recorded path.

Linköpings University recently published a Master Thesis, *Developing High-level Behaviours for the Boston Dynamics Spot Using Automated Planning,* conducted by Nisa Andersson [Andersson, 2023]. The thesis used an automated planner to navigate and manipulate objects in complex tasks. These tasks involve navigation to targeted positions, object recognition, and the execution of pick-and-place operations. The object is then finally delivered to a chosen position. Andersson implemented a delegation system designed by the Artificial Intelligence and Integrated Computer Systems (AIICS) division. This system is built upon a Task Specification Tree (TST) architecture, which formalises the decomposition of complex tasks for both single and multi-agent systems. Within this architecture, tasks are structured in a hierarchical tree format, where internal nodes represent control statements and external nodes represent elementary actions. Three principal node types are used: Control nodes, Interaction nodes, and Elementary Action nodes. Control nodes, for example, are critical for organising task sequences and managing the concurrent execution of actions, facilitating the coordination necessary for complex robotic behaviours. Interaction nodes handle the complexities of agent interactions, allowing multiple robots or systems to collaborate towards a common objective. Elementary Action nodes, as described by [Andersson, 2023], are the direct points of interaction between the robot and its environment. These nodes enable the robot to perform tasks like manipulating objects and navigating through spaces.

The delegation system integrated within the TST architecture adds a layer of complexity and autonomy to robotic operations. This system enables dynamic task al-

location across different agents, enhancing adaptability and scalability. It ensures that tasks are not only defined in a structured manner but are also delegated and managed across various components of the robotic system, enabling a robust and flexible response to operational demands [Andersson, 2023]. The work with the Boston Dynamics Spot robot provides a valuable case study on the potential of advanced robotic behaviours and planning systems for enhancing autonomous operations in real-world applications. The thesis by Andersson exemplifies how task specification and management systems can improve the functionality and efficiency of robotic systems.

1.2 Objectives

The goal of this project is to design and implement a method for defining a robotic guard tour that is executed on a construction site, while monitoring specific aspects along the route. This involves developing prototype software to control the robot along its path, using cameras and thermal cameras to assess the state of identified entities, and employing relevant computer vision algorithms. The steps included in this research are therefore further categorised into:

1. Define a robotic guard tour using the premade functionality of the robot. Investigate ways to externalise, programmatically generate, and execute routes.
2. Detect anomalies with the help of a mounted thermal camera and computer vision algorithms. Orient the external IR camera on the robot, ensuring it points in the correct direction during the path.

By combining the above steps, the final question that will be answered is:

- How can the robot change its security guard tour dynamically in real-time depending on detected anomalies?

1.3 Delimitations

This project is designed as a proof of concept to demonstrate the feasibility and potential effectiveness of integrating autonomous robots, in this case Boston Dynamics' Spot, into security and surveillance systems. The primary focus is on developing and testing a prototype that leverages robotic automation, along with sensing technologies like thermal imaging and computer vision, for improved security patrol operations. The safety and privacy concerns related to the use of robots and video surveillance will not be further discussed or investigated. Key delimitations include:

- **Scope:** The project aims to illustrate the potential of using mobile robots in security applications, not to develop a fully optimised or market-ready product.
- **Computer Vision Model:** The development of the computer vision model is exploratory. The model is designed to show how such technology can aid autonomous robots in security tasks. The RGB camera should work as a complement to the thermal camera, and using thermal data for the detection of anomalies is the primary focus during the tour.
- **Operational Environment:** Testing is conducted in a controlled environment to focus on the system's capabilities and limitations.
- **Safety and Privacy** This project will not go into the safety and privacy concerns of using robots and video surveillance. This topic is acknowledged but not further investigated.

1.4 Individual Contributions

This project is a shared effort between Johanna Häggström Wedding and Ella Thunborg. The work has been divided equally throughout the full project period, and both have contributed equally.

2

Theory

This chapter will provide the relevant theory about autonomous mobile robots, the control structure behaviour trees and thermal imaging.

2.1 Autonomous Mobile Robots

This section will explain the necessary theory about autonomous mobile robots for this project. The information relies on a report published by the International Federation of Robotics (IFR) in 2021 [IFR, 2021].

IFR defines autonomous mobile robots (AMRs) as a *"mobile platform that traverses the specified operating environment autonomously"* [IFR, 2021, p.5]. This definition includes the ability to navigate using obstacle and collision avoidance. Obstacles in the environment should be detected with sensors, and the robot should be able to change its path planning accordingly, meaning that it should be able to define a free path and not follow a predefined one. AMRs can be configured in several ways, from a mobile base such as a robotic vacuum cleaner to an advanced automated warehouse robot with a custom user interface. Some use cases include information robots, security robots, and healthcare robots. An example of a mobile robot is Spot, see Figure 2.1.

Figure 2.1 The quadruped autonomous mobile robot Spot developed by Boston Dynamics.

Autonomy and Navigation in AMRs

The International Federation of Robotics also states that *"autonomy in AMRs relates principally to navigation"* [IFR, 2021, p.6]. Their advanced navigation capabilities allow them to move through environments like factories and warehouses without following fixed paths, dynamically avoiding obstacles and recalculating routes as needed. Unlike conventional robots that operate based on fixed programs, AMRs adapt to their surroundings and internal states, using sensor data to navigate safely and efficiently even in unpredictable situations. Traditional robots can provide high precision at a high rate while executing different tasks, but the system is unable to adapt to the existing condition of the environment and make advanced decisions based on sensor inputs. [IFR, 2021]

Autonomous navigation for AMRs involves the ability to determine and execute a path between two points. In practice, the system should be capable of avoiding obstacles and dynamically finding new paths immediately. This capability comes from the advancement of Simultaneous Localization and Mapping (SLAM) algorithms [IFR, 2021]. These algorithms are designed to simultaneously create a map of the environment and pinpoint the robot's location within this map using sensor data. Given the limitations associated with each positioning technology—such as accuracy in distance measurement and performance under various environmental conditions like low visibility or extreme temperatures—SLAM algorithms typically integrate data from a diverse set of sources, including Light Imaging Detection and Ranging (LiDAR), radar, GPS, odometry, and ultrasound. In modern developments, Visual SLAM (VSLAM) has been introduced, enhancing SLAM algorithms with data from image sensors. This addition not only improves the environmental mapping with detailed visual information but also raises the complexity of data processing. VSLAM algorithms process large volumes of camera data using advanced techniques to filter out irrelevant data, which requires substantial processing power. Despite these challenges, the development of VSLAM is essential for enabling AMRs to accurately identify and react to various objects and individuals they encounter. [IFR, 2021]

2.2 Behaviour Trees

Behaviour trees (BTs) offer a structured approach to implementing autonomous control logic in robotics, that combines complexity management with modular design principles. This section is derived from the insights provided by Michele Colledanchise and Petter Ögren, focusing on the foundational aspects, advantages, and robotic applications of BTs [Colledanchise and Ögren, 2017].

Behaviour trees represent a methodology for managing the transition between tasks within autonomous agents, such as robots, emphasising modularity and reactiv-

ity. Colledanchise and Ögren define reactivity and modularity as *"... the ability to quickly react to changes"* respectively *"...the degree to which a system's components may be separated into building blocks, and recombined"* [Colledanchise and Ögren, 2017, p.5]. Originally developed for the computer game industry, BTs aimed at enhancing the modularity of non playable characters (NPC's) control structures [Colledanchise and Ögren, 2017, p.4-5]. BTs have expanded beyond their gaming origins, finding utility in autonomous vehicles, industrial robotics, and social robotics, among other domains. Their adaptability and modular nature make them suitable for a wide range of tasks requiring dynamic response capabilities [Colledanchise and Ögren, 2017, p. 15-21].

A BT is constructed from control flow nodes and execution nodes, arranged in a directed, rooted tree. The execution starts from the root node, with "ticks" propagating to child nodes based on control flow logic, ensuring an organised execution model. In this arrangement, each node is connected using standard hierarchical terminology, referring to them as 'parent' and 'child'. The root node is defined as having no parents, distinguishing it from all other nodes, which each have one parent. Control flow nodes are specifically required to have at least one child. [Colledanchise and Ögren, 2017, p.6]. For an example of a BT with an included subtree, see Figure 2.2. The root node is the top one, having three children. This BT will execute from top to bottom and left to right.

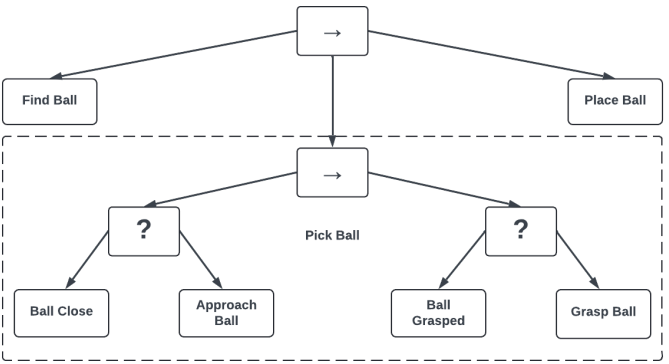


Figure 2.2 A high level BT for a robot that finds a ball, picks it up and places it. The pick ball subtree structure is included in the middle of the picture. This figure is inspired by Fig 1.1 displayed in [Colledanchise and Ögren, 2017, p.4].

The most basic language for a BT includes four types of control flow nodes: Sequence, Fallback, Parallel and Decorator. The execution nodes have two types: Action and Condition [Colledanchise and Ögren, 2017, p.6]. In Figure 2.2 the Se-

quence node is the arrow to the right, and the question mark defines a Fallback node. The Sequence node will send ticks to each child from left to right. It stops and returns either 'Failure' or 'Running' if any child returns one of these statuses, meaning that it does not send ticks to any subsequent children in that case. If all children return success, then the Sequence node will also return success. The Fallback node sends ticks to its children from left to right. It stops and returns 'Success' or 'Running' as soon as one child returns one of these statuses. The node returns 'Failure' only if all its children return 'Failure'. If a child returns 'Running' or 'Success', the Fallback node does not send ticks to any subsequent children. [Colledanchise and Ögren, 2017, p. 6-7]

Comparing BTs with models like finite state machines (FSMs) and hierachical finite state machines (HFSMs), BTs excel by avoiding the reactivity-modularity trade-off, thanks to their use of two-way control transfers. This design principle enhances both the system's responsiveness and its developmental modularity [Colledanchise and Ögren, 2017, p.23-25]. BTs can be extended with stochastic modelling, automated planning, and machine learning, offering new dimensions for analysis and application. These extensions underscore BTs' flexibility and robustness in complex systems.

2.3 Thermal Imaging

The images captured by thermal cameras differ significantly from standard RGB cameras, primarily in their ability to visualise heat variations in grayscale for the captured images. In thermal imaging, each pixel is assigned a value from 0 to 255, which represents the intensity of thermal radiation detected. This is in contrast to RGB cameras that encode colour information across multiple channels. Consequently, in a thermal image, pixels with values of 0 and 255 correspond to the minimum and maximum intensities, respectively [FLIR Systems, 2024a]. The analysis of temperature variations within the frame involves the use of both the median and mean values to highlight significant differences, indicative of temperature variations for specific objects. According to [Nilsson, 2023, p.91], thermal network cameras operate based on the principle of infrared (IR) radiation. Thermal infrared radiation, a form of light invisible to the human eye, serves as the basis for thermal imaging. These cameras are most effective when there is a notable temperature difference within the scene. Nilsson states that the primary purpose of thermal imaging is to detect irregularities and suspicious activities due to its limited ability to provide identification information. The growing importance of thermal imaging technology in video surveillance systems stems from its affordability and versatility. In contrast to conventional cameras, thermal cameras demonstrate exceptional performance in low-light and poor visibility conditions, including those caused by fog and smoke for example, making them ideal for poorly illuminated areas.

Furthermore, thermal cameras offer privacy benefits by enabling detection without identification, which makes them suitable for locations such as train platforms or fenced-off areas. These cameras are particularly useful in environments like railway tunnels or airstrips, as they are unaffected by bright lights or laser beams. Integrating thermal cameras with video analytics solutions further enhances their utility, enabling functionalities such as detecting individuals in dark areas and alerting security staff. [Nilsson, 2023, p.107]

Computer Vision on Thermal Images

In recent years, the integration of computer vision techniques with thermal imaging has emerged as a powerful tool for detecting and tracking people in various scenarios. Computer vision algorithms applied to thermal images typically involve several key steps, including preprocessing, feature extraction, object detection and classification. Preprocessing techniques such as image enhancement and noise reduction are often employed to improve the quality of thermal images before further analysis [Ansari and Salankar, 2017]. Feature extraction methods such as histograms of oriented gradients (HOG) and local binary patterns (LBP) have been adapted to thermal images for capturing relevant patterns and textures [Munian et al., 2023]. Object detection algorithms, including deep learning-based approaches such as convolutional neural networks (CNNs), have shown promising results in detecting people in thermal imagery [Gupta et al., 2020].

The integration of computer vision with thermal imaging has enabled a wide range of applications in surveillance and security. Thermal imaging coupled with computer vision algorithms allows for the detection and tracking of people, even in low-light or adverse weather conditions. These systems are crucial for border security, law enforcement, and search and rescue operations, where traditional visible light cameras may be insufficient. Moreover, thermal imaging in combination with computer vision is increasingly being used in smart surveillance systems for crowd monitoring, detecting unauthorised access, and identifying suspicious behaviour [Tu et al., 2024].

Recent advancements in the field, such as the work by Tsai et al. [Tsai et al., 2022], have demonstrated the effectiveness of using deep learning with thermal imaging for human detection in heavy smoke scenarios. Their study proposed to use a thermal imaging camera with a deep learning model, specifically YOLOv4, for real-time object detection in low-visibility smoky fire scenarios. The model has a precision of detecting people in standing, sitting, and lying postures, where it achieves over 95 % precision for the location of people in standing postures with a frame rate of 30.1 frames per second. For people in lying postures, which is the most difficult to detect, it has a precision of over 80 %. This real-time result can provide valuable information for timely rescue efforts and the protection of firefighters in dangerous,

smoky fire situations.

Computer vision on thermal images presents a powerful tool for various real-world applications, offering insights into human presence and behaviour and enabling automated analysis of thermal data. By leveraging advanced algorithms and methodologies, researchers and practitioners can further enhance the capabilities of computer vision systems on thermal imagery, contributing to advancements in fields such as security, public safety, and surveillance.

You Only Look Once (YOLO)

The You Only Look Once (YOLO) algorithm revolutionised real-time object detection when it was first developed by Joseph Redman and Ali Farhadi in 2015 [Kili Technology, 2024]. YOLO is known for providing fast and accurate object detection and is widely used in applications such as video surveillance systems and for autonomous vehicles [Kili Technology, 2024]. Powered by convolutional neural networks (CNNs), YOLO achieves high accuracy by predicting both the object class and its location in images. It employs a single-shot detection approach, enabling efficient object detection in a single pass through the network [Kili Technology, 2024]. Additionally, the YOLO model demonstrates good performance in detecting small objects. This is due to the grid-based approach employed by the model. It employs a fully convolutional network architecture, which enables the efficient utilisation of GPUs during both training and inference. Furthermore, anchor boxes facilitate the ability of the model to detect objects of varying scales.

Despite these strengths, the YOLO model does have some limitations. While the model excels in object detection, it is less effective in tasks such as image or instance segmentation [Kili Technology, 2024]. Furthermore, its accuracy may not match that of other methods such as RetinaNet or Mask R-CNN, particularly in scenarios requiring high precision. Even though YOLO can detect small objects well it may struggle with very small objects, especially when they are close to other objects, because of its grid-based approach. Additionally, it lacks tracking capabilities, limiting its suitability for applications requiring continuous object monitoring over time, such as certain video surveillance scenarios [Kili Technology, 2024].

The YOLOv8 model is one of the latest iterations of the YOLO model developed by Ultralytics in 2023. It offers cutting-edge performance when it comes to speed and accuracy, where the YOLOv8 model is both faster and more accurate than its older versions [Ultralytics, 2024]. Some key features Ultralytics points out with YOLOv8 are that it uses new advanced backbone and neck architectures, contributing to improvements in object detection and feature extraction. Instead of using an anchor-based approach, it employs an anchor-free split Ultralytics head, contributing to better accuracy for the model. Ultralytics has optimised the accuracy and

speed tradeoff, making it continuously suitable for real-time detection. It offers a variety of pretrained models, making it more suitable for a wider range of applications [Ultralytics, 2024]. Besides object detection, it supports instance segmentation, pose/keypoint detection, oriented detection, and classification. In summary, the YOLO algorithm, developed in 2015, revolutionised the field of real-time object detection by combining speed and accuracy. Despite certain limitations, YOLOv8, the latest iteration by Ultralytics, offers enhanced performance and versatility for real-time detection needs.

3

Experimental Setup

This chapter will give a detailed description of the experimental setup. The external cameras will first be explained, and then the necessary information about the robot will be presented.

3.1 Camera Components

In this project external cameras were mounted on the robot and a thermal camera sensor and RGB camera sensor was connected to a Main Unit. The components can be seen in Figure 3.1.

Figure 3.1 From left to right the main unit [Axis Communications, 2024c], the thermal modular camera [Axis Communications, 2024a] and the RGB modular camera [Axis Communications, 2024b].

Main Unit

The AXIS FA54 Main Unit can connect up to four AXIS FA sensor units. Two different sensors are connected to the main unit, the AXIS FA1080 and AXIS FA1150. It can complement larger systems or serve as the foundation for a small-scale surveillance setup. The key feature is that it offers simultaneous streaming

from all four sensor units, Quad View capability, Forensic Wide Dynamic Range (WDR), and Zipstream technology for efficient bandwidth usage. These core capabilities are explained below:

- **Quad View Capability:** All the four sensor units can be viewed at the same time.
- **Wide Dynamic Range:** A notable difference in light intensity between the darkest and brightest parts of a setting.
- **Forensic WDR:** A cutting-edge WDR technology that significantly minimises visible distortion in the captured frames. This technology allows for clear detail visibility even in settings with a wide dynamic range.
- **Zipstream technology:** A compression method that retains important details in captured frames while removing unnecessary information.

It includes built-in analytics, corridor format support, High-Definition Multimedia Interface (HDMI) connectivity, two-way audio, configurable I/O (Input/Output) ports, and local storage options. [Axis Communications, 2024c]

Thermal Camera

The AXIS FA1080-E Thermal Sensor Unit is a thermal imaging device that can be used for a wide range of applications. It works by creating images based on the heat that radiates from an object, vehicle, or person. This makes it less sensitive to lighting conditions, such as shadows, backlight, darkness, and camouflaged objects. This means that it can guarantee reliable images under various conditions.

The FA1080-E is equipped with an uncooled microbolometer and a 2.2 mm fixed lens, enabling it to capture thermal images with a resolution of 208 x 156 pixels at 8.3 frames per second (fps). It has an operating temperature range between -30 to 55 °C. This resolution, combined with the use of analytics, enables the device to detect people, vehicles, and objects in all lighting conditions. The 2.2 mm lens offers a 63 degree field of view, which captures a wide perspective on the scene. Thanks to its compact size, the sensor is versatile and can be mounted behind surfaces, on walls, or in various setups, making it suitable for a range of applications where detecting objects in challenging lighting conditions is required. [Axis Communications, 2024a]

RGB Camera

The AXIS FA1105 Sensor Unit represents a cutting-edge device designed for discreet indoor surveillance. This CMOS sensor unit is ideally suited to applications

in stores, banks, and airports and can also be integrated into machines. Due to its small size, it can be installed in tight places or mounted discreetly at eye level, thus enabling greater detailed images.

The FA1105 provides a 1080p resolution, 1920 x 1080 pixels at 25/30 fps, and a 111 degree horizontal field of view for wide area coverage. It has an operating temperature range between -20 to 50 °C. It supports forensic WDR at all times, enabling details to be visible even in high-contrast lighting conditions. This feature is particularly useful in a backlight scenario where a person is standing in front of a very bright background. [Axis Communications, 2024b]

3.2 Spot

The rest of the text in this chapter will cover the necessary information about Spot and the Software Development Kit (SDK) developed by Boston Dynamics. This text is derived from the SDK version 3.2.2 [Boston Dynamics, 2024j].

As mentioned in the previous chapter, Spot is a quadruped agile mobile robot developed by Boston Dynamics, see Figure 3.2 for a visual representation. Spot is equipped with additional payloads, in this case an arm and a LiDAR. It is designed to operate in different types of environments and terrains [Boston Dynamics, 2024b], from factory floors to construction sites to research labs and more. It can be used to provide information in routine operations, conduct site inspections, enhance safety protocols, and potentially minimise the risk associated with hazardous scenarios [Boston Dynamics, 2024k].

Figure 3.2 Spot from Boston Dynamics.

Hardware Specifications

[Boston Dynamics, 2024a] states that Spot weighs 32.5 kg without the arm, has a length of 1100 mm, a width of 500 mm, and a height of 840 mm. It has a maximum movement speed of 1.6 m/s. The legs have 3 degrees of freedom (DOF) each, adding up to a total of 12 DOF. The arm adds 6 degrees of freedom and a gripper to the system. It weighs 8 kg, is 984 mm long, and can reach a height of 1800 mm when mounted on the robot [Boston Dynamics, 2024i].

The robot is equipped with five pairs of stereo cameras with an operating range of 4 m and a 360-degree field of view [Boston Dynamics, 2024a]. They are capable of capturing images and videos in black and white, helping the robot avoid obstacles in its nearby environment. Each hip and knee consist of two actuators, respectively one actuator. The hip joints are referred to as HX and HY for each plane of rotation, the internal coordinate system can be seen in Figure 3.3. [Boston Dynamics, 2024a] has therefore defined the movement as:

- 12 degrees of freedom, 3 per leg
- Hip Joint X-axis (HX) +/- 45 degrees from vertical (45 degrees of internal and external rotation from vertical)
- Hip Joint Y-axis (HY): +/- 91 degrees with 50 degrees bias from vertical (flexion/extension).
- Knee: +/- 14-160 degrees from straight (flexion extension range from 14 to 160 degrees).

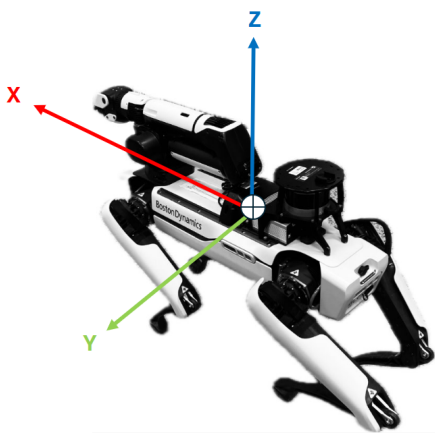


Figure 3.3 The internal Coordinate system, drawing inspired by [Boston Dynamics, 2024a].

3.3 Spot Software Development Kit

Spot Software Development Kit (Spot SDK) can be used when developing applications and payloads for Spot. It consists of conceptual documentation, a Python client library, Payload developer documentation, Spot API protocol definition, and the Spot SDK Repository.

API and Communication

The Spot API empowers applications to manage Spot, access sensor data, and develop and incorporate payloads. Following a client-server model, the Spot API facilitates communication between client applications and services operating on Spot via a network connection [Boston Dynamics, 2024d]. Figure 3.4 presents a high-level overview of the robot's services.

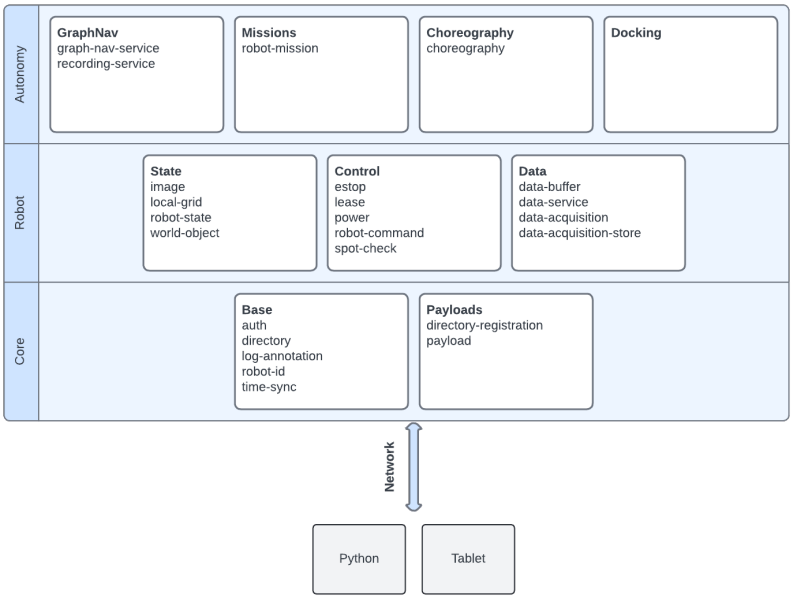


Figure 3.4 A diagram of the high-level overview of the robot's services, illustration inspired by [Boston Dynamics, 2024d].

It is possible to run the client applications on tablets, laptops, cloud-based applications, or payloads that are connected to Spot. As for the connection between the components, any IP network should work. The network choice depends on the usage and application environment. The four available solutions according to [Boston Dynamics, 2024h] are listed and explained below:

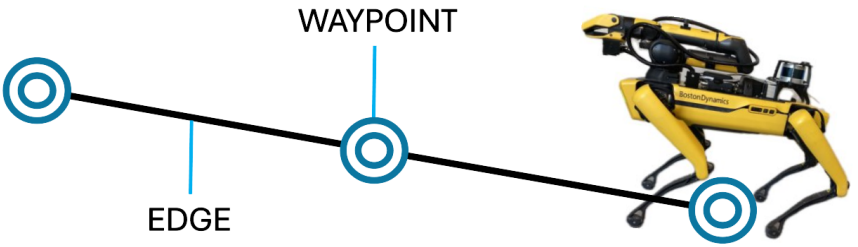
- **Spot as connected peer:** It is possible to run applications on computers that are directly connected to Spot using different ports: the rear RJ-45 port, the DB-25 payload port, or the RJ-45 port on a Spot GXP payload. This setup offers a dependable and fast communication link without the need for additional infrastructure. However, it does impose restrictions on where the application can be deployed.
- **Spot as WiFi access point:** Applications located close to Spot can connect to its WiFi access point and communicate directly, without the need for any external networking infrastructure.
- **Spot as WiFi client:** Spot has the capability to join an existing WiFi network, allowing applications to connect to the same network and communicate with Spot. While this method extends the potential distance between the application and Spot, it is important to be mindful of issues like dead zones in the network.
- **Spot via custom communication links:** For network design scenarios where the options mentioned earlier are not sufficient, custom communication links like cell modems and Persistent Systems radios can serve as a bridge between Spot and applications. These alternatives become valuable when the standard connectivity methods may not fully meet the requirements.

Protocol Buffers and gRPC

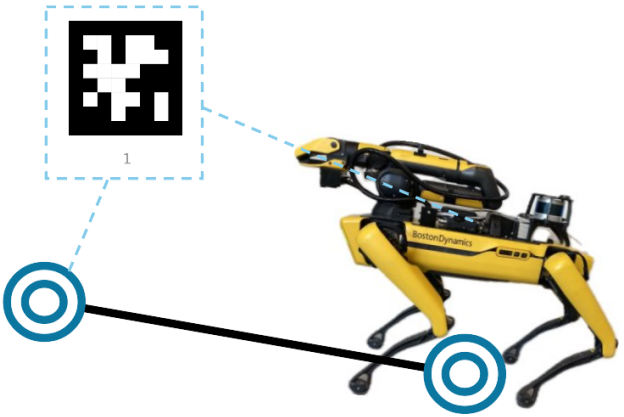
gRPC is the primary application-level protocol for the Spot API. gRPC is an open-source remote procedure call (RPC) framework developed by Google. It is designed to enable communication between applications and services running on different machines or in different environments [gRPC, 2024]. Protocol Buffers allow developers to define the structure and type of data being exchanged. This system ensures that the data is language-agnostic and optimised for communication, supporting a wide array of programming languages and environments [Boston Dynamics, 2024h].

GraphNav

The autonomy of the robot relies heavily on the GraphNav service and its functionalities for location-based localization and locomotion. It allows the user to perform tasks such as localising the robot's position, navigating predefined routes, and downloading map data [Boston Dynamics, 2024f]. Maps can be recorded using the GraphNavRecording service or by operating the robot manually by using the tablet's Autowalk feature [Boston Dynamics, 2024e]. These maps consist of waypoints, locations, and edges, which are the connections between waypoints. Figure 3.5 provides a visualisation of this [Boston Dynamics, 2024f]. The key concepts derived from [Boston Dynamics, 2024f] text about the GraphNav Service are explained below.



(a) Edges and waypoints representation.



(b) The robot uses fiducials for localisation

Figure 3.5 Visual representation of waypoints, edges and fiducials. Inspired by [Boston Dynamics, 2024f].

Waypoints. Waypoints serve as specific locations for the robot's navigation, each accompanied by a snapshot of feature data from its respective location. This feature data, derived from the robot's onboard sensors, is crucial for the robot to continually monitor and adjust its position as it moves between waypoints. Additionally, waypoints capture features such as detected fiducials. Fiducials are QR codes that are used to help with localisation, visualised in Figure 3.5b and are strategically placed in less sensor-rich areas to aid the robot in navigating through challenging terrain. Furthermore, the integration of a mounted LiDAR sensor enhances the robot's sensory capabilities. The LiDAR sensor provides richer information for each waypoint, significantly improving the robot's ability to accurately track its position across a broader range of environments. Figure 3.6 shows waypoints in a recorded map. In waypoint 6, the robot will stand with its front facing in the same direction as the red arrow. Consequently, when moving to waypoint 7, the robot will make a rotation

and end up with its front pointing in the same direction as the new red arrow. This is in line with the coordinate system in Figure 3.3.

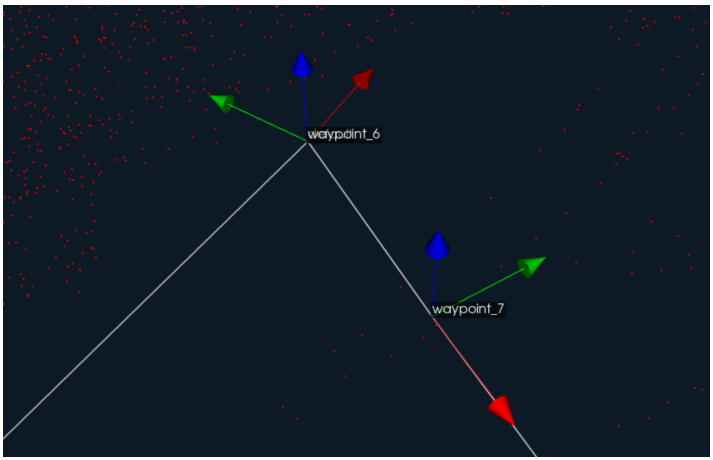


Figure 3.6 Waypoints represented in a recorded map.

Edges. The edges in the map, represented by white lines in Figure 3.6, show how the robot moves between the waypoints. These edges do both contain a relative pose of the connected waypoints and a description of how the robot should move along the edge; for example, if an edge were in a staircase, the edge would contain information about the staircase. The robot navigates from each waypoint to the next one by following these edges and staying within a corridor defined by each edge. Usually, this corridor is up to two metres wide, but this can be changed by the user. Graph edges are formed consistently within local contexts, but there is no overarching "global frame" for the entire map. When multiple paths exist between two waypoints in the graph, combining the transformations along these paths may yield distinct results in the resulting transformations between the two waypoints.

Mapping the environment. In order to map and record the environment, the robot is to be manually driven. The resulting graph can be used to localise the robot within the mapped environment. Once it has been recorded, the robot can autonomously traverse the environment within that graph. The map must be downloaded from Spot to persist.

Implementing GraphNav. The GraphNavService RPCs facilitate interaction with the robot's navigation functionalities. These RPCs include SetLocalization for manual localization triggering, NavigateRoute for directing the robot along predefined routes, NavigateTo for guiding the robot to specific waypoints, GetLocalization-State for retrieving localization status and sensor data, ClearGraph for clearing the graph structure and DownloadGraph/UploadGraph for transferring waypoint and

graph topology data. Similarly, the GraphNavRecordingService RPCs are utilised for map recording functionalities. Key RPCs in this service include StartRecording for initiating waypoint sequence recording, CreateWaypoint/CreateEdge for adding waypoints and edges during recording, and StopRecording for pausing the recording service. To summarise, implementing the GraphNav Service involves four main phases according to [Boston Dynamics, 2024e]:

1. **Recording a Map:** This can be done manually using the tablet controller's Autowalk feature or through client applications using the GraphNavRecordingService RPC. The Autowalk Service will later be explained in detail.

2. **Uploading the Recorded Map to the Robot:** Once the map is recorded, the data needs to be uploaded to the robot.

3. **Initializing the Robot's Location Within the Map:** Before navigation can begin, the robot's initial location within the map needs to be established.

4. **Commanding the Robot to Navigate to Specific Waypoints Within the Map:** With the map recorded, uploaded, and the robot's initial location initialised, the robot can be commanded to navigate to specific waypoints within the map. The robot will try to follow the recorded route as closely as possible but can adapt its path if obstacles are encountered. However, if the deviation from the recorded route is significant, the robot may declare itself stuck.

Mission Service

The Mission Service is used when developing high-level autonomous behaviour for Spot, implementing behaviour trees for decision-making. The BT consists of structural nodes (such as Sequence, Selector, Repeat, etc.) and action nodes (such as Condition, BosdynRobotCommand, BosdynNavigateTo, etc.). The structural nodes control how the tree is parsed, and the action nodes perform a defined action. A selection of available structural and action nodes are listed in Table 3.1 and Table 3.2. They are both based on concepts presented by [Boston Dynamics, 2024g] in their documentation on the Mission Service. The tables have been adapted to focus on elements directly relevant to this project.

Table 3.1 Structural nodes

Node	Description
Sequence	Executes a series of child nodes sequentially, stopping if a node fails. This allows for complex behaviours through nested sequences.
Selector	Attempts to run child nodes one by one, succeeding if any single node succeeds.
Repeat	Repeats execution of its children nodes a specified number of times or until a failure occurs.
Retry	Continues to execute child nodes until successful completion.
ForDuration	Executes child nodes repeatedly within a specified time frame until a node fails.
SimpleParallel	Runs multiple nodes concurrently, allowing for simultaneous tasks.
Switch	Chooses one child node to execute based on specified conditions.

Table 3.2 Action nodes

Node	Description
Condition	Evaluates conditions to return true or false, guiding the decision-making process.
BosdynRobotState	Retrieves and provides information on various robot metrics such as battery level and system states.
BosdynRobotCommand	Commands the robot with specific actions like standing, sitting, or navigating to a target location.
BosdynPowerRequest	Controls the robot's power state, turning it on or off as required.
BosdynNavigateTo	Guides the robot to move autonomously, with adjustable parameters for navigation.
BosdynGraphNavState	Saves navigation states to a central data store for use in conditional assessments.
RemoteGrpc	Allows for the customisation of mission behaviour, such as integrating sensor readings into the robot's tasks.
Prompt	Engages a supervisor through queries, requiring input to proceed with tasks.
DefineBlackboard	Sets up a variable on the blackboard for child nodes to utilize.
SetBlackboard	Assigns values to previously defined blackboard variables.

In the context of behaviour trees, a blackboard is a messaging capability that allows nodes to share the state in a behaviour tree [Boston Dynamics, 2024g]. This can be used for reading and writing the current state, which allows behaviour trees to be more flexible and scalable.

Autowalk Service

The Autowalk service provides an abstraction of the mission service, enabling API clients to define high-level autonomous behaviour for robots using the autowalk format [Boston Dynamics, 2024c]. In other words, the autowalk feature makes it possible for users to record and replay a sequence of missions or autonomous behaviours. Recording and replaying the autowalk can be done directly by using the tablet. For example, when recording an inspection tour, fiducials should first be placed around the route. They should be positioned in a manner that allows the robot to see them when recording and replaying the autowalk. Then, the robot should walk along the intended route while being manually controlled, and the recording is on. After that, the robot can follow the same route without being controlled by a person, and it can avoid obstacles along the route.

The autowalk app, deployed on the tablet, generates a mission behaviour tree based on the recorded autowalk. The behaviour tree for the autowalk mission becomes highly complex as it manages various failure scenarios, resulting in a significant depth of 42 layers and a breadth of thousands of nodes [Boston Dynamics, 2024c]. Nevertheless, at its fundamental level, the structure remains relatively simple, essentially providing the robot with instructions to navigate to a specific location and perform a specific action.

However, modifying the simple "go here, do this" instruction can be challenging due to the complexity of the behaviour tree. The autowalk functionality can be regarded as a high-level mission, which enforces a pattern of "go here, do this" sequential behaviour. This approach is more straightforward but less flexible than missions. Nevertheless, the autowalk service provides methods for breaking this sequential structure. These methods suggest the use of behaviour trees, either by inserting BTs into an autowalk action or by making the autowalk a root node to insert that node into other BTs. [Boston Dynamics, 2024c]

4

Method

In this chapter, the method for the project will be presented. First, the camera setup will be explained and visualised, followed by the process of generating a computer vision model for detecting anomalies. Subsequently, the construction of a robotised guard tour will be explained, including the communication between the external camera and the robot. Finally, a specific section will outline how the testing process was conducted for this project.

4.1 Camera Setup

When mounting the cameras on Spot, there were mainly two considerations:

1. The cameras should not affect, disturb, or in any way compromise Spot's movement. This means that the cameras needs to be fully integrated into the robot, operating autonomously within the robot's network. It is important that the cameras can function as an independent unit, but the power could be drawn from either a mounted powerbank or from Spot directly.

2. The cameras needs to have free sight, and the full field of view should be used. This will create limitations on where the cameras can be mounted on the robot.

Based on these considerations, the cameras were mounted on the lower part of the gripper, ensuring optimal integration and field of view without compromising Spot's mobility, see Figure 4.1. The main unit and the power bank were mounted on the body, using the available rails. To prevent the cables from getting caught in the joints or winding around the arm, a protective plastic pipe was placed at three strategically placed mounting points on the arm. The effectiveness and positioning of these mounting points were carefully evaluated during the robot's operation, making sure

that the arm could move freely. All components used for the proper mounting of the cameras, including the external materials, were custom-designed in CAD and produced via 3D printing.

In order to place the cameras with free sight at the bottom of the gripper, simple trigonometry was used. The vertical field of views for the thermal and RGB cameras are 42 and 61 degrees respectively. Therefore, being the critical one, the RGB camera was used in the calculations. The camera needs to be placed in a position where the tip of the gripper cannot be seen in the captured frames.

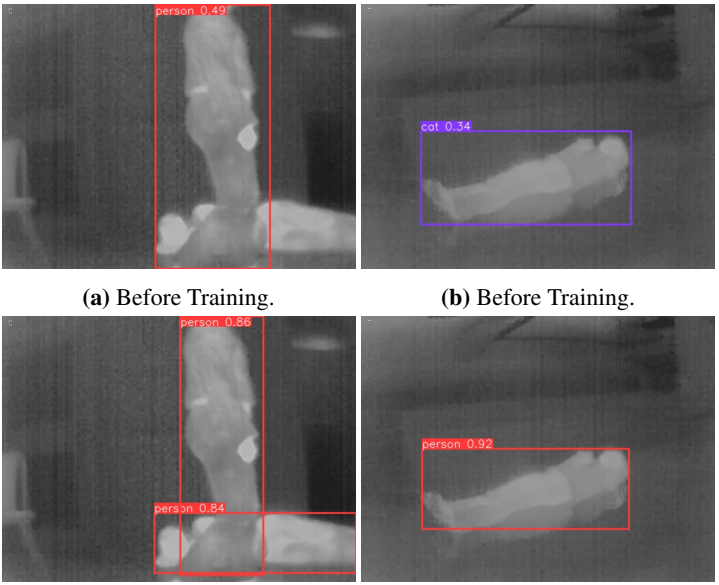
Figure 4.1 Camera placement on Spot.

4.2 Computer Vision

As stated in the introduction of this thesis, the model for detecting anomalies is designed to showcase how such technology can aid autonomous robots in security tasks, especially using the thermal camera. Consequently, the decision was made to focus on detecting anomalies related to people in various postures. This choice is particularly relevant for surveillance purposes. Firstly, data on human figures is readily accessible, making it practical to implement. Secondly, from a surveillance perspective, monitoring postures can be crucial—for instance, during the night, no one should be present at certain locations, and any detected presence can immediately signal a breach. Lastly, focusing on a construction site is especially relevant due to the high risks associated with the environment. Construction sites are inherently dangerous places where swift identification of unusual or incorrect postures could preempt accidents or alert to hazardous situations, significantly enhancing safety and operational security. According to the Swedish Work Environment Authority (swe: Arbetsmiljöverket), the number of accidents and occupational illnesses among construction workers is significantly higher, twice as high, than among other workers. The causes include falls from heights, accidents while using different types of machinery, and a variety of stress injuries [Arbetsmiljöverket, 2023].

Initially, the YOLOv8 model performs poorly to accurately detect people when us-

ing a thermal camera. This problem can be seen in Figure 4.2a and Figure 4.2b. The model's confidence in detecting people is typically below 50%, and it often fails to detect people when they are lying down. Instead, it may misidentify humans as other objects, such as a cat, as shown in this example.



(a) Before Training. **(b)** Before Training.

(c) After Training. **(d)** After Training.

Figure 4.2 Comparison of model performance before and after training.

Dataset

In order to obtain better results with the YOLOv8 model, a custom dataset was created. This was done by using Roboflow, which is a website where you can easily create, annotate, and customise your own dataset. The dataset itself contains images from thermal videos with people in different postures. It contains different backgrounds and multiple people as well. The dataset is built upon videos from three different sources. One of the sources of images comes from videos that have been captured within the robotics lab and videos from the construction site. The Kaggle AAU TIR dataset [Aalborg University, 2024], and its thermal images have been used, as well as the FLIR ADAS dataset [FLIR Systems, 2024b]. The dataset created only contains one class, which is the "person" class. After uploading the images to Roboflow, the images are annotated. It is important to include annotations on images without a person so that the model does not over-detect people. Once the dataset is annotated, it can be exported to the required format for the YOLO model. When the dataset exists in the correct format for YOLO, the training of the

model starts. Depending on the results that the trained model shows, it can be further improved by adding more data. For example, if the model is performing poorly at detecting people in a lying posture, adding more images of people in lying postures can improve the model. As can be seen in Figure 4.2c and Figure 4.2d the trained model performs with a higher accuracy, in these figures above 80%.

4.3 Robotised Guard Tour

The robotised guard tour is explained in the following sections. This includes how to create a map, how to plan the route, how the behaviour tree is built, and how to communicate with the external cameras and robot. The final section will summarise the generation and execution of the entire inspection tour by visualising the high level workflow.

Generating the Map

The first step in creating the robotised guard tour is to record a map. In order for the robot to be able to navigate a specific route, it needs to have created a waypoint there, which it can only do within a recording of a map. As previously mentioned in Section 3.3 this can be done in the two following different ways:

- By manually recording a map with the tablet controller using the Autowalk feature.
- By recording a map through the client applications utilising the GraphNavRecordingService RPC.

The second option should be used as it allows the recording to be manipulated during the process, providing greater control over the map creation. For instance, default waypoints can be added to the map while recording, enabling the user to orient the robot in more precise positions and customise the map according to specific requirements.

In order to start a recording, the robot needs to have a fiducial in its view to initialise its position. The robot should always have at least one fiducial in sight in order to minimise the risk of the robot getting lost. Then the *Recording_Command_Line* script can be used to develop a map that uses the GraphNavRecordingService which is essential for creating detailed maps of the robot's environment, since it enables the recording of waypoints and edges as the robot navigates through its surroundings. The GraphNavRecordingServiceClient is used for starting and stopping the recording, creating waypoints and edges, and managing the recording status. This script also makes it possible to generate new edges between waypoints that the robot

has not traversed directly between. As long as the robot has been to a point in the recorded map, it is possible to create edges without the need to go there if the corresponding waypoints are known. An example of this can be seen in Figure 4.3, where an edge has been manually added in between waypoint 5 and waypoint 12.

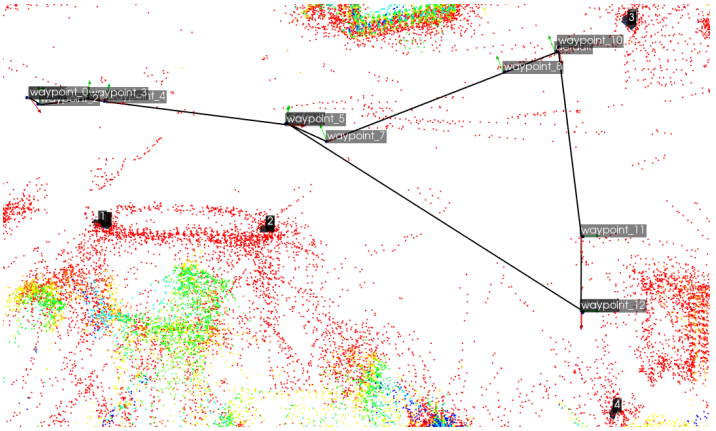


Figure 4.3 Example of a small map recorded in the robotics lab, with edges, waypoints and fiducials visible.

Generating the Route

Before constructing the behaviour tree, the map must be analysed and modified. First, all waypoints on the map needs to be stored and converted into NavigateTo nodes. The waypoint snapshot files are stored in a directory for the specific map. These waypoint snapshot filenames do not correspond to the waypoint IDs needed in order to create the navigational nodes. Access to the waypoint names is only possible when connected to the robot and utilizing the GraphNav service. Therefore, once the map is recorded, all the names of the waypoints are transferred to a text file. The text file created can be used to generate the NavigateTo nodes for each waypoint, which are used when creating the behaviour tree.

Deciding where deviations from the original route should occur is the next step. This decision varies based on the specific environment and the objectives of the inspection tour. In this project, important features such as restricted or forbidden areas, junctions offering multiple routes, and designated inspection points are identified on the map. It is also possible to determine where edges are missing and between which waypoints new edges need to be added. This addition of edges must be performed while connected to the robot, and accurate knowledge of the waypoint names stored in the text file is needed.

Building the Behaviour Tree

In order to enable the robot to make decisions based on the camera input and to deviate from its original route, a behaviour tree has been implemented. The basic structure of the behaviour tree used can be divided into the following subtrees:

Sequences – This subtree consists of a Sequence node that executes every child node in it, from the first one at the top to the last one at the bottom. These nodes can contain different kinds of nodes, such as NavigateTo nodes, RobotCommand nodes, or other subtrees. The structure of it can be seen in Figure 4.4, when executing this behaviour tree the robot would navigate from waypoint 1 to waypoint 2 and lastly to waypoint 3.

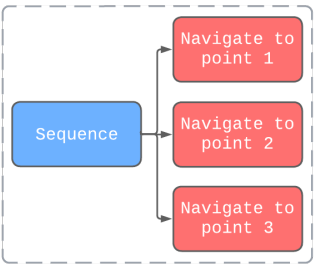


Figure 4.4 Subtree for a mission that navigates to point 1, 2 and then 3.

Inspection without action subtrees – This subtree contains a Prompt node that asks a question to a Python script using the computer vision model. The Python script analyses a certain amount of frames from the camera feed and then gives a response based on that camera input. After receiving the answer, it moves on to the next node in the sequence without doing anything different based on the input, other than inspecting that point. The subtree for this can be seen in Figure 4.5.

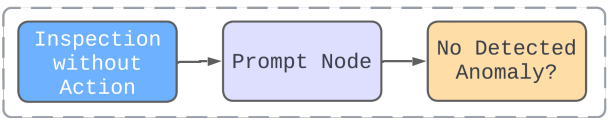


Figure 4.5 Subtree for the inspection without action subtree.

Inspection subtrees – This subtree uses a Selector node where one of its children needs to have a successful execution in order for the Selector node to succeed and move on to the next node in a sequence. In this Selector node, one child has a Prompt node that communicates with the camera feed, and if the camera feed detects

an anomaly, the Prompt node fails its execution since it expects no anomaly to be detected and therefore executes its other child, which can be a sequence of nodes or a single mission. How this subtree is built is shown in Figure 4.6a. The inspection subtree might be used when the robot is supposed to investigate something in a specific point, and upon detecting something, it takes a small detour as seen in Figure 4.6b.

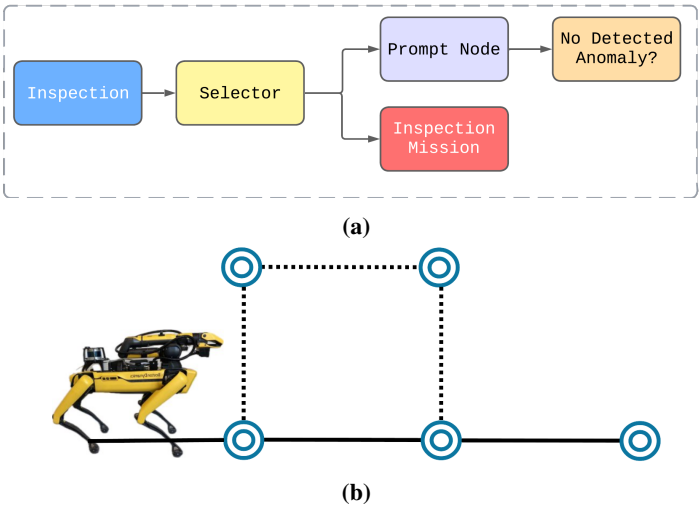
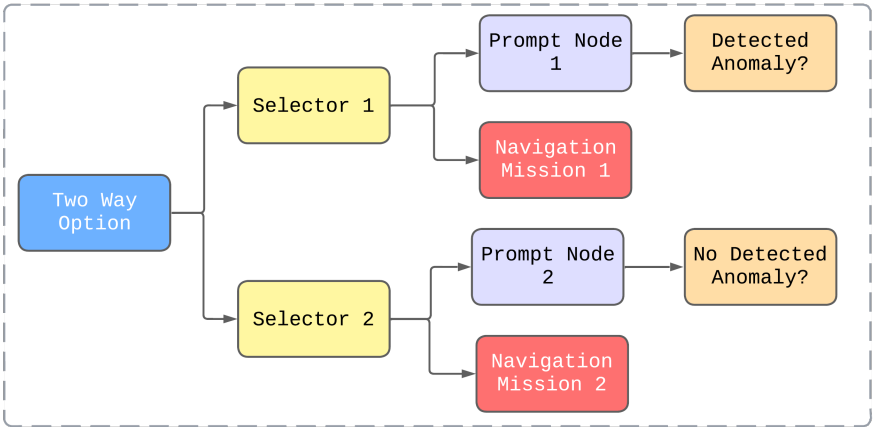
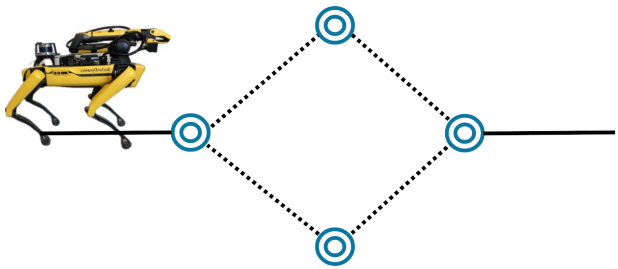


Figure 4.6 Subfigures (a) and (b): Subtree for an inspection route (top) and a scenario of when the inspection subtree could be used (bottom).

Two Way Option subtrees – This subtree is like a double inspection, where one of the Prompt nodes expects a detected anomaly and the other expects no detected anomaly. When the first Prompt node detects an anomaly, it jumps down to Prompt node two, and when it detects something there, it executes Navigation Mission 2, since the Conditional node will fail, and the Selector node runs its children until one of them succeeds. If it does not detect something on Prompt node 1, it executes Navigation Mission 1, and then jumps down to the next Prompt node, where it will get a success for the Prompt node when no anomaly is detected. This subtree can be seen in Figure 4.7a and can typically be used in a scenario shown in Figure 4.7b where from a given point there are two equally interesting routes leading to the same point, and the robot executes only one of its missions depending on if a anomaly has been detected or not.



(a)



(b)

Figure 4.7 Subfigures (a) and (b): Subtree for a two-way option that executes mission one if no anomaly is true (top) and a scenario of when the two-way option subtree could be used (bottom).

The modularity of these behaviour trees and subtrees allows for the easy addition of an inspection route along a predefined path. An example of how a small route could be constructed with different subtrees can be seen in Figure 4.8. In this example, the robot would have a base route, meaning no person was detected, from waypoint 1 straight to waypoint 5, followed by a sit-down command. In the event that the robot detects a person, it will also navigate to waypoints 2, 3, and 4 before proceeding to waypoint 5 and finally executing the sit mission, which makes the robot sit down. This could be considered a Behaviour Tree with two subtrees in it: one inspection subtree and one sequence subtree.

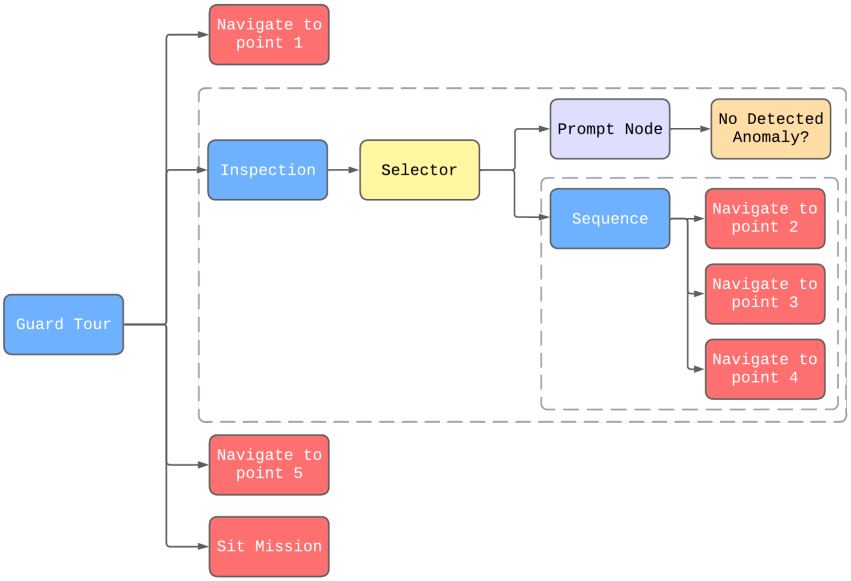


Figure 4.8 A small behaviour tree using the Inspection subtree and the sequence for navigating to several points.

Communication between Camera and Robot

The Main Unit is connected by Ethernet to Spot's rear Ethernet port at the back of the core. It is then possible to access the camera sensors through the Main Unit while being connected to Spot by WiFi. See Figure 4.9 for a visual representation.

The BT has communication nodes called Prompt nodes, which are described in Table 3.2, and the use of them can be seen in Section 4.3. When the BT reaches a Prompt node, the Prompt node will ask the supervisor a question and wait for an answer before continuing. In this setup, the supervisor is the Python script responsible for the computer vision that is running on the laptop. Figure 4.10 gives a visual representation. Spot is running the mission, and at the same time, the Supervisor is constantly checking for a question to appear. When that is true, the Supervisor starts the recording and analyses whether a person has been detected. The answer will then be transmitted back in the same way to the Promt node, and the condition of the node decides which path the robot will take.

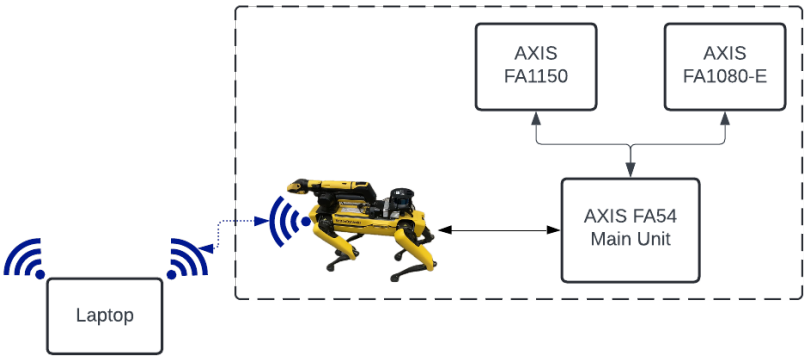


Figure 4.9 Connections in the system. The laptop is connected to Spot by WiFi, and the camera sits on top of Spot, connected to its rear Ethernet port.

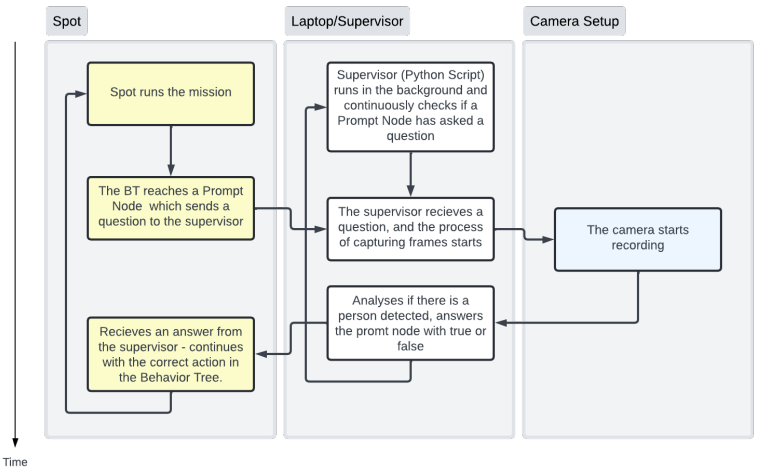


Figure 4.10 A flow chart of the process for accessing the external sensor data during the execution of a BT.

Workflow

The final step is to execute the generated behaviour tree. The correct map needs to be uploaded to the robot before, and the robot should be placed in sight of a fiducial. The behaviour tree can then be executed by using the example code from the Python library *Replay_Mission*. In order to replay a mission, which is the guard tour, it is

necessary to first record the map, generate the route, and then build the behaviour tree. As shown in Figure 4.11, which gives a high-level overview of how to create and execute a robotic surveillance route with Spot, the robot can change its original path during real-time execution.

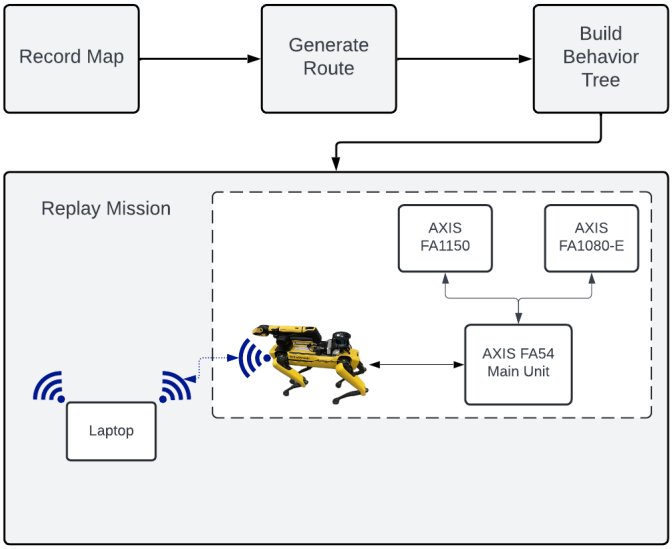


Figure 4.11 Workflow for creating and executing a robotised guard tour.

4.4 Testing

Throughout this project, testing was conducted in two different settings: the Robot Lab at LTH and a construction site in Lund named Vipan. The Robot Lab was the primary site for developing and iteratively improving the various modular designs for the BT, along with the gradual optimisation of the entire system. In contrast, access to Vipan was limited to 30 minutes per week, but it provided a more dynamic and extensive environment for testing. This setting allowed for the examination of Spot's capability to navigate on larger surveillance tours and adapt to the environmental changes that are included on a construction site. However, as Vipan was only accessed for a short time, the final result will also come from the Mechanical Engineering building (M-building) at LTH. This site was not used for testing and iteratively improving the solution, but only as a final step to collect results.

5

Results

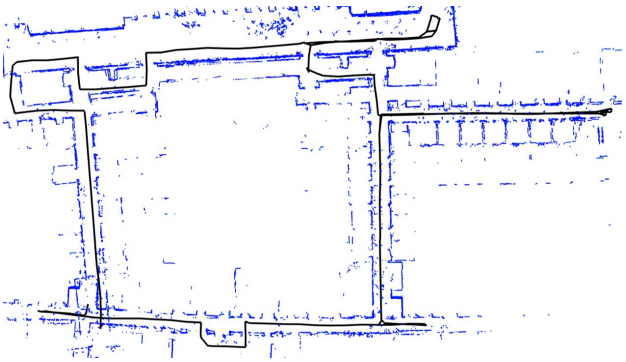
This chapter will provide the results of this project. The collection of results took place at two locations: M-building at LTH and the Vipan Construction Site. The robotised guard tour will first be explained and visualised for both locations, and lastly, corresponding BTs will be presented.

5.1 Robotised Guard Tours

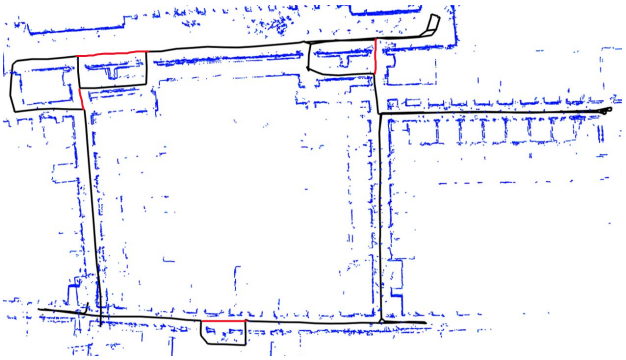
The robotised guard tours are created as described in Section 4.3. After creating the behaviour trees and uploading the map to the robot, the behaviour tree can be executed as a mission. The results from the execution of these missions in the M-building and at the construction site will be presented.

M-Building

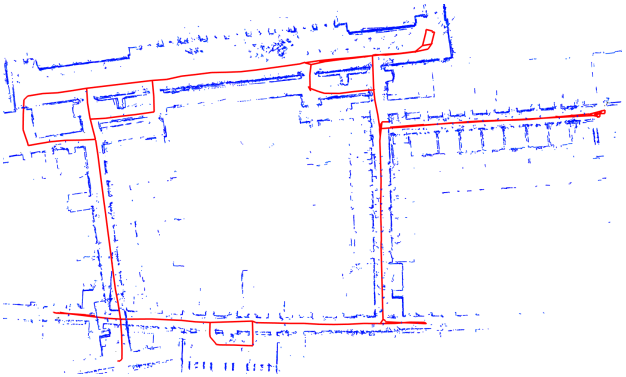
A map was recorded on the ground floor of the Mechanical Engineering building (M-building at LTH). The original map is shown in Figure 5.1a, and additional edges were manually added as described in Section 4.3. The locations of these added edges are illustrated in Figure 5.1b, where the red lines represent the manually added edges. The resulting map, which was later used for conducting the robotised guard tour, can be viewed in Figure 5.1c.



(a) The originally recorded map and its path represented as a solid black line.



(b) The originally recorded map, with red lines where edges were manually added.



(c) The final manipulated map with and its path as a solid red line.

Figure 5.1 Subfigures (a), (b), and (c): The difference between the original map and the manipulated map.

In total, this guard tour includes nine important points, as shown in Figure 5.2. During the surveillance tour, the robot initiates detection at these specific points. A total of 8 tours were executed. Responses from the supervisor, transmitted via a Prompt node, are presented as True or False in Table 5.1. Here, "False" indicates that no person was detected, while "True" indicates that a person was detected at that point. Each response corresponds to a specific point on the map. The computer vision model gave a correct answer for every detection.

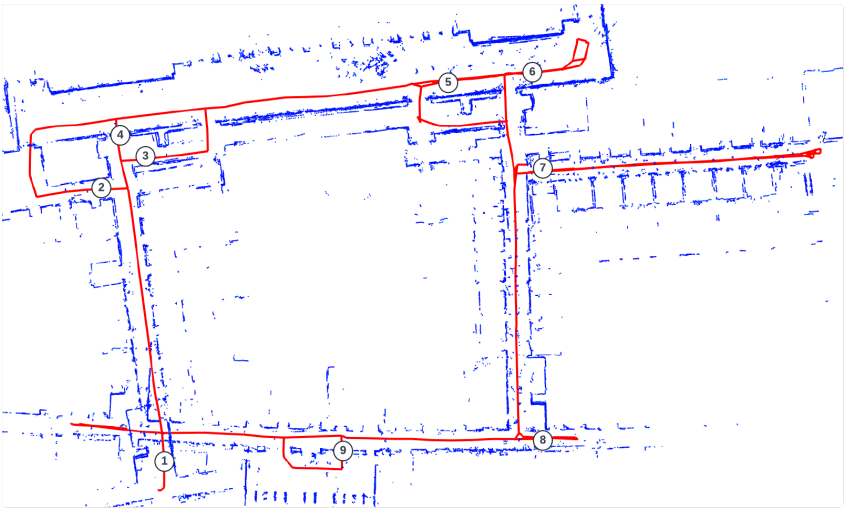


Figure 5.2 The points are numbered where the robot runs its detection and can take different routes.

Table 5.1 Results for detection of the rounds.

Point \ Round	1 & 8	2	3 & 7	4 & 6	5
1	False	True	False	True	False
2	False	True	False	True	True
3	False	True	True	True	False
4	False	True	False	True	False
5	False	True	True	True	True
6	False	True	False	True	False
7	False	True	False	True	False
8	False	True	False	True	False
9	False	False	True	True	True

The time it takes for the robot to complete each inspection round varies depending on the presence of people, as the route adjusts based on this input. Variations can also occur if changes in the surroundings cause the robot to take longer to localise itself at certain points. The times for each round, from start to finish, are documented in Table 5.2.

Table 5.2 Execution time for each round.

Round	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Time (min:sec)	4:58	7:40	5:50	7:55	5:30	7:41	5:08	4:15

Rounds 1 and 8 represent the base route, which is the route the robot takes if it detects no people, and it can be visualised as a comparison between the route the robot takes when it detects something at every point in Figure 5.3. The base route is drawn in black, while the full route is drawn in red. The graphs come from the data collected from the robot's periodic recordings of x and y coordinates during its inspection route. These coordinates are extracted from the robot's kinematic state and recorded relative to the 'gpe' frame, which represents the robot's ground plane estimate. The plots for all the rounds can be found in Section A.1.

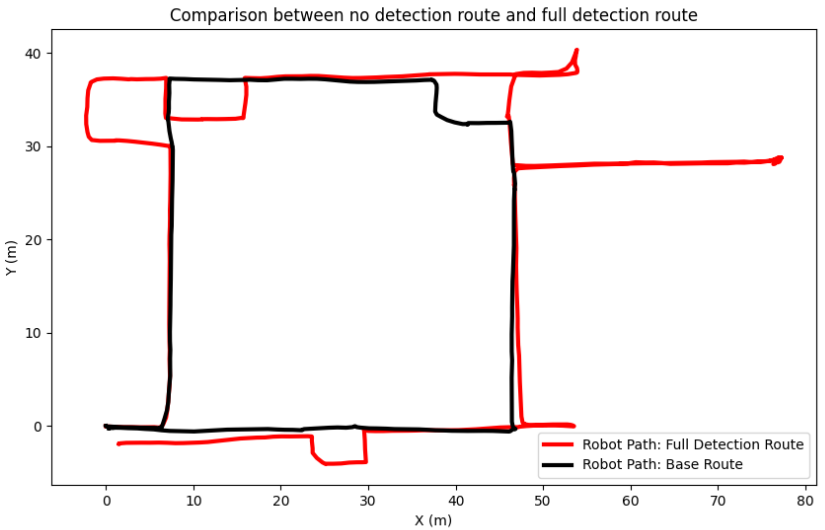
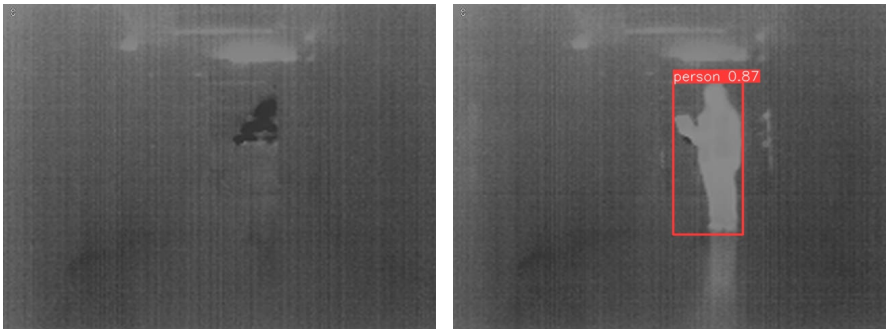


Figure 5.3 The base route in black represents the robot's movement when no one is detected. In contrast, the route in red illustrates the robot's movement when a person is detected at each inspection point.

For this robotised guard tour, the computer vision model predicts the specific points that were shown in Figure 5.2. The following result can be seen from the predictions from the model, where Figure 5.4 shows different results from point 2 and Figure 5.5 shows different results from point 6.



(a) **(b)**

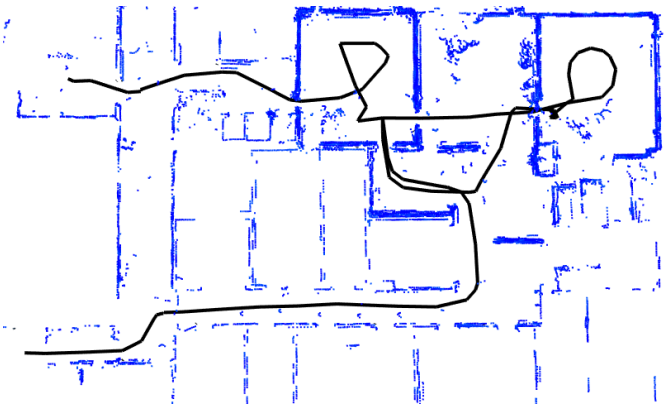
Figure 5.4 Subfigures (a) and (b): The difference between a false and true value in point 2.

(a) **(b)**

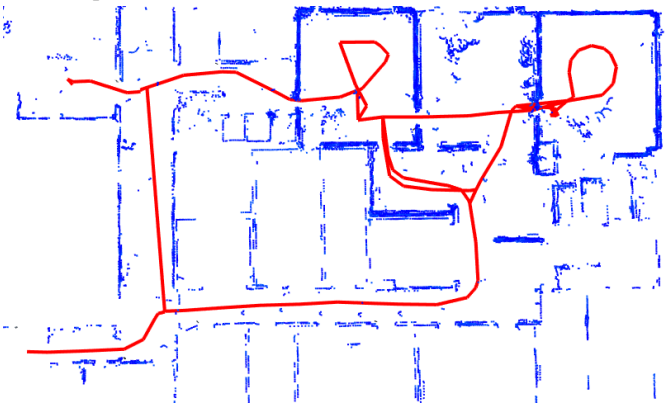
Figure 5.5 Subfigures (a) and (b): The difference between a false and true value in point 6.

Construction Site

On the construction site, a map was recorded in order to try the robotised guard tour method in a more dynamic environment. The original map is shown in Figure 5.6a, and additional edges were manually added as described in Section 4.3. The resulting map, which was later used for conducting the robotised guard tour, can be viewed in Figure 5.6b.



(a) The originally recorded map on the construction site and the recorded path as a solid black line.



(b) The final manipulated map for the construction site and the new possible path as a solid red line.

Figure 5.6 Subfigures (a) and (b): The difference between the original map and the manipulated map on the construction site.

The inspection tour on the construction site has a total of three inspection points, where the robot also changes its route at each and every one of those. In Table 5.3, the four different routes conducted on the site and where they have detected a person or not are visualised. This table indicates whether the robot detected a person or not. "True" means it detected a person, and "False" means that no person was detected. The computer vision model successfully identified if there was a person or not in each round executed.

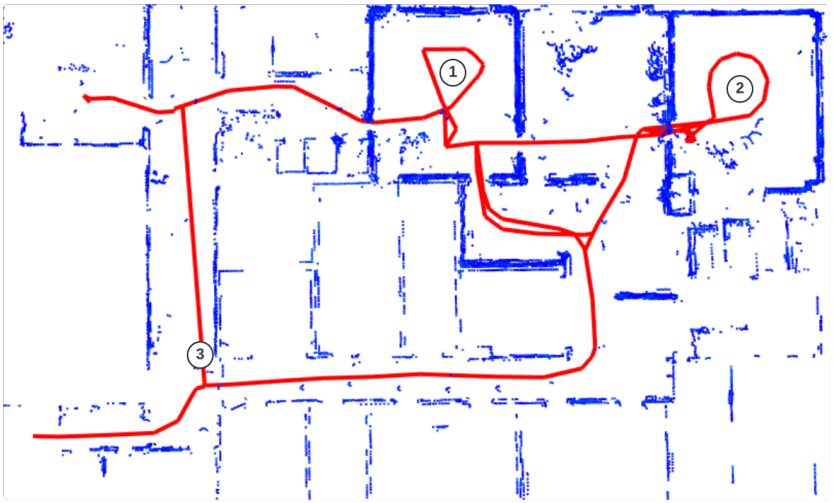


Figure 5.7 The points are numbered where the robot runs its detection and can take different routes.

Table 5.3 Results for detections of the rounds.

Point \ Round	1	2	3	4
1	False	False	True	True
2	False	False	True	False
3	True	False	True	False

Depending on where the detection has been made and how many detections there have been, the times for each round vary. It can be seen in Table 5.4.

Table 5.4 Execution time for each round at the construction site.

Round	**1**	**2**	**3**	**4**
Time (min:sec)	2:32	2:09	3:08	2:25

After execution, round 2 represents the base route, which means the route the robot would take if no person were detected. It is visualised in black in Figure 5.8, while the red represents route 3, which detected a person at each inspection point. All routes and how they look dependent on where a person has been detected can be found in Section A.3. In the same way as for the M-building scenario, these graphs

have been collected by extracting the coordinates from the robot's kinematic state relative to the 'gpe' frame.

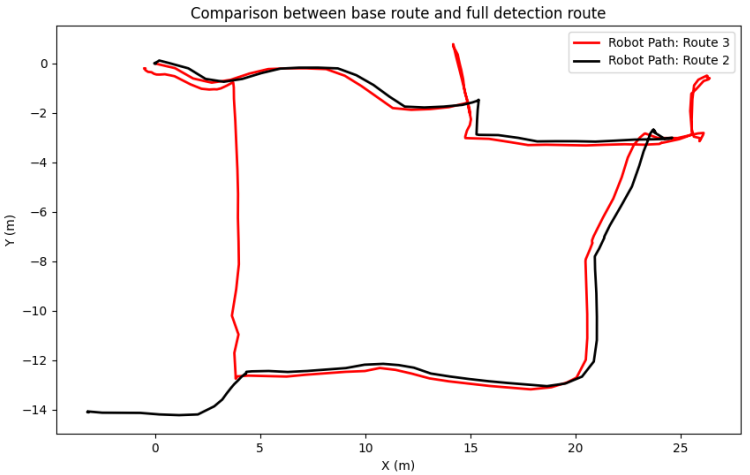


Figure 5.8 The base route is represented in black when the robot does not detect any people, versus the full detection route in red when the robot detects a person at each inspection point.

5.2 Full Behaviour Tree

In this section, the resulting behaviour tree for the M-building and the construction site will be presented. The BTs presented are the exact BTs that was executed for every round in the M-building respectively the construction site.

M-Building

In Figure 5.9, the BT for the M-building can be seen. This base structure includes one Inspection Without Action, two Two Way Options and three Inspections. Each one of those subtrees can be found in Section A.2.

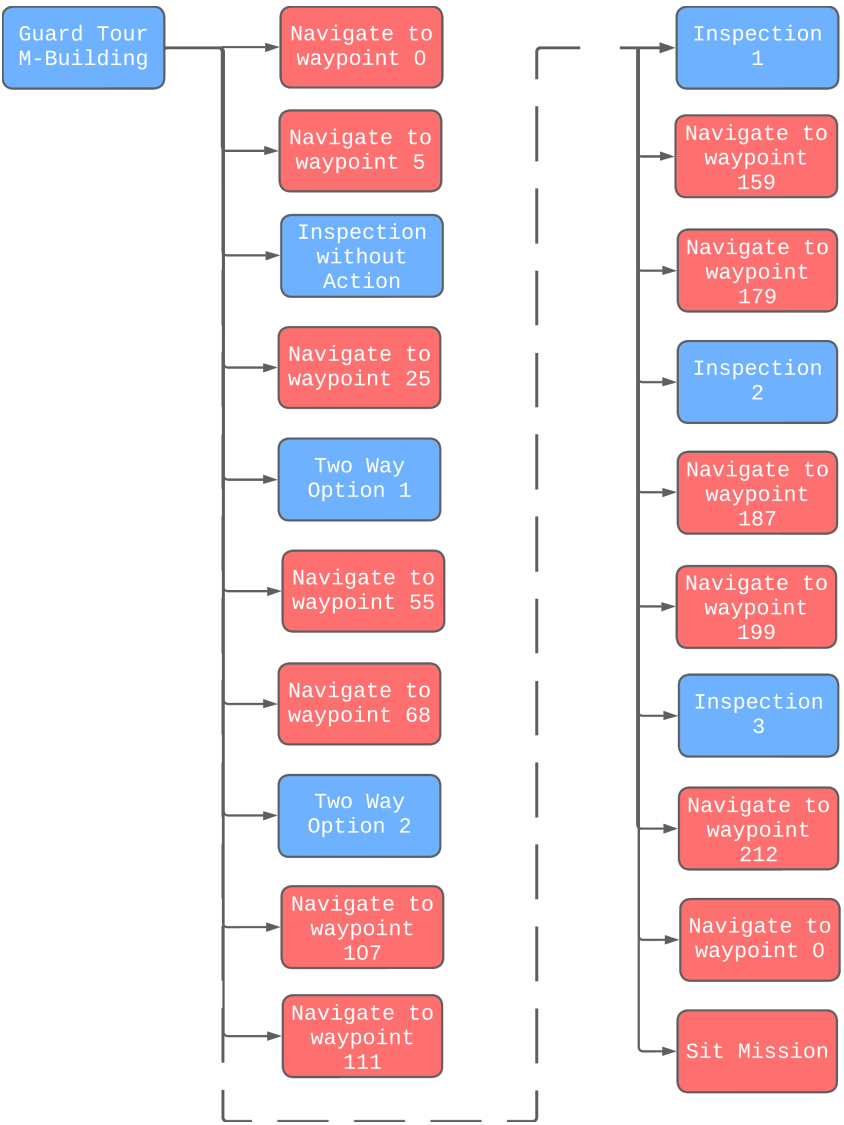


Figure 5.9 The behaviour tree base structure for the inspection route in the M-building.

Construction Site

In Figure 5.10, the full behaviour tree executed at the construction site can be seen. This structure includes three different Inspection actions.

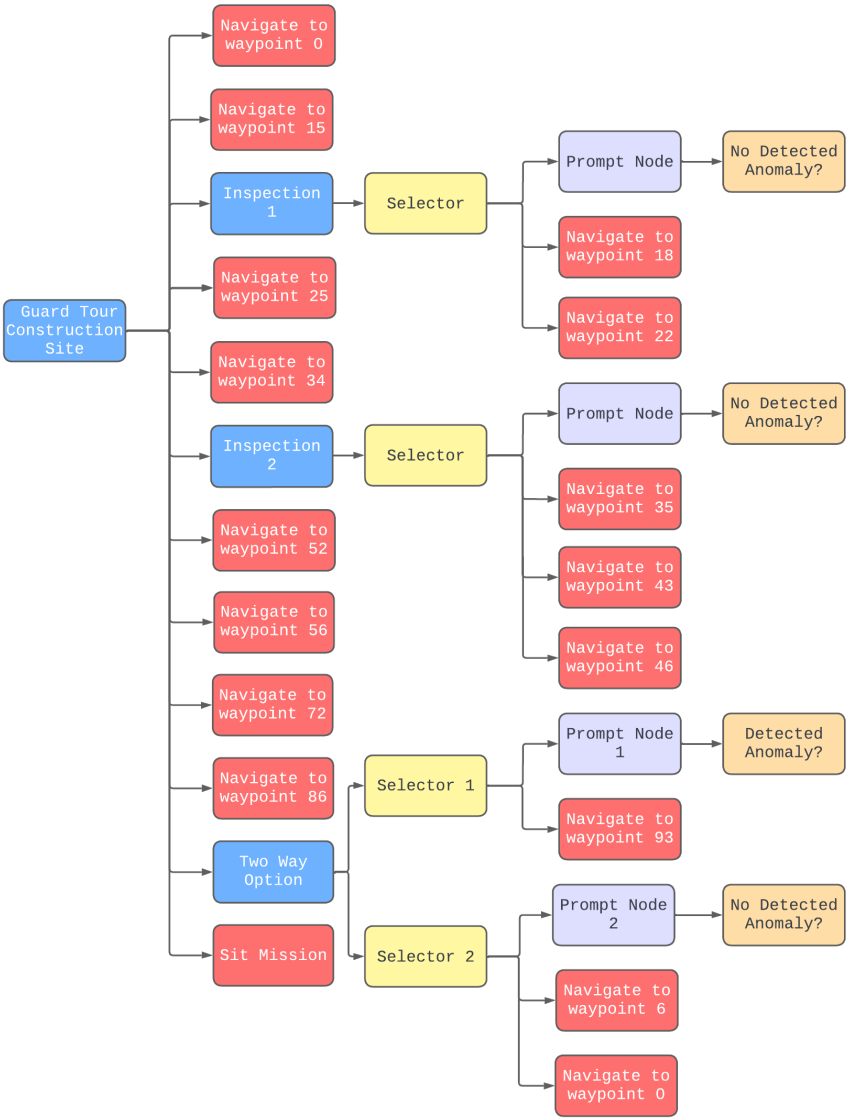


Figure 5.10 The full behaviour tree for the inspection route at the construction site.

6

Discussion

The final result in Chapter 5 gives a first approach on how to integrate a mobile robot, such as Spot, into a surveillance system by using a behaviour tree. There are many other ways this could have been solved, as well as further improved, but the goal of changing the robot's initial path dependent on an external camera is achieved. The method for achieving this goal can be replicated and used for other environments and purposes. Compared to only using a recorded autowalk that can later be replayed, this system offers a more dynamic approach where communication with the robot can happen during execution. This approach breaks the sequential 'go here, do that' behavior that autowalks automatically inherit.

Behaviour Tree

The BT provides a structured method for implementing a sequence of actions, offering easier implementation and beneficial modularity. Even if each location needs to be handled separately, using a BT and the different subtrees makes the process of building a robotised guard tour easier and more manageable, and the different types of actions could be transferred between maps and integrated into the full BT with smaller modifications. This can be seen in Figure 5.9 compared to Figure 5.10, where the inspection subtrees have been created in the same modular way, even if the map is different. In Figure 5.10, all the inspections have the exact same type of structure; the only thing that differs is the NavigateTo nodes between them. Therefore, using a BT proved to be an efficient way to accomplish this task. Although everything in this behaviour tree can be accomplished with a Python script, this would rapidly become too large to manage effectively. When working with Spot, missions can also be recorded one by one and then seamlessly integrated into a BT, a capability not supported in a script in the same way. This capability is, however, limited when a map is needed for the mission; then everything needs to be recorded on the same map. For a mission fully unrelated to a map, this is however possible to do. The next step would be to further automate the workflow process of generating the final BT, which is further discussed in Section 6.1.

Performance

During execution, both on the construction site and in the M-building, the robot performed as expected. At each inspection point where a person was detected, the robot correctly altered its route. The same is true when the robot does not detect a person; it continues its route as planned. This result can be seen on each graph in Section A.1 and Section A.3, with its decision-making tables in Table 5.1 and Table 5.3. In the less dynamic environment of the M-building, it was possible to test all types of subtrees, and the robot performed generally well. There were no major issues with the performance, even if the robot sometimes needed extra time to understand its position on the map. In the more difficult and dynamic environment, the resulting BT is smaller, and both Inspection 1 and Inspection 2 differ from the original route. This was a result of large changes in the construction between the recording of the map and the execution of the tour. The difference in how the robot walked can be seen in Figure 5.7 and Figure 5.8. The robot was unable to traverse in the original circle, and at one point Spot walked up on a pile of doors, which is not a desired behaviour. For example, a human-made inspection tour would not include such a behaviour, but as this is a first approach to accomplishing this with a BT, there might be future adaptations that could prevent this from happening. It should also be stated that the dynamic environment at the construction site also makes surveillance with fixed cameras more difficult, which is not that much of an issue at a non-dynamic location. However, the need to further investigate, react upon, and cover greater areas than what fixed cameras can achieve is still relevant in both types of locations.

As shown in Table 5.2, the execution time can vary even when the same path is executed. This variation is expected, as the robot might need extra time to avoid obstacles and to localize and initialize itself towards a fiducial. It is also important to note the time saved by not traversing the entire route each time, which would have been necessary if the robot followed a recorded autowalk without deviation. When Spot encounters obstacles, it can navigate around them, provided the deviation from the original route is not too extensive. To enhance the system's robustness and success rate, additional edges could be added to the map, giving the robot multiple options to reach a waypoint. The robot would then initially choose the shortest path, but if it encounters an obstacle that it can not get around, it would select an alternative route to reach the destination. This will be further discussed in Section 6.1.

In Chapter 5, it is possible to see that the map, after adding additional edges, also changes in whole. This can be seen in Figure 5.1a and Figure 5.1c. The manipulated map differs slightly from the original map. This means that the way presented as a solution for generating new edges might create an unknown behaviour for the robot; however, during the resulting surveillance tour, it did not seem to affect the robot's behaviour. This was true for both the locations where the results were gathered. To

fully investigate the performance of the robot after changes have been made to the original map, additional extensive testing should be done in different environments.

Computer Vision

The computer vision model employed serves solely to showcase how such technology can be integrated into a system. In the operational environment of this project, the detection of people is the primary focus, and the thermal camera achieves this with over 80% confidence during execution as shown in Figure 5.4b and Figure 5.5b. It is essential to note that this confidence level may vary based on factors such as the model's training data, which is specific to the environment. Consequently, the model has not been tested in new settings. While enhancing the dataset with more diverse data could improve results, given the constraints of this project, the current outcome is satisfactory.

During the process, communication between the Supervisor, Spot, and the camera was precise; however, the communication link was slow. The YOLOv8 model required significantly more time to process frames captured from the camera when it was connected to Spot's Ethernet port and the supervisor to the WiFi than when the camera was connected directly to the Supervisor. The Prompt node did not send a question at the wrong time or missed an answer at any point of this thesis, meaning that this way of communicating is stable but not ideal if time is a limiting factor. The robot did, however, only look for people when it was still, which meant that the delay did not affect the outcome. If the detection should be continuous during the full tour, ways to speed up the data transfer should be investigated.

6.1 Future Work

The first thing that can be done as future work concerning the considered problem is to further extend the behaviour trees by including more nodes and covering greater areas. The extension could be developed in the same way as the final BTs in this project. For example, when the robot encounters an obstacle or becomes disoriented in certain locations, it interrupts, finishes the mission, and sits down in that location. In order to enhance the robustness of the BT, implementing the condition *If possible - Navigate to - Else go back* would be beneficial and help avoid scenarios where the robot enters a room where it cannot traverse. It is also possible to improve the CV model, both when it comes to detecting people and for more use cases, such as open doors, overheated equipment, or other unusual behaviours. Right now, the CV model only works as a tool for investigating how the robot could react when anomalies are detected; this area has the potential for extension and refinement. Along with improving the CV model, the reaction from the robot could also be more realistic and useful. For example, if a door that should be closed is opened, this is something that the robot could fix upon detection. Right now, the arm is not

moved during the execution of the BT. This is, of course, an extension that could further improve the BT, but because of limited time, this was not further considered.

Apart from extending the BT, there is also the potential to make it more dynamic. As stated in the discussion, the current solution is dependent on a pre-recorded map and would not work in an unknown environment. During the execution of the full mission, the various options that the robot can pursue are known in advance, meaning that there are a finite number of different paths that the robot could take. There are two main investigations that could be made on this topic. The first one is how to generate an edge outside of the GraphNavRecordingService on the robot. The second is how to change the BT during execution so that the path is not as predefined as the current solution. There might be the option to use the blackboard functionality with BosdynRobotCommand nodes to achieve this. Depending on where an anomaly is detected, a supervisor could send a command to the blackboard that the BosdynRobotCommand could listen to and use for walking around. To build on this concept, the anchoring optimisation function could be applied to include a blueprint or Building Information Modelling (BIM) model in the system. The user could then send a NavigateToAnchor command to get to a position (x,y,z) relative to the seed frame. This would mean that the robot would follow its path to the waypoint closest to that position and then continue travelling on a new, unknown edge to reach its desired destination. Currently, there is no other information on how to combine this into the BT, and because of the limited time frame for this project, this has not been investigated. It is also important to note that many times, having strict control over where the robot can walk is needed and wanted, meaning that making the robot walk outside of its decided route might not be desirable.

Boston Dynamics do not provide a simulation of Spot; however, there is one open-source solution available in Webots if using ROS2 [MASKOR, 2024]. Boston Dynamics recommend using the SDK directly, but they do refer to a ROS2 wrapper developed in collaboration between the Mobile Autonomous Systems & Cognitive Robotics Institute (MASKOR) at FH Aachen and the Boston Dynamics AI Institute. [Kirsch et al., 2024] The documentation on how to use the ROS2 wrapper is limited, but since it is derived from a solution in ROS1, the documentation from this one could assumably be used. The suggested solution of using prerecorded maps and building a BT from that could, in the future, be transferred to a solution in ROS2. The functionality would be the same, but it could be beneficial to use a middleware for a more modular solution. Here, it is important to state that the simulation cannot use exactly the same code as is needed directly by the robot, but it could still be useful as an extra tool for testing. Using ROS2 could simplify including more than one robot and also offer an easier way to build upon relevant open-source solutions. It can also can make it easier to include a blueprint or BIM model and align that with the robot's path.

7

Conclusion

This thesis has successfully developed and demonstrated a method for defining a robotic guard tour that dynamically adapts its route in response to detected anomalies. Addressing the research question, "How can a robot change its security path in real-time depending on detected anomalies?" the system uses behaviour trees to facilitate real-time decision-making. This enables the robot, equipped with an external thermal camera, to communicate with a supervisory Python script during execution and to adjust its patrol route based on the data received.

The system was tested in two distinct environments: the dynamically changing Vipan construction site and the constant M-building. In both settings, the robot demonstrated its capability to effectively modify its patrol paths. Importantly, while the robot's route options are finite, they are not limited to a single predefined path. This adaptability showcases the utility of behaviour trees in managing robotic behaviours in varied conditions, allowing for multiple but defined patrol routes.

Future work should focus on expanding the behaviour trees to include more nodes and cover a broader range of scenarios, improving the robot's responsiveness to diverse anomalies. Additionally, the potential to make the behaviour trees more dynamic, including the ability to navigate outside of pre-recorded maps and react adaptively to unexpected obstacles, can be explored. The use of ROS2 and potential integration with BIM models could further align the robot's navigational paths with architectural blueprints, enhancing precision in patrol routes and expanding the system's applicability in unstructured environments.

A

Appendix

This chapter will first provide all graphs for the resulting robotised guard tours in the M-building. The final subtrees from the tours executed in the M-building will also be visualised. Lastly, all the graphs from the construction site will be presented.

A.1 M-Building Graphs

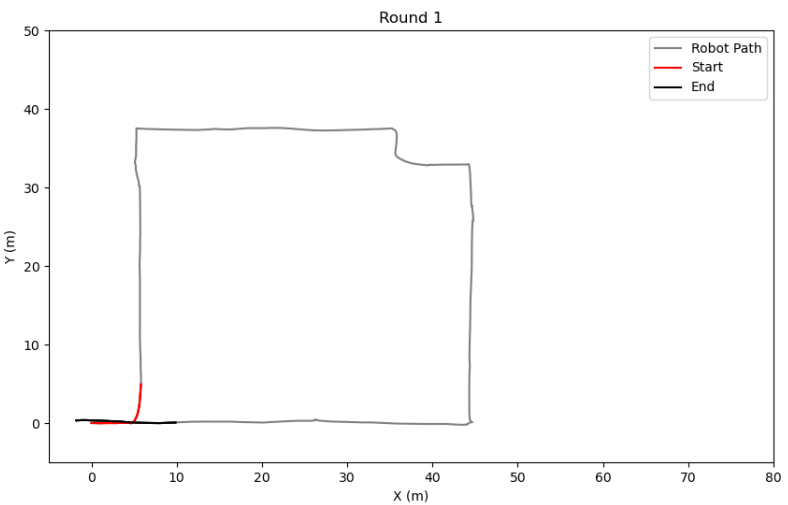


Figure A.1 Round 1 at LTH.

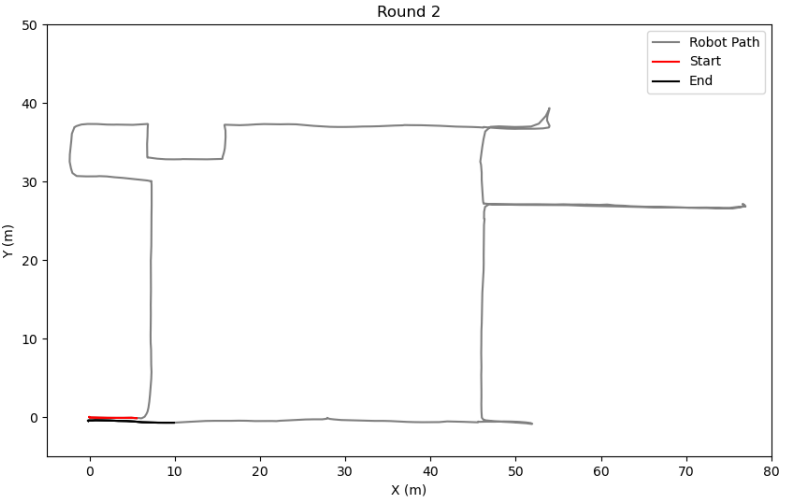


Figure A.2 Round 2 at LTH.

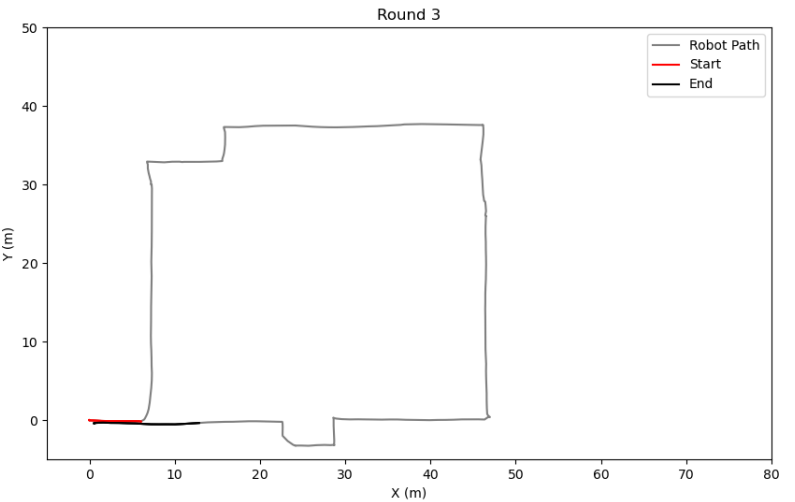


Figure A.3 Round 3 at LTH.

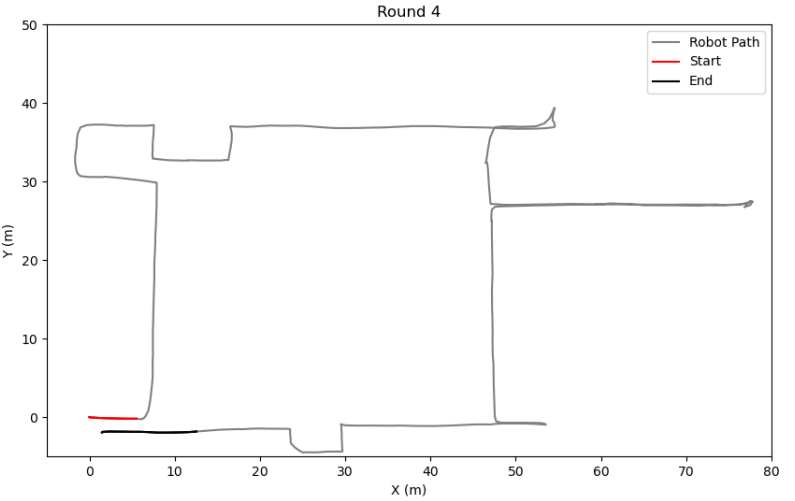


Figure A.4 Round 4 at LTH.

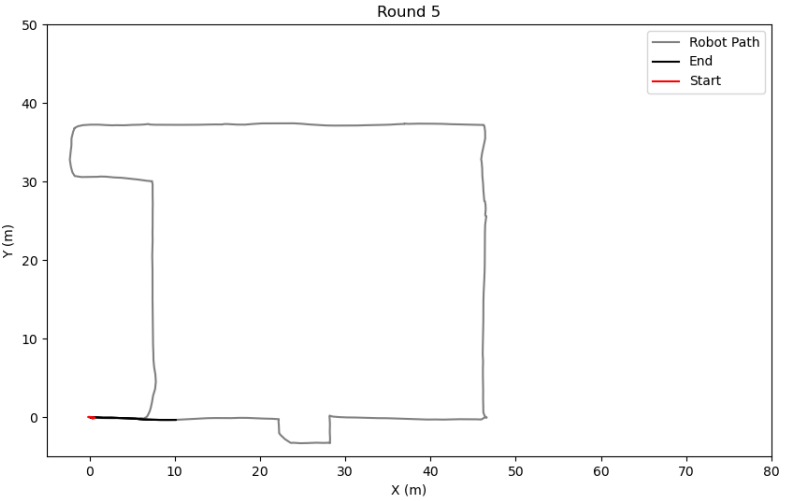


Figure A.5 Round 5 at LTH.

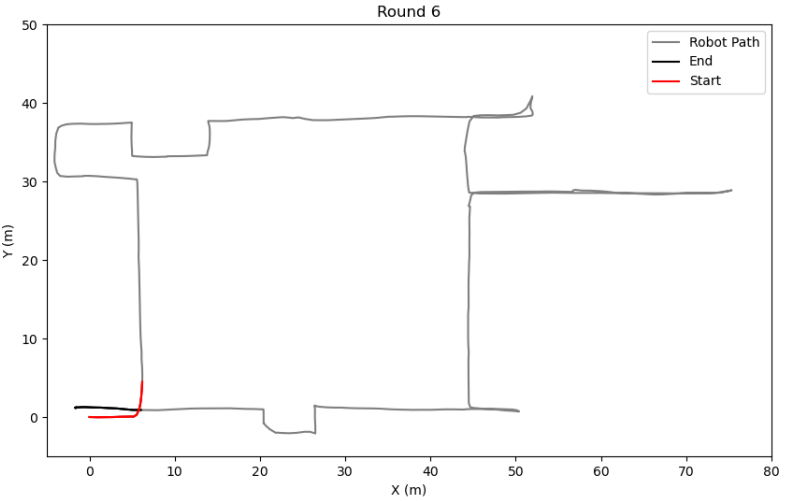


Figure A.6 Round 6 at LTH.

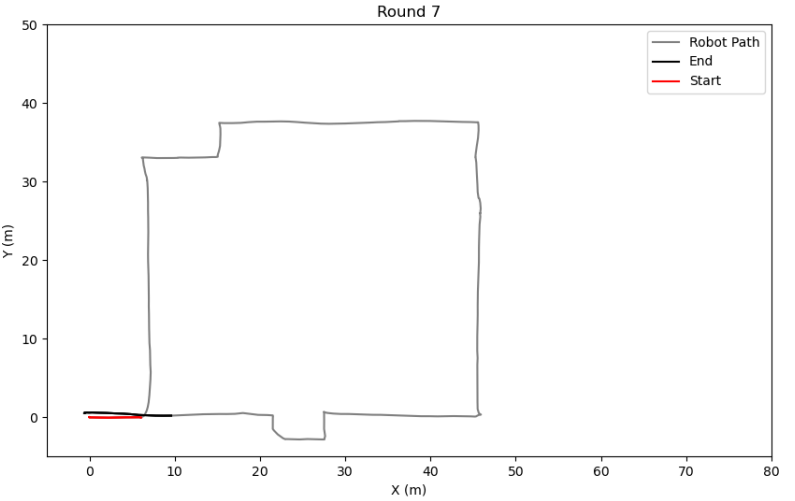


Figure A.7 Round 7 at LTH.

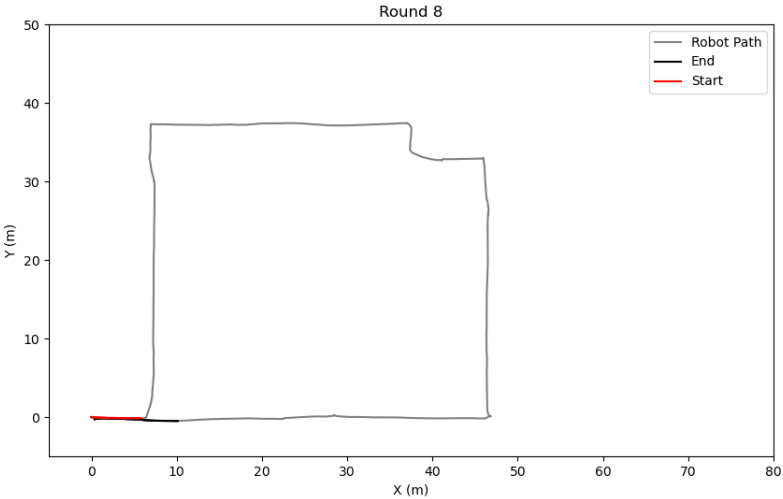


Figure A.8 Round 8 at LTH.

A.2 Subtrees

The Inspection without action subtree can be seen in Figure A.9.

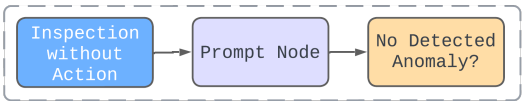


Figure A.9 The inspection without action subtree in the behaviour tree for the M-building.

The first two-way option subtree can be seen in Figure A.10. This subtree itself includes a Two Way Option and an Inspection.

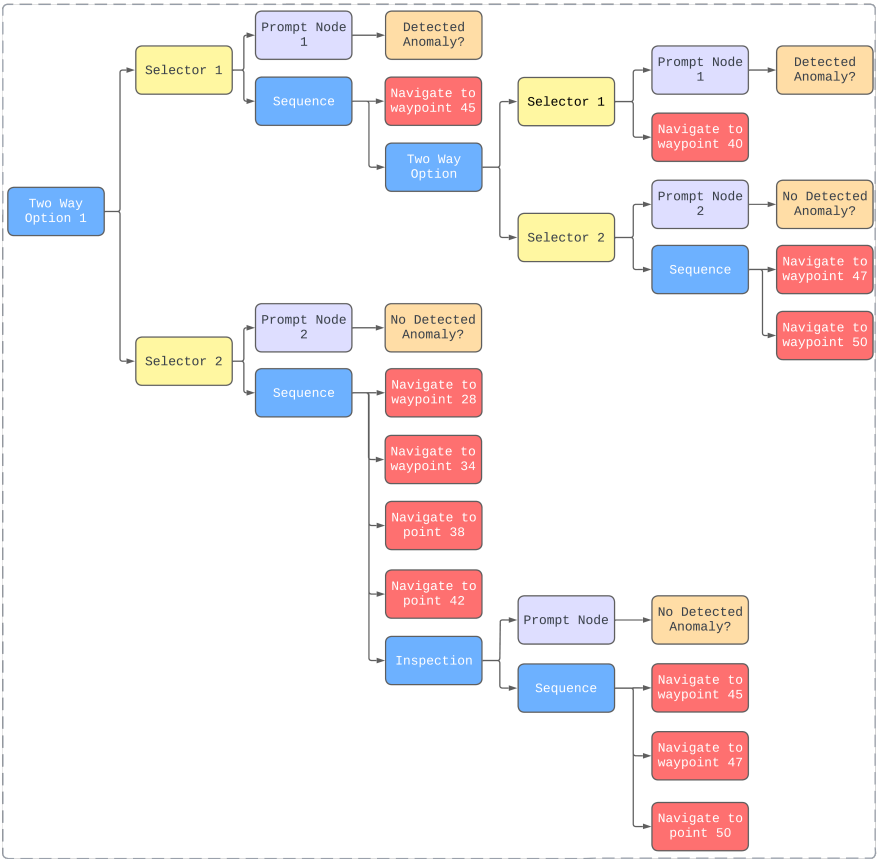


Figure A.10 The subtree for Two Way Option 1.

The second two-way option subtree can be seen in Figure A.11. This subtree includes an additional Inspection.

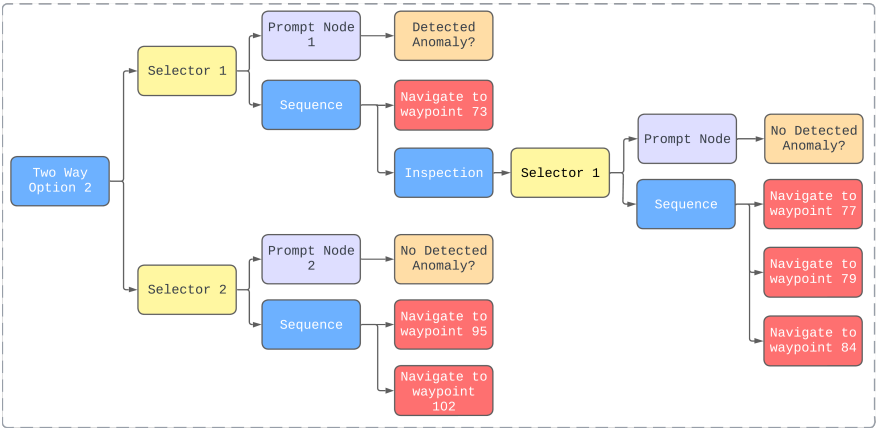
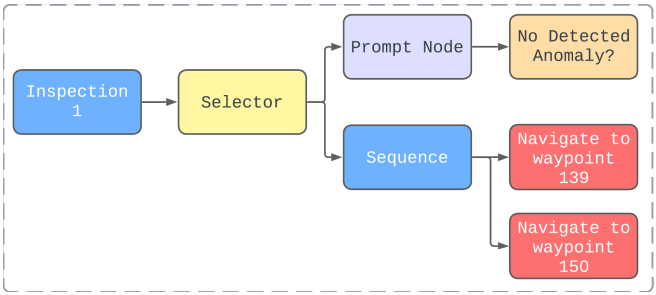
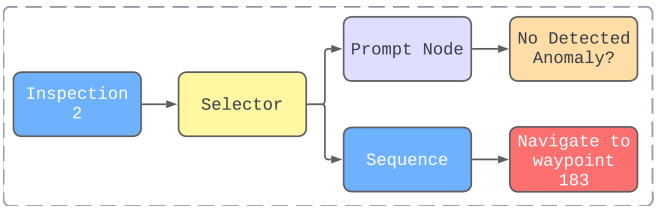


Figure A.11 The subtree for Two Way Option 2.

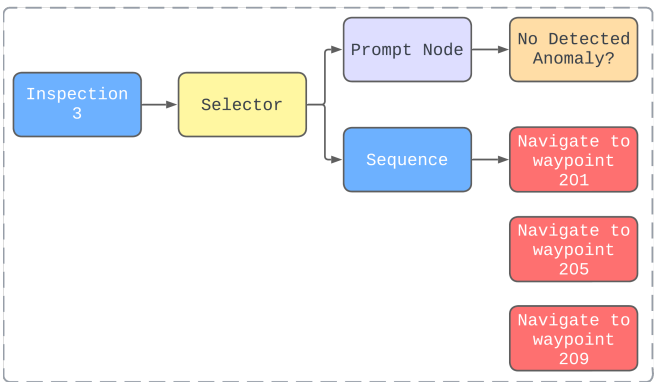
All the inspection subtrees from the base structure can be seen in Figure A.12.



(a) The originally recorded map on the construction site.



(b) The final manipulated map for the construction site.



(c)

Figure A.12 Subfigures (a), (b) and (c): The Inspection subtrees from the base structure of the BT.

A.3 Construction Site

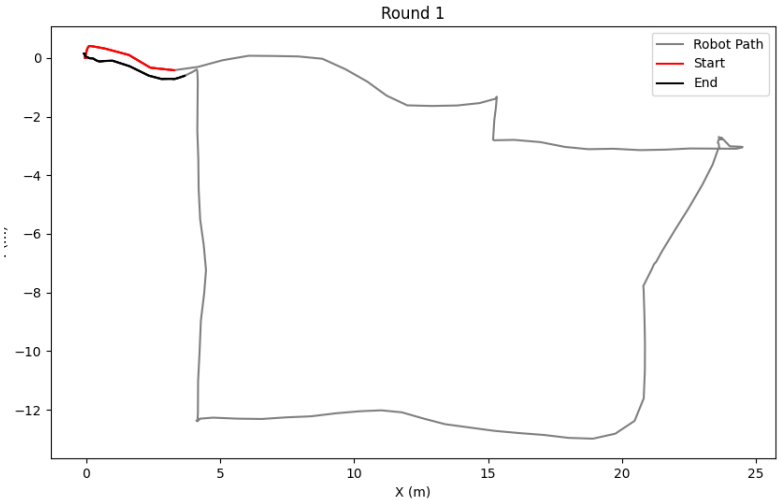


Figure A.13 Round 1 at the Construction Site.

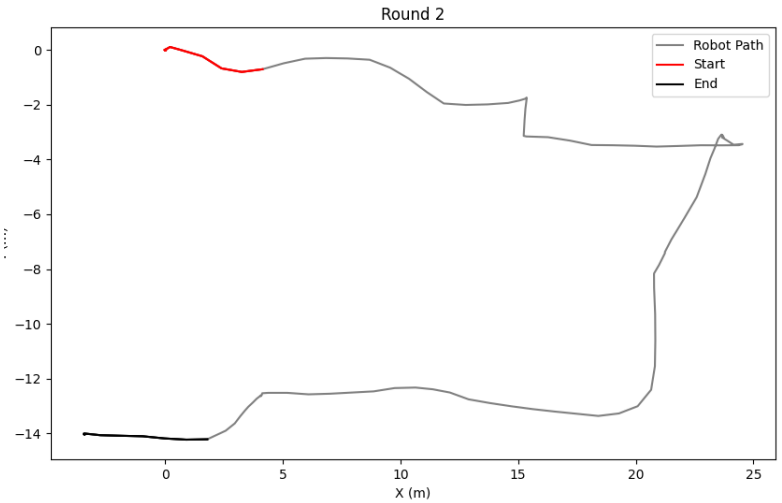


Figure A.14 Round 2 at the Construction Site.

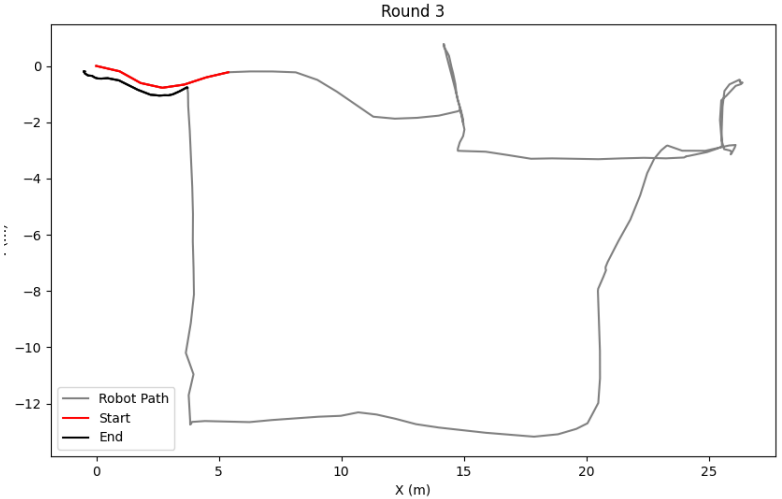


Figure A.15 Round 3 at the Construction Site.

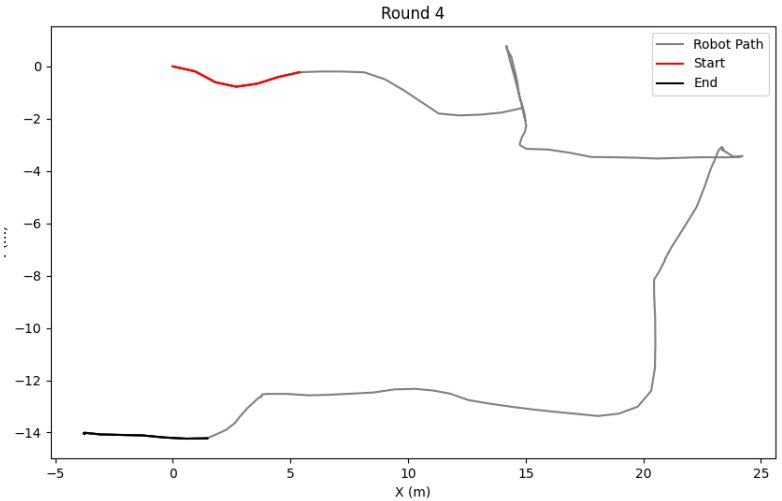


Figure A.16 Round 4 at the Construction Site.

Bibliography

Aalborg University (2024). *Trimodal People Segmentation Dataset*. https://www.kaggle.com/datasets/aalborguniversity/trimodal-people-segmentation/data.

Andersson, N. (2023). *Developing High-level Behaviours for the Boston Dynamics Spot Using Automated Planning*. LIU-IDA/LITH-EX-A–23/049–SE. Master's Thesis. Linköping University, Department of Computer and Information Science, Linköping, Sweden.

Ansari, S. and P. S. Salankar (2017). "An overview on thermal image processing". *Annals of Computer Science and Information Systems* **10**, pp. 111–120. URL: https://annals-csis.org/Volume_10/drp/pdf/111.pdf.

Arbetsmiljöverket (2023). *Riskutsatta branscher*. Accessed: 27-05-2024. URL: https://www.av.se/arbetsmiljoarbete-och-inspektioner/arbetsgivarens-ansvar-for-arbetsmiljon/upphandling-och-arbetsmiljo/riskutsatta-branscher/.

Axis Communications (2024a). *Axis fa1080-e thermal sensor unit*. Accessed March 26, 2024, from https://www.axis.com/products/axis-fa1080-e-thermal-sensor-unit.

Axis Communications (2024b). *Axis fa1105 sensor unit*. Accessed March 26, 2024, from https://www.axis.com/products/axis-fa1105-sensor-unit.

Axis Communications (2024c). *Axis fa54 main unit*. Accessed March 26, 2024, from https://www.axis.com/products/axis-fa54-main-unit.

Boston Dynamics (2024a). *About Spot*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/docs/concepts/about_spot.

Boston Dynamics (2024b). *About the Spot robot*. Accessed April 4, 2024, from https://support.bostondynamics.com/s/article/About-the-Spot-robot.

Boston Dynamics (2024c). *Autowalk service*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/docs/concepts/autonomy/autowalk_service.

Boston Dynamics (2024d). *Concepts*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/docs/concepts/readme.

Boston Dynamics (2024e). *Graphnav service*. Accessed: 08-05-2024. Boston Dynamics. URL: https://dev.bostondynamics.com/docs/concepts/autonomy/graphnav_service.

Boston Dynamics (2024f). *Graphnav technology summary*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/docs/concepts/autonomy/graphnav_tech_summary.

Boston Dynamics (2024g). *Mission service*. Accessed: 08-05-2024. Boston Dynamics. URL: https://dev.bostondynamics.com/docs/concepts/autonomy/missions_service.

Boston Dynamics (2024h). *Networking*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/docs/concepts/networking.

Boston Dynamics (2024i). *Spot arm: end-effector payload specification*. Accessed: 06-05-2024. URL: https://support.bostondynamics.com/s/article/Spot-Arm-End-effector-payload-specification.

Boston Dynamics (2024j). *Spot SDK*. Accessed: 07-05-2024. URL: https://dev.bostondynamics.com/.

Boston Dynamics (2024k). *Spot the agile robot*. Accessed: 04-04-2024. URL: https://bostondynamics.com/products/spot/.

Colledanchise, M. and P. Ögren (2017). *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, Boca Raton, FL, USA.

De Santis, F. (2023). *Autonomous Mobile Robots: Configuration of an Automated Inspection System*. Master's Degree in Mechatronic Engineering. MA thesis. Politecnico di Torino, Turin, Italy.

FLIR Systems (2024a). *Picking a thermal color palette*. FLIR. Accessed: 27-05-2024. URL: https://www.flir.com/discover/industrial/picking-a-thermal-color-palette/.

FLIR Systems, I. (2024b). *Flir adas dataset*. Accessed: 27-05-2024. URL: https://www.flir.com/oem/adas/adas-dataset-form.

gRPC (2024). *About grpc*. Accessed: 05-04-2024. URL: https://grpc.io/about/.

Gupta, S., P. Sharma, D. Sharma, V. Gupta, and N. Sambyal (2020). "Detection and localization of potholes in thermal images using deep neural networks". *Multimedia Tools and Applications* **79**:35, pp. 26265–26284. ISSN: 1573-7721. DOI: 10.1007/s11042-020-09293-8.

IFR (2021). “A mobile revolution - how mobility is reshaping robotics”. *IRF Journal*.

Kili Technology (2024). *YOLO Algorithm: Real-Time Object Detection from A to Z*. Accessed: 03-04-2024. URL: https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z.

Kirsch, M., S. Pawar, C. Gollok, S. Schiffer, A. Ferrein, J. Barry, D. Gonzalez, T. Pang, D. Surovik, J. Wang, D. Watkins, and T. Persson (2024). *Spot ROS2 driver*. Accessed: 02-05-2024. URL: https://github.com/bdaiinstitute/spot_ros2/tree/spot-sdk-3.3.2.

MASKOR (2024). *Webots ROS 2 Spot*. GitHub. URL: https://github.com/MASKOR/webots_ros2_spot.

Mazzeo, P. L., (Ed.) (2023). *Intelligent Video Surveillance New Perspectives*. IntechOpen, London, United Kingdom. ISBN: 978-1-80356-341-1. DOI: 10.5772/intechopen.100777. URL: http://dx.doi.org/10.5772/intechopen.100777.

Munian, Y., A. Martinez-Molina, and M. Alamaniotis (2023). “Comparative analysis of thermogram and pre-processed hog images using machine learning classifiers”. In: *2023 14th International Conference on Information, Intelligence, Systems Applications (IISA)*, pp. 1–8. DOI: 10.1109/IISA59645.2023.10345890.

Nilsson, F. (2023). *Intelligent Network Video: Understanding Modern Video Surveillance Systems*. 3rd ed. Book. URL: https://doi.org/10.4324/9781003412205.

Nilsson, O. (2022). *Building Dense Reconstructions with SLAM and Spot*. MSc Thesis TFRT-6158. MA thesis. Lund University, Lund, Sweden.

Tsai, P.-F., C.-H. Liao, and S.-M. Yuan (2022). “Using deep learning with thermal imaging for human detection in heavy smoke scenarios”. *Sensors* **22**:14. ISSN: 1424-8220. DOI: 10.3390/s22145351. URL: https://www.mdpi.com/1424-8220/22/14/5351.

Tu, M. D., K. T. Le, and M. D. Phung (2024). “Object detection in thermal images using deep learning for unmanned aerial vehicles”. DOI: 10.1109/sii58957.2024.10417611. URL: http://dx.doi.org/10.1109/SII58957.2024.10417611.

Ultralytics (2024). *YOLOv8 Overview*. Accessed: 03-04-2024. URL: https://docs.ultralytics.com/models/yolov8.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2024

Document Number
TFRT-6237

Author(s)
Johanna Häggström Wedding
Ella Thunborg

Supervisor
Björn Olofsson, Dept. of Automatic Control, Lund University
Mathias Haage, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Robotised Guard Tours in Security Systems

Abstract

With the growing market for enhanced security combined with recent advancements in robotic technology, the possibility of integrating these two fields is of considerable interest. While traditional surveillance relies on mounted cameras and scheduled guard tours by humans, the next step would be to use a robot to complement a person in such systems. This project explores the integration of autonomous robotics into security systems by developing and implementing a robotic guard tour using Boston Dynamics' Spot robot. An external thermal camera and computer vision algorithms were used to monitor the environment and respond to anomalies. Utilising Spot's existing capability, the primary objective was to evaluate the robot's ability to dynamically alter its pre-defined inspection route based on real-time sensor input, specifically the detection of anomalies.

Results, gathered at the Mechanical Engineering building at LTH and the Vipan construction site in Lund, show that the robot effectively adjusted its routes in response to detected anomalies, demonstrating enhanced surveillance capabilities. The findings in different settings—LTH's controlled environment versus the dynamically changing conditions at the construction site—proved that the robot could successfully conduct guard tours under both constant and variable conditions. The use of behaviour trees enabled decision-making and route management, suggesting potential improvements for future deployments to achieve more dynamic and autonomous surveillance operations. This thesis highlights the adaptability of autonomous robotic systems in complex and dynamic environments.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-73

Recipient's notes

Security classification

http://www.control.lth.se/publications/