Streamlining Robotic Development: Cost and Time Savings through Digital Twins and Simulations

A thesis presented in partial fulfillment of the requirements for the degree of

Master of Engineering

in

Mechatronics

by
Craig Martin-Smith

Supervisors:
Prof. Johan Potgieter
Dr. Gabe Redding

School of Food and Advanced Technology
Massey University
Palmerston North, New Zealand

May 28, 2024

Acknowledgements

I would like to extend my sincerest gratitude to my supervisors, Professor Johan Potgieter and Dr Gabe Redding, for his unwavering guidance and support throughout my research journey. I am also deeply indebted to Massey Ventures Limited, the Ken and Elizabeth Bursary committee, and the Massey University Master's Research Bursary committee for their generous financial support, which has been instrumental in facilitating my studies.

My heartfelt thanks also go to my colleagues for their collaboration and engaging discussions, which have significantly enriched my research and learning experience.

Lastly, I would like to express my deepest appreciation to my family and partner for their unconditional love, patience, and unwavering support, which have been my primary source of motivation throughout this journey.

Each of you has played a vital role in making this achievement possible, and I am eternally grateful for your contributions.

Abstract

This thesis explores the integration of Digital Twins (DTs) and advanced simulations in robotic development, underscored by recent GPU technology advancements that enhance high-fidelity simulations' viability. The study focuses on autonomous mobile robots and aims to address the limitations of traditional field testing, which includes high costs and substantial environmental impacts. The research utilises DTs—accurate virtual replicas of physical robots—to improve development efficiency, reduce reliance on physical prototypes, and enhance cost-effectiveness.

Featuring four detailed case studies — a line painting robot, an indoor camera-based localisation system, a forestry tool carrier, and an inspection robot—this research illustrates the effectiveness of Digital Twins in verifying and refining robot functionalities prior to physical implementation. The case studies reveal significant efficiencies: the line painting robot saw a time reduction of one month and savings of NZD 8, 080; the localisation system achieved a 44% reduction in costs, saving NZD 9, 380; the forestry robot reduced costs by 20.84%, saving NZD 42, 500; and the inspection robot shortened its development timeline by two months, resulting in savings of NZD 19, 280.

Overall, the results affirm that DTs and simulations, bolstered by GPU enhancements, can significantly accelerate development timelines and enhance the precision and quality of robotic systems. This thesis substantiates the transformative impact of DTs and simulations in robotic development, highlighting their crucial role in advancing robotic systems capable of operating in diverse and challenging environments.

Contents

List of Figures

CHAPTER 1

Introduction

In the rapidly evolving field of mobile robotics, field testing has traditionally played a crucial role in validating and refining robotic systems. This process, essential for ensuring robots' practical applicability and reliability, involves extensive real-world scenario testing throughout development to iron out any operational flaws. However, traditional field testing faces substantial challenges with the diversification of application areas and the increasing complexity of robotic systems. These challenges include high logistical costs, time consumption, and potential safety risks in unpredictable environments. With the rising demand for sophisticated and versatile robots, alternative methodologies able to enhance development efficiency and reduce dependency on physical prototyping and field testing are critical to the future of robotics development [1], [2]. While necessary, this approach incurs significant financial and time costs, often resulting in extended development cycles and increased environmental footprints.

Recent advancements in hardware for accelerated computing, such as GPUs, have opened new avenues for high-fidelity simulations [3], enabling the integration of far more complex and comprehensive systems into the simulated world. Digital Twins (DTs), detailed and data-linked virtual copies of real systems, and robot simulation technologies have emerged as promising tools, offering detailed virtual replicas of physical systems and environments. One study by [1] showed that simulations could reveal most of the bugs analysed in outdoor robot navigation, and others have shown that these methods are effective for flying vehicles [4]. Despite the evident potential of DTs and simulation technologies, the robotics development cycle still predominantly adheres to traditional methodologies, which are marked by inefficiencies and high costs [5].

This research aims to explore the integration of DTs and advanced simulations in robotic

development to streamline the development process, reduce costs, and enhance efficiency. Objectives included examining existing DT and simulation integration methodologies, identifying bottlenecks in current workflows, and assessing the impact of these technologies through real-world case studies. Ultimately, this research attempts to convey the benefits of integrating the rising technologies of DTs and simulations into robotic development and quantifying the potential benefits of such integration compared to traditional methodologies.

The research scope focuses on improving development process efficiency for autonomous mobile robots by integrating DTs and simulations rather than the robots' operational efficiency. It leverages four targeted case studies to provide a nuanced understanding of how these technologies can be applied across different scenarios in autonomous robotics:

- Case A - A Line Painting Robot DT and path planning validation.
- Case B - The evaluation of a QR code-based visual localisation system.
- Case C - A mechanical DT approximating the behaviour of a large forestry tool carrier.
- Case D - A DT of a 4-wheel inspection robot with ROS 2 integration.

The thesis is structured to address the research objectives and questions systematically. Following this introduction, the literature review will provide a comprehensive background on DT and simulation technologies in robotic development. The methodology is detailed in a general methods and tools chapter, establishing the foundational approaches and technologies used throughout the research, followed by four additional case study chapters. The four case study chapters discuss how these methodologies were specifically applied and adapted to each scenario. Subsequent chapters present the findings from these real-world applications and discuss the implications of these results. The final chapters summarise the research findings, outline the contributions to the field, suggest areas for future research, and conclude the study.

CHAPTER 2

Literature Review

In the context of this research, which aims to address the inefficiencies of traditional robotic development cycles through the integration of Digital Twins and advanced simulations, this Literature Review seeks to lay the foundational understanding necessary for evaluating the potential of these technologies in robotics. The literature review is structured to explore the current state and advancements within two critical areas: DTs and simulation technologies in robotics, both underpinned by the increasing demands for automation and enhanced computational capabilities.

This section will delve into existing methodologies, examining how DTs and simulations are currently utilised in robotic development and identify where these methods fall short or succeed in improving efficiency, reducing costs, and enhancing the development lifecycle. Moreover, it seeks to uncover the specific challenges and limitations faced in integrating these technologies into robotics, providing a context for the research questions this thesis intends to answer:

- How do DTs and advanced simulations influence the development cycles in robotics when compared to traditional methods?
- Combined with DTs, which simulation tools and environments yield significant benefits for robotic development?

This review intends to directly support this research's overarching goal: to demonstrate and advocate for integrating DTs and advanced simulations to revolutionise traditional approaches to robotic development. By systematically addressing these research questions and examining relevant literature, this review aims to provide a comprehensive backdrop against which the proposed innovations can be critically assessed.

2.1 Robotic Simulation

The domain of robotic simulations has undergone a significant transformation, fundamentally influencing the development, testing, and deployment of robotic systems. These early computer simulations, beginning in the 1970s and 1980s, were initially basic, focusing on model verification and programming with limited visualisations and kinematic models [6]. The 1980s marked a leap by introducing complex dynamics and control algorithms, enabling more accurate predictions of physical robot behaviour [7].

The introduction of 3D graphics into robot simulations is closely tied to the history of 3D modelling. The first significant application of 3D modelling in robotics was in the late 1970s and early 1980s when researchers developed computer-aided design (CAD) and computer-aided manufacturing (CAM) systems for industrial robots [8]. These systems allowed engineers to design, simulate, and test robot movements and tasks in a virtual environment before implementing them in the real world. Two early examples of such systems were the Robot Simulation System (RSS) developed by General Motors in 1979 [9] and the Robot Programming and Simulation System (RPSS) developed using IBM PCs in 1988 [10]. These systems used wireframe models and simple shading techniques to represent the robots and their environments.

The integration of 3D graphics in the late 1980s and the 1990s signified another milestone, allowing for detailed simulations of robot-environment interactions and industrial production environments [11]. This era saw simulations becoming more immersive, refining design and operational strategies. In the 21st century, real-time simulation capabilities were welcomed as crucial for autonomous robotics [12], [13], further advanced by artificial intelligence and machine learning in the 2000s [3], [14], [15].

A variety of open-source and commercial tools characterise contemporary robotic simulations. These tools cater to diverse applications, from surgical robotics to human-robot interaction and assistive environments [16], [17]. Advancements in physics engines and computational power have broadened their scope, offering deeper insights into the fundamental aspects of robotic systems [17], [18]. However, the abundance of these tools, each with unique capabilities, presents challenges in selecting the most suitable one for specific tasks [19]–[21].

Robotic simulations today are integral to ensuring safety, efficiency, and robustness in dynamic scenarios. They facilitate rapid prototyping and easy code portability from simulated to real environments, addressing the needs of complex robotic systems [17]. Their role in reducing the need for extensive physical testing has led to faster development cycles and cost reductions,

particularly in areas where real-world testing is impractical [22].

These advancements have propelled the robotics field forward and opened new possibilities for application and innovation. The following sections of this review will delve deeper into the specific tools, methodologies, and advancements in robotic simulations, exploring their roles in various sectors and potential future developments.

2.1.1 Simulation Tools and Software in Robotics

With the growing interest in robotics for physical operations, simulation is increasingly used for rapidly prototyping controllers, not simply offline computation and visualisation [19]. These robotic simulation tools, both open-source and commercial, have become critical resources in fields such as medical robotics, human-robot interaction, and assistive environments where robots work closely with humans [16]. These tools offer a practical and cost-effective platform for testing algorithms, ensuring safety, efficiency, and robustness in dynamic scenarios. Importantly, they also enable easy code portability from simulated to real environments, thus fulfilling the needs of complex systems, such as humanoid or kinematic chain robots [17].

Recent advancements in physics engines, video game engines, and computer processing speed have led to a surge in the development of versatile robotic simulators [18]. They offer a deeper understanding of robotic systems' fundamental physics and concepts, which may be challenging without hands-on experimentation and visualisation [17]. However, choosing the right simulation tool can often be challenging due to the vast array of options available, each offering different capabilities, dependencies, and maintenance requirements [17]. A few studies have specifically explored some available options, such as [19], [20] and [21].

Nevertheless, the value of simulation in robotics cannot be understated. The recognition of its importance continues to grow, suggesting a promising future for these tools and their ongoing development [16]–[18], [22].

2.1.1.1 General-Purpose Simulation Environments

General-purpose math-based platforms like MATLAB/Simulink [23], [24] and 3D game engines such as Unity 3D and Epic Games' Unreal Engine are adaptable for diverse applications. MATLAB/Simulink excels in mathematical modelling and control system design, which is helpful in early algorithmic development and more advanced mathematical processing. Unity and Unreal Engine are known for their 3D visualisations and physics engines, ideal for detailed and immersive simulations and visualisations [25]. While versatile, these tools typically require additional development for advanced robotic features, such as the framework developed by

Ganoni and Mukundan [26] using Unreal Engine 4 as their multi-robot simulation framework platform.

2.1.1.2 Robotics-Specific Simulation Environments

Dedicated robotic environments like Gazebo and ROS's RViz are tailored specifically for robotic applications. Gazebo offers a comprehensive simulation of complex robotic systems, including realistic physics and sensor emulation [27]. RViz provides real-time visualisation within the ROS ecosystem, crucial for monitoring and planning [28]. These platforms are designed to handle the requirements of robotic simulations, offering accuracy in kinematics and dynamics essential for robotic systems.

2.1.1.3 Physics and Visual Simulation in Robotics

Various physics engines, including Open Dynamics Engine (ODE) [28], NVIDIA's PhysX engine [29], Bullet [30], and DART [31], find widespread use. A user survey review by Ivaldi et al. [19] discussed these simulators' features and usability. The increased demand for precision in simulating modern robot dynamics has led to a somewhat fragmented field, complicating the selection of the right tool for a project or predicting the effort and performance outcomes [21].

The core of these simulations lies in their physics engines and visual fidelity. Physics engines like ODE and NVIDIA's PhysX, integrated into platforms like Webots [32] and Unity, respectively, ensure accurate emulation of real-world dynamics. Visual simulation, especially in environments like Unity, Unreal Engine, and Isaac Sim, offers realistic rendering crucial for testing and refining robotic designs [33], [34].

Physics engines are the bedrock of robotics simulations, enabling accurate emulation of real-world physics within the constraints of computational efficiency. Despite notable challenges like multi-disciplinary expertise requirements and simulation speed management, they are pivotal in developing safer and more efficient robots. This section delves into these physics packages and their collision detection methods, offering a comparative lens on various tools.

One study proposed new simulation performance metrics to address this issue [21] by using a new software framework to conduct extensive comparative simulations across multiple engines and discovered that each engine excels in its intended area. MuJoCo [35], explicitly designed for robotics, outperformed others in robotics-specific tests. Conversely, engines designed for gaming performed better in their domain. Their research recommended using ODE with a reasonable timeframe for applications involving wheeled robots and collisions.

In this complex landscape, Webots primarily utilises ODE for physics computations,

leveraging the Gilbert-Johnson-Keerthi (GJK) algorithm [36] for convex shape collisions and a custom algorithm for handling triangle mesh collisions. Unity and Unreal Engine, on the other hand, rely on NVIDIA's PhysX engine [29], using GJK for convex collisions and a triangle mesh collider for complex mesh collisions.

Gazebo Sim supports a range of physics engines, including ODE, Bullet, Simbody, and DART, which employ various algorithms depending on the simulation requirements. This versatility sets Gazebo apart from other platforms. Similarly, NVIDIA's Isaac Sim and the CARLA simulator utilise the PhysX engine, employing advanced features like GPU acceleration and real-time physics for complex robotic system development. The emphasis on physics accuracy in these platforms reflects their goal of modelling complex real-world scenarios involving robotic or autonomous systems.

2.1.1.4 Free Open and Closed-Source Packages

Gazebo, a pivotal element of open-source robotics simulation tools, is widely recognised in academia and industry for its comprehensive simulation capabilities. Initially part of the Player Project at the University of Southern California, Gazebo has evolved into a standalone simulator, noted for its extensive features [37]. Its integration with the Robot Operating System (ROS) [38] has facilitated the widespread adoption of diverse robotic applications, from educational to industrial automation. Gazebo's ability to simulate a variety of sensors and actuators in indoor and outdoor environments has solidified its role as a critical tool in robotics research and development [39].

Webots, a widely adopted open-source robotics simulator, provides a complete development environment to model, program, and simulate robots. It is known for its realistic physics, high-quality 3D rendering, and support for various robots and sensors [32]. An often-included part of the Robot Operating System (ROS), Gazebo Sim offers a realistic rendering of dynamic environments and supports a broad range of sensor and actuator models. Its versatility and integration with ROS have made it a popular choice in academia and industry [39].

CARLA (Car Learning to Act) [40] and ZeroSim are other noteworthy open-source simulators. CARLA is specifically designed for autonomous driving research, providing open digital twins of urban layouts, vehicle dynamics, and a realistic sensor suite for testing autonomous driving solutions. ZeroSim, developed on the Unity 3D platform and powered by ROS, offers a versatile robotics simulation engine. It is user-friendly, supports swift development, and caters to various robotic simulations, including robotic arms and ground and drone-based mobile robots [41].

On the commercial/closed-source side, Isaac Sim and Unity 3D [42] have significantly

impacted the field. Isaac Sim, powered by NVIDIA Omniverse [43], provides a high-fidelity, physically accurate environment for training and testing robots, leveraging advanced real-time ray tracing and even generative AI capabilities using the extension Isaac Gym [43], [44]. Unity 3D, a powerful game development engine independently adapted for robotics simulation [20], [45], offers high-quality graphics, physics simulation, and a comprehensive scripting API, making it an attractive choice for simulating various robotic systems [42].

Research by [46] compared a subset of these software packages but found no single best simulator compatible with ROS version 2. However, Ignition and Webots showed higher stability, and PyBullet and Coppeliasim were more efficient regarding resource usage. Additionally, [47] demonstrated that Unity3D, combined with ROS middleware (ROS-Unity3D), could be a viable alternative to the ROS-Gazebo simulation environment. [41] also provides a ROS interface with Unity 3D; however, there appears to be minimal information available online and no ongoing development at the time of writing.

Ultimately, the choice of simulation software depends on the specific project requirements. While traditional simulation environments like Gazebo are efficient for smaller environments and integrating with ROS, Unity3D emerges as a promising alternative for larger environments and applications requiring realistic visual data [48]. The ongoing challenge for developers is to optimise these tools for various tasks, including developing digital twins. However, dedicated robotic simulators appear to be a better option for interfacing with ROS than game engines for specific tasks.

2.2 Digital Twins in Robotics

A Digital Twin (DT) is a virtual replica of a physical entity that can be continuously updated with data from its physical counterpart [49], [50]. It is an integrated multi-physics, multiscale, probabilistic simulation of a complex product that uses the best available physical models, sensor updates, and interfaces to mirror the life of its corresponding twin [51], [52]. This concept allows the creation of a digital model that can simulate, verify, control, and predict the entire life cycle process of the physical system [52]. Riding the wave of Industry 4.0, the concept of a DT has been gaining prominence, with various definitions spotlighting different facets such as data flow or services [12], [14], [53]–[55] offers a classification based on data flow (figure 1), distinguishing between the following:

- Digital Model (DM) - A pure simulation with no real-world link.

- Digital Shadow (DS) - A 3D visual "mirror" without two-way interaction.
- Digital Twin (DT) - A two-way interface mirrors and controls real-world states.

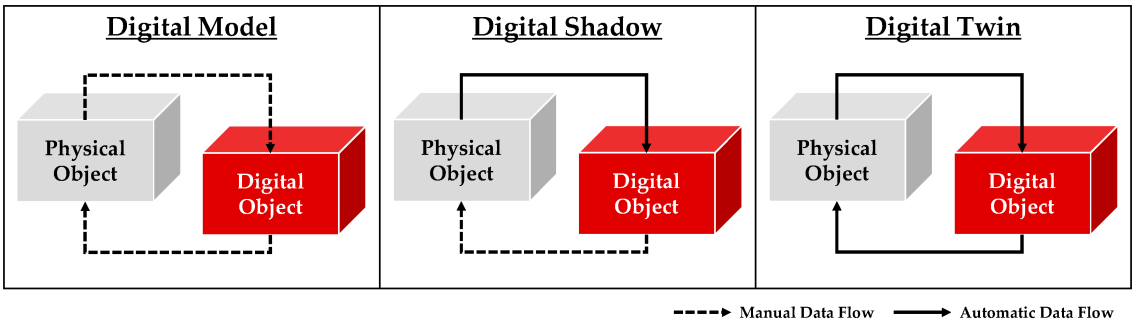


Figure 1: Diagram describing the dataflows of physical and digital objects proposed by [56]

For robotics, a DT is a virtual representation encompassing the system's physical properties, operational dynamics, onboard sensors, actuators, and complex control algorithms [57]. DTs also replicate the intended operational environment, achieved through 3D scanning [52], requiring substantial processing to portray the robot's deployment area's visual, physical, and material characteristics [57]. DTs also enable the synchronisation of the virtual and physical worlds, allowing for real-time remote monitoring [52], [58], control, and simulations for intelligent decision-making [53], [59], [60]. DTs support the design, operation, and analysis of complex systems in various domains, such as manufacturing [53], aerospace [50], healthcare [61], [62], and smart farming. Because they contain comprehensive information about the systems and processes of the twinned original system, they can offer a better understanding of cyber-physical systems during their design time and more efficient operations of these systems [12]. DTs enable predictive maintenance in operations, identifying potential failures before they occur and reducing downtime [52]. DTs can aid in training operators [63], enabling them to familiarise themselves with the system within a risk-free simulated environment, thus enhancing operational safety and efficiency [64], [65].

DTs are used extensively for simulations and data visualisations during the design and development stage and the recreation of physical boundaries and external processes that interact with robotic devices, providing accurate information on their surroundings for better mission planning [52], [60]. Furthermore, they support the development of intelligent control systems and optimise the control performance of physical systems [52]. Finally, digital twins can be used to create a reliable analysis platform for visualising and optimising data in production environments designed with smart technologies [12], [52].

Multiple publications have underscored the benefits of Digital Twins in a general sense, from enhancing health and safety [59] and fostering collaborative robot development [64] to expediting iterative development, testing [66] and even training artificial intelligence (AI) models [58], [67]. Digital Twins allow exhaustive testing and optimisation in a virtual environment, reducing the likelihood of design or operational issues and yielding considerable cost savings and shorter time-to-market [52].

2.2.1 Challenges

The effectiveness of digital twins in robotics faces hurdles such as ensuring real-time behavioural mirroring between the virtual and physical models and optimising the control performance within the digital twin framework. Additionally, the complexities of multibody physical simulation [53] and the need for instant sensor feedback complicate real-time decision-making processes, especially in rapid operational environments. Interoperability challenges, particularly in robot teleoperation and 3D mapping of unfamiliar environments [59], [60], further complicate the development and application of digital twins in robotics. Addressing these challenges is critical for successfully integrating digital twins with robotics, especially in safety-critical and real-time decision-making scenarios. Issues such as real-time data synchronisation, simulation accuracy, and adapting digital twins to complex, dynamic interfaces between robots, humans, and environments present notable difficulties. Moreover, implementing digital twins requires a robust communications infrastructure for effective data collection and analysis, highlighting the need for continuous research and development to overcome these obstacles and fully leverage the benefits of digital twins in enhancing robotic systems.

2.2.2 Reality Capture

Reality Capture is a technique for digitising physical spaces and objects into accurate 3D models, widely used in construction for Building Information Modelling (BIM) and 3D asset creation. It offers a quicker alternative to traditional 3D modelling or CAD, particularly in generating digital representations.

Three-dimensional scanning techniques have evolved significantly recently, opening new possibilities for object modelling and environment mapping. Notably, 3D LiDAR Scanning and photogrammetry have emerged as core technologies in the field for decades. In recent years, Neural Radiance Fields (NeRF) [65] have expanded the field to incorporate recent advances in neural networks. Each technique has unique characteristics, offering different advantages and

drawbacks regarding precision, accuracy, and application potential.

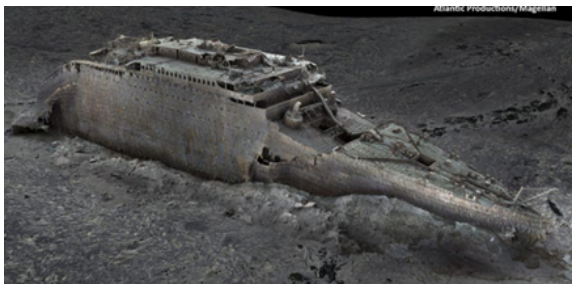
3D scanning predominantly focuses on two primary targets: environments and objects. Regarding environments, 3D scanning captures the entirety of a space or a specific locale, such as a room, a building, or even an outdoor setting; this could include everything from the layout and size of the area to the minute details of textures and colours. The captured data provides a rich, detailed, and precise digital representation of the environment that can be used for various purposes, including but not limited to architectural planning [68]–[70] virtual reality experiences, [71], or simulations [65]. On the other hand, when the target of the scan is an object, the aim is to capture the object's exact shape, size, and sometimes its surface texture; this could be anything from a small household item, a large mechanical part, even recreating a damaged vehicle [72]. The resultant 3D digital model serves many purposes, such as product development, quality control, or digital archiving [73]. In both cases, 3D scanning aids in creating a digital twin, offering an incredibly accurate and detailed digital counterpart of the real-world target.

2.2.2.1 3D Lidar Mapping

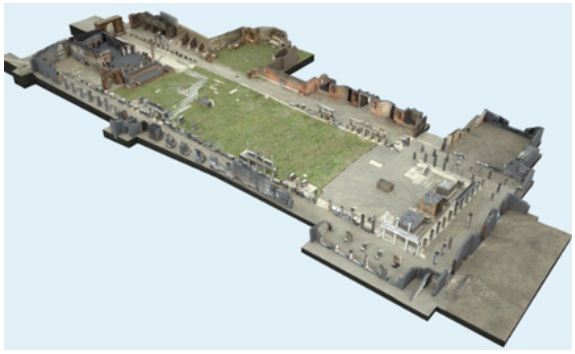
3D Lidar mapping, a cutting-edge technique, employs pulsed laser light to measure distances to a target, crafting precise three-dimensional models of objects or environments. This technology operates by dispatching a light pulse and timing the interval for the light to bounce back after striking an object, thereby calculating the distance to that object [74], [75]. Lidar can swiftly and accurately map extensive areas [76], rendering it instrumental in fields such as surveying [33], forestry [77], and autonomous vehicle navigation. The accuracy of 3D Lidar mapping is highly contingent on the quality of the Lidar sensor and the conditions during data collection, with factors such as atmospheric conditions, target reflectivity, and the angle at which the laser strikes the target potentially affecting the data accuracy. Despite these potential sources of error, Lidar is universally recognised as one of the most precise and dependable methods of 3D scanning.

LiDAR scanning has been widely adopted across diverse industries, including transport, infrastructure, archaeology, environmental monitoring, structural engineering, urban planning, and agriculture, thanks to its ability to generate high-resolution, accurate 3D models even in challenging environments and lighting conditions [33], [34], [71], [76], [78]–[82]. This versatile technology can be utilised from aerial, terrestrial, or mobile platforms, each offering unique advantages and limitations depending on the application [33], [76], [79], [83].

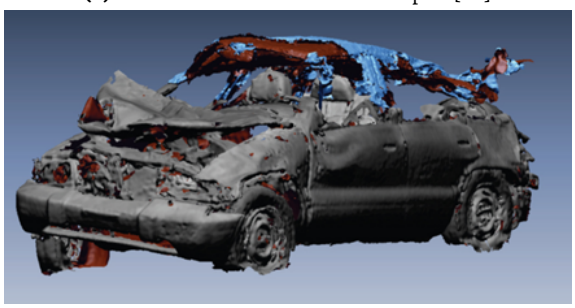
Despite its powerful capabilities for 3D environment scanning, LiDAR scanning has certain limitations, such as higher equipment costs and the need for complex processing algorithms. The



(a) 3D scan of the Titanic on the seafloor [84]



(b) 3D render of the forum in Pompeii [40]



(c) 3D scanning used for accident reconstruction and analysis [72]

Figure 2: 3D Scanning examples for a variety of purposes.

mesh quality primarily depends on the point cloud's density and accuracy and the robustness of the employed mesh reconstruction algorithm [85]–[87].

While LiDAR scanning can yield accurate, high-resolution 3D models in varied environments, it often demands specialised equipment and entails higher costs. Conversely, photogrammetry provides a more cost-effective solution, albeit potentially struggling with reflective or transparent surfaces and fluctuating lighting conditions [88].

In conclusion, LiDAR scanning, with its high-resolution point cloud generation capacity, is a versatile and widely adopted technique in numerous industries. Despite its higher costs than photogrammetry, LiDAR scanning provides distinct benefits such as penetrating vegetation, performing well under varied lighting conditions, and capturing 3D data in real-time. The continuous advancements in LiDAR technology and its integration with other sensing modalities further enhance the precision and depth of 3D environment scanning. With the emergence of affordable LiDAR scanners, such as those found in the latest iPhones, 3D LiDAR

scanning is becoming increasingly accessible even to smaller operations [89]. Choosing between LiDAR scanning and photogrammetry will ultimately depend on each application's requirements and constraints.

2.2.2.2 Photogrammetry

Photogrammetry is a process that transforms multiple two-dimensional images, taken from various perspectives, into three-dimensional models. It involves pinpointing a consistent point across multiple photographs and employing triangulation to ascertain the location of that point in 3D space. Because of its cost-effectiveness and accessibility, photogrammetry is widely used in geology, architecture, and video game design. Its accuracy relies on several factors, including image resolution, the angle and distance of image capture, and the software used for image processing. Under optimal conditions, photogrammetry can be highly accurate [90], though environmental factors, such as lighting, can affect the results [91]. Despite its dependence on photography, which limits its application in specific environments, the accuracy and versatility of photogrammetry make it a valuable tool.

The photogrammetry process can be broken down into the following steps:

1. **Image Acquisition**: Overlapping photographs of the object or scene are taken from various angles.
2. **Feature Detection**: Identifiable features, such as corners or edges, are pinpointed in each image.
3. **Feature Matching**: Correspondence between features in different images is established.
4. **Camera Pose** Estimation: The position and orientation of the cameras for each image are calculated.
5. **Triangulation:** A 3D point cloud is generated by estimating the 3D coordinates of each matched feature.
6. **Mesh Reconstruction**: A 3D mesh model is created by connecting the points in the point cloud.
7. **Texture Mapping**: Image textures are applied to the mesh to simulate a realistic appearance.

Photogrammetry is a dominant technique for 3D environment scanning, primarily due to its ability to generate detailed and accurate 3D models using relatively low-cost equipment and

software [90], [91]. It can be performed using aerial [92], [93], terrestrial [70], or close-range platforms [69], with each having its unique advantages and limitations depending on the application [90]. The advent of Unmanned Aerial Vehicles (UAVs) has further expanded photogrammetry's potential, offering a cost-effective and flexible method of acquiring high-resolution imagery for 3D scanning [92], [94].

Figure 3: An old log building was scanned using photogrammetry with Reality Capture and imported to the Unreal Engine 5 game engine [95]

The photogrammetry process consists of capturing overlapping images, extracting features from these images, and establishing correspondences by matching these features. The 3D scene is then reconstructed using mathematical techniques such as triangulation and bundle adjustment [96]. Despite its power, photogrammetry faces challenges such as dealing with complex scenes, occlusions, and varying lighting conditions [80], [90].

For mesh generation, a dense point cloud is created post-feature extraction and image matching, which serves as the basis for mesh reconstruction. Techniques like Delaunay triangulation [97] or Poisson surface reconstruction [86] can be utilised. The resulting mesh quality depends on factors such as point cloud density, accuracy, and the robustness of the mesh reconstruction algorithm [98].

In conclusion, photogrammetry is a versatile 3D scanning technique used across various industries, offering an accessible and cost-effective method of acquiring high-resolution

imagery. Despite its challenges, it remains a powerful tool for tasks requiring accurate, detailed 3D models.

2.2.2.3 Neural Radiance Fields (NeRF)

Neural Radiance Fields (NeRF) have emerged as a groundbreaking development in 3D scanning, utilising artificial intelligence to transform 2D images into detailed 3D models. This innovative technique, leveraging deep neural networks, interprets 2D photographs to infer three-dimensional structures, potentially diminishing the dependency on manual modelling processes. The effectiveness of NeRF hinges on the quality of the input images and the neural network's processing abilities, with the potential for enhanced precision and accuracy as the model learns from more data. Despite being a relatively novel approach, NeRF has rapidly become pivotal in 3D environment scanning, illustrating remarkable capabilities in rendering complex scenes with high fidelity from sparse datasets, thus attracting considerable interest within computer vision and graphic design spheres [99], [100].

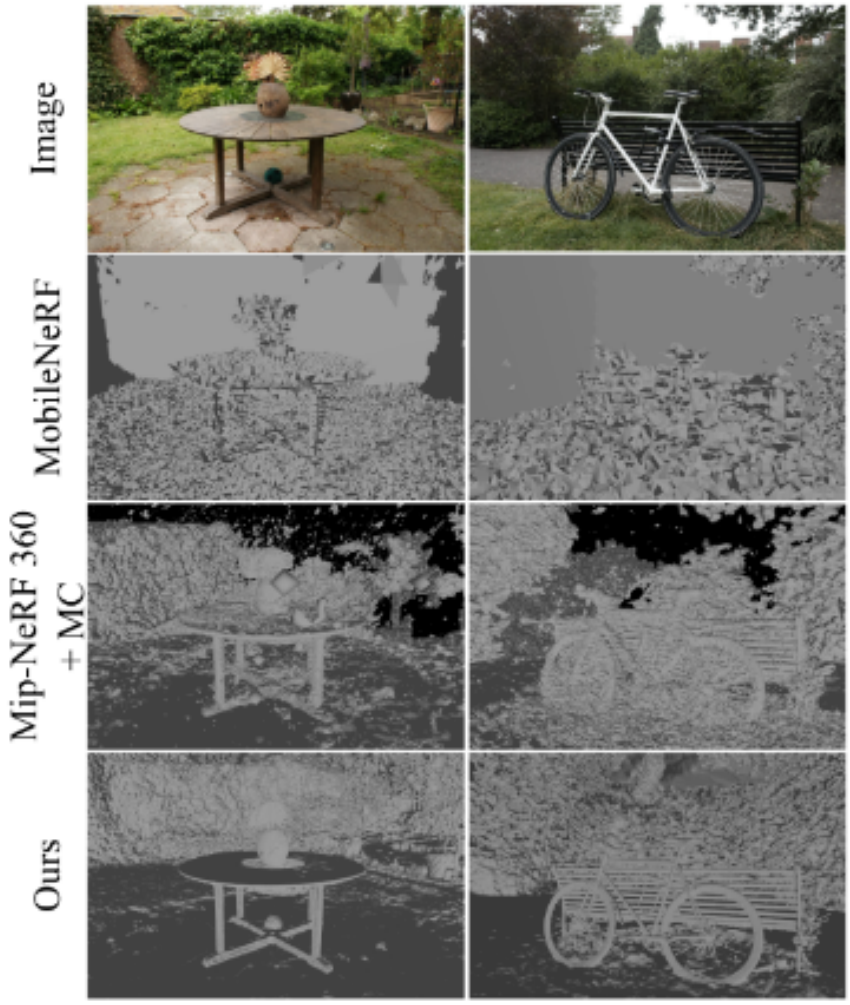


Figure 4: Improved details mesh fidelity achieved by [101]

Subsequent research has significantly extended NeRF's utility, addressing its initial

constraints by introducing solutions for dynamic scene rendering, scalability for expansive environments, and real-time rendering advancements [100], [102]–[106]. D-NeRF [102] and S-NeRF [107] have refined NeRF's framework to effectively manage time-varying scenes and sparsely imaged environments, respectively. Meanwhile, Grid-NeRF [104] corrects perspective distortions in reconstructions, and NeRFMeshing [101] dramatically enhances mesh generation through intermediate mesh representation and optimisation techniques. These advancements have widened NeRF's applicability across virtual and augmented reality, robotics, and autonomous navigation, though challenges persist in areas such as data requirement, computational demand, and handling occlusions [108]. Addressing mesh generation issues, the Signed Surface Approximation Network (SSAN) by [101] signifies a substantial improvement in 3D volumetric representation and mesh extraction.

These developments underscore NeRF's evolving role in 3D environment scanning and modelling, showcasing a continuous trajectory toward overcoming limitations and expanding its capabilities across various domains.

2.3 Examples of Use

Implementing simulations in robotic development is a multifaceted process, encompassing standard practices and methodologies and varying significantly across different sectors.

2.3.1 Manufacturing Robotics

In manufacturing, especially in the automotive sector, simulations are used to design and optimise robotic assembly lines. The case of robotic arms in automotive manufacturing exemplifies this, where simulations are crucial for programming precise tasks like welding and painting, leading to more efficient production processes [8]. BMW collaborated with Nvidia in mid-2023 to create a Digital Twin of a full-size car factory [109] to provide a simulation platform for developing and optimising their systems.

The BMW Group has undertaken a pioneering project by incorporating the NVIDIA Omniverse platform into its global production network. This move is particularly emphasised in their new electric vehicle plant in Debrecen, Hungary. The integration represents a shift towards a digital-first approach in automotive manufacturing, using Omniverse for extensive virtual planning and optimisation of production processes. This innovation leads to increased efficiency and promotes sustainable manufacturing practices.

The Omniverse platform enabled BMW to simulate factory operations in real-time, significantly improving planning accuracy and operational flexibility. This digital twin

technology facilitates seamless reconfigurations of factory layouts for new vehicle models without necessitating physical presence, thus optimising worker safety and production line ergonomics. Moreover, BMW's integration of complementary NVIDIA technologies, such as the Isaac robotics platform and the EGX edge computing platform, enhances the factory's material flow and logistics, demonstrating a sophisticated application of intelligent robotics in manufacturing.

The collective use of these technologies has culminated in a more streamlined and flexible planning process, reducing time requirements while increasing precision. The objective is to produce highly customised vehicles efficiently across BMW's global factories, reinforcing the brand's commitment to innovation and customer satisfaction.

2.3.2 Research

Disney's use of BD droids in Star Wars: Galaxy's Edge [110] exemplifies advanced digital twins, simulations, and artificial intelligence applications. These autonomous droids enhance the guest experience by interacting lifelike within the park's complex environment. This development draws from the innovative research conducted by Disney Research, aimed at reducing the gap between virtual simulations and real-world applications to improve operational effectiveness and interaction capabilities in dynamic public spaces.

In their research paper [111], Disney Research outlines the use of transformer-inspired architectures to augment simulation representations of robots. This approach addresses the challenge of sim-to-real gaps, such as those caused by friction or other unpredictable behaviours, by predicting the actual state of robotic components. By focusing on the augmentation of individual building blocks rather than the entire simulation, the method enhances modularity, generalizability, and robustness, contributing significantly to the field.

2.3.3 Logistics

Amazon Robotics has introduced Proteus, its first fully autonomous warehouse robot, to its extensive fleet [112]. Proteus uses various sensors and advanced technologies to navigate safely within facilities. Utilising NVIDIA's Isaac Sim on Omniverse, the team has created realistic simulations to expedite the deployment of Proteus. This workflow has significantly improved the performance of its autonomous systems, notably increasing its marker detection success rate. Moreover, Amazon Robotics is leveraging Omniverse to develop full-fidelity digital twins of warehouses, enhancing operational efficiency and speeding up the development process, transitioning from months to days.

2.3.4 Agricultural Robotics

Agricultural robotics use simulations to design robots capable of navigating diverse terrains and weather conditions. An example is the development of autonomous tractors, where simulations test navigation systems and obstacle detection algorithms, ensuring effective operation in agricultural settings. [113] reviews several professional simulators and custom-built virtual environments for agricultural robotic applications, such as weed control, field scouting, and harvesting. They also compare three selected simulators' key features and performance efficiency: V-REP, Gazebo, and ARGoS. They demonstrate a simulation case study using V-REP to create a virtual citrus orchard and a robotic platform for scouting and data collection. They suggest that an open-source software platform for agricultural robotics will significantly accelerate effective collaborations between different research groups.

2.3.5 Healthcare Robotics

The field of healthcare robotics, particularly in surgical applications, has increasingly leveraged advanced simulation technologies. These simulations serve as crucial platforms for training surgeons and planning complex procedures, enhancing the precision and safety of robotic-assisted surgeries [13].

Simulators like the da Vinci Skills Simulator and the RobotiX Mentor play a pivotal role in medical training. They offer highly realistic virtual environments where surgeons can practice various surgical procedures. This environment provides invaluable immediate feedback, facilitating skill development in a risk-free setting [13]. The importance of such simulators is further underscored by studies like those of [62] and [114], which validate the effectiveness of virtual reality and augmented reality technologies in enhancing the fidelity and educational value of these simulations.

Furthermore, the Digital Twin concept is being progressively incorporated into robotic surgery. This integration involves creating detailed virtual replicas of the surgical environment, including patient-specific anatomy, which surgeons can use for pre-operative planning and rehearsal [12]. This innovation allows for precise tailoring of surgical procedures to individual patient anatomy, offering the potential for more personalised and optimised surgical interventions.

The integration of simulation technologies in healthcare robotics, particularly in surgical training and planning, signifies a significant leap forward. It enriches surgical training and paves the way for more personalised and effective surgical interventions, ultimately leading to improved patient outcomes [12], [14], [62], [114].

2.3.6 Human-robot collaboration

Human-robot collaboration (HRC) represents a rapidly evolving field in robotics, increasingly pivotal in diverse sectors such as manufacturing, healthcare, and service industries [115]. Simulations in HRC are essential for designing systems where humans and robots work side by side safely and efficiently. These simulations intricately model human behaviour, ergonomics, and interactive dynamics, ensuring that robots can adapt to human movements and operate nearby without risk. Toyota, a forerunner in deploying collaborative robots, extensively uses simulations to refine the safety and functionality of HRC systems in their manufacturing plants [116]. Recent studies, such as those by [43], [109], have also highlighted the importance of simulations in understanding and optimising HRC scenarios. These studies emphasise the role of simulations in predicting potential safety hazards and ergonomics issues, thereby facilitating the creation of more intuitive and responsive robotic systems. Integrating technology and simulation-based testing is critical in paving the way for more effective and human-centric robotic systems.

2.3.7 Advanced Simulation Applications

Advanced simulation techniques in robotics have evolved to incorporate real-time operations, integration with artificial intelligence (AI), and the development of novel tools, all of which are shaping future trends in robotic simulations.

2.3.8 Future Trends in Robotic Simulations

The evolution of robotic simulations is characterised by a shift towards more sophisticated, integrated systems. A key emerging trend is the development of continuously evolving digital twins. These dynamic digital replicas closely mirror their physical counterparts, enabling real-time analysis and optimisation [12]. Advanced AI techniques, like reinforcement learning and generative models, are being integrated to create simulations that are more predictive and autonomous [43]. This synergy of AI and simulation technologies is paving the way for a new era in robotics, where systems are responsive and self-evolving. As we delve deeper into these technological advancements, the role of digital twins in robotics comes to the forefront, signifying a pivotal shift in how robotic systems are developed and maintained. The subsequent sections explore this concept in greater detail.

2.3.9 Integration with Artificial Intelligence

AI's integration into robotic simulations represents a significant leap forward. AI algorithms, especially those in machine learning and deep learning, are now being used to simulate complex decision-making processes and adaptive behaviours within robots [67]. These algorithms allow for simulations that can learn from their environment over time [117], offering insights into potential challenges and robotic responses in diverse scenarios. AI-driven simulations enhance the predictive capabilities of these tools, accelerating the development of more intelligent and autonomous robotic systems [15]. Cloud-based platforms allow simulations to run on powerful remote servers, offering scalability and accessibility for large-scale projects, even for lower-scale companies [116].

2.4 Conclusions

In conclusion, this literature review has highlighted the transformative potential of digital twin technology when integrated with robotic simulations. This synergy offers precise, real-time reflections of both robots and their operational environments, significantly enhancing the accuracy of simulations and providing invaluable insights for system enhancements. Despite the clear advantages, the review underscores a notable research gap in applying 3D-scanned digital twins within robotic simulations—an area ripe with potential but currently underutilised. The necessity for deeper investigation into effective methods of merging 3D-scanned data with robotic simulations is evident. Moreover, the quality of these 3D scans emerges as a pivotal element, influencing the overall efficacy of digital twins in such simulations. Addressing these challenges and harnessing the combined power of digital twins, 3D scanning, and robotics simulations could usher in revolutionary developments in robotics.

Additionally, the review points out the absence of studies quantifying the benefits of simulation-based approaches over traditional field testing. Consequently, we propose several research questions to explore the impact of digital twins and simulations on the development velocity, efficiency, and overall quality of robotics, alongside the challenges and optimal software tools for their integration. By delving into these inquiries, future research could significantly advance the understanding and application of these technologies. This review sets the stage for a new direction in robotics research that promises substantial advancements and underscores the need for further scholarly exploration.

CHAPTER 3

Methods

This chapter outlines the general tools and processes for creating a simulation framework and software pipeline. The following sections will describe the proposed differences in the robot development cycle, analyse these techniques, and evaluate the relevant software processes.

3.1 Robot Development Cycle

The Traditional Robot Development Cycle is a structured approach that guides the transformation of a robotic concept into a functional system. Illustrated in figure 5, this cycle starts with the Scoping phase, where the project's goals and constraints are defined, setting the stage for the entire development process.

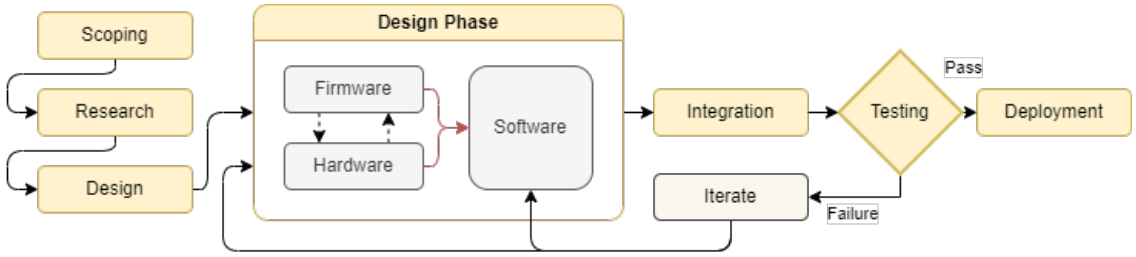


Figure 5: Traditional Robot Development Cycle

The subsequent phases of the traditional robot development cycle start with the Research Phase, where a thorough review of existing technologies and cutting-edge advancements in robotics informs critical design and development decisions, ensuring the project is built on a solid foundation of current knowledge and best practices. The Design Phase follows, where specific decisions about the robot's capabilities, architecture, and components are established. During this phase, the project is divided into concurrent hardware and firmware development tracks, often creating a bottleneck, as software development must wait for the completion of hardware and firmware, thereby delaying the software team's involvement and slowing overall

progress. Once the hardware and firmware are completed, software development begins, and the project moves into the Integration Phase, where all components are integrated into a cohesive system and tested to ensure they meet specified performance criteria. Successful testing leads to deployment, while failures trigger a return to the Iterate loop for necessary adjustments, highlighting the challenges of sequential dependencies and the need for methodologies that enable parallel development to enhance project efficiency.

The traditional development cycle illustrates the challenges of sequential dependencies in robotics development. Delays in software development, pending the readiness of hardware and firmware, highlight the need for methodologies that enable parallel development to enhance efficiency and foster innovation.

3.1.1 Digital Twin Integration

This section examines the integration of Digital Twins in the development cycle, highlighting their role in facilitating parallel development and mitigating traditional bottlenecks. DTs allow for the simultaneous advancement of software and hardware development tracks, a departure from the linear progression seen in conventional cycles.

These advanced simulations can significantly reduce development time by enabling simultaneous progress in software programming, system integration, and performance testing, all while the physical components are still under development. Additionally, this approach has the potential to reduce overall project costs. Expenses typically associated with repeated physical prototyping and testing can be minimised, as many iterations can now occur in a virtual environment, which requires far fewer resources. Moreover, the environmental impact of development is also lessened as Digital Twins reduce the need for physical materials and the waste generated from iterative physical prototypes.

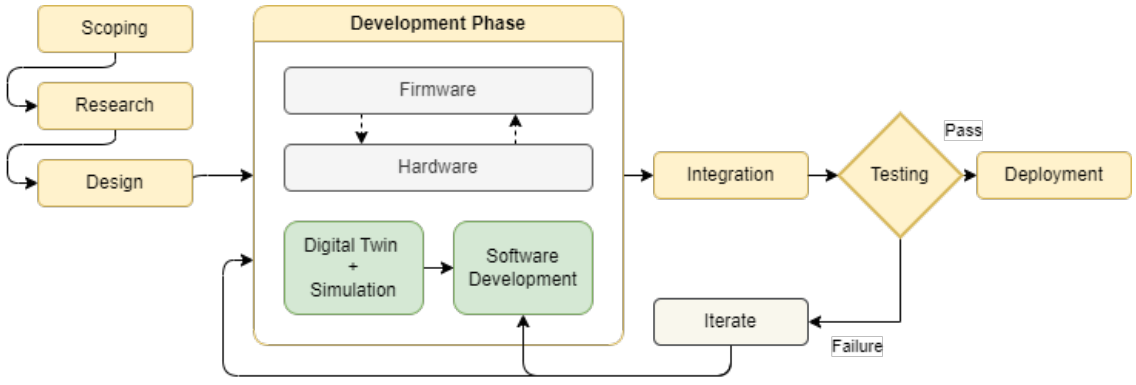


Figure 6: Parallel Robot Development Cycle with Digital Twins

By setting this foundation, the subsequent sections and case studies will explore specific

instances and methodologies where Digital Twins have been effectively employed, demonstrating their practical applications and the tangible benefits they bring to robotics development. This discussion aims to provide a comprehensive understanding of how Digital Twins enhance the efficiency of development cycles and contribute to more sustainable engineering practices.

The utilisation of Digital Twins is explored further in subsequent case studies, illustrating their practical benefits in streamlining development processes and highlighting their potential to transform traditional robotics development methodologies.

3.2 Software and Tools

This section describes the technical software stack used for the project through the case studies, highlighting appropriate tools for different purposes in reality capture, modelling, and simulation and their associated workflows.

3.2.1 Reality Capture and Modelling

- **Polycam:** A cloud-processed photogrammetry app used for quick and effective 3D Scanning.
- **CAD Assistant:** Used for converting scanned data into usable CAD formats.
- **Solidworks 3D and Blender 3D:** These tools are used for detailed CAD and mesh modelling. Blender is primarily used to process and clean mesh files generated by reality capture techniques.

3.2.2 Digital Twins and Simulation

Webots by Cyberbotics and Nvidia's Isaac Sim were the principal simulation systems selected due to their distinct advantages for different simulation goals.

3.2.2.1 Nvidia Omniverse and Isaac Sim

Isaac Sim is a simulation platform designed to enhance the realism and efficiency of robotic simulations. It provides exceptional photorealism through its RTX raytracing and advanced physics simulation using Nvidia's PhysX engine. The platform's strong integration with ROS enhances its capability to develop and test more complex robotic navigation systems, particularly for indoor robotics and Autonomous Guided Vehicles (AGVs). Omniverse is a platform that provides a collaborative virtual environment where developers can build and operate simulated realities, leveraging its powerful rendering capabilities, real-time

photorealism, and physics accuracy to create complex, interactive scenes. It is particularly beneficial for creating Digital Twins that can interact dynamically with their virtual environments. Isaac Sim, tailored specifically for robotics, offers high-fidelity simulation tools that integrate seamlessly with ROS, facilitating the development and testing of robotic systems within realistic scenarios. This platform excels in simulating intelligent machine behaviours and sensor interactions, making it ideal for testing navigation, manipulation, and perception algorithms before physical deployment.

3.2.2.2 Cyberbotics Webots

Webots is particularly valued for its user-friendly interface, which is established, portable, and cross-platform, making it accessible to those new to robotic simulation. It offers continuously improving support for ROS 1 and 2, facilitating seamless integration with robotic systems, which is crucial for quickly developing primary control and navigation systems. Webots is suited for simulations requiring various robotic features like tracked movement, GPS navigation, and pen devices without the high computing demands associated with more photorealistic environments. These characteristics made Webots ideal for scenarios where visual fidelity was less critical than versatility and ease of integration.

3.2.3 Programming Languages

- **C / C++:** In advanced workflows, C is utilised predominantly for firmware programming and low-level control systems, where maximum efficiency and direct hardware interaction are essential. Its capacity for low-overhead operations makes it ideal for applications requiring stringent performance criteria, ensuring that the system can handle real-time processing demands without compromise. C++ is employed extensively in developing complex software modules, leveraging its object-oriented nature and rich standard library. This language is particularly suited for creating sophisticated systems that require robust structure and scalable architecture, allowing developers to manage complex functionalities more effectively.

- **Python:** Python is widely used for scripting, data analysis, and interfacing within simulation environments like ROS and Webots. Popular for its ease of use and vast library ecosystem, Python is instrumental in testing and integration tasks, where its flexibility and powerful third-party packages streamline complex analytical and procedural operations.

3.3 Digital Twins

This section will elaborate on creating and integrating Digital Twins within the development cycle, detailing how these virtual replicas are built, maintained, and utilised to mirror real-world systems accurately.

A DT for robot simulation comprises several primary elements designed to accurately replicate the physical robot's functionalities and behaviours in a virtual environment. At its core, the digital twin includes a detailed 3D model of the robot, accurately reflecting its physical counterpart in terms of geometry, dimensions, and aesthetics. This model is integrated with kinematic and dynamic simulations that mimic the robot's actual movements and interactions with the environment, ensuring that the twin can simulate real-world physics to a level fit for purpose. Sensor emulation involves virtualised sensors that behave like their real-world equivalents, providing necessary data inputs such as proximity, vision, and tactile responses that the robot would encounter in operation.

The control system of the twin mirrors the actual control algorithms used by the physical robot, enabling developers to test changes in control strategies under various simulated conditions without the risk and expense associated with physical trials. Additionally, the digital twin includes a communication interface that syncs with the robot's actual control hardware and software platforms (as shown in Figure 7), such as ROS (Robot Operating System), to ensure seamless data exchange and operation harmonisation between the virtual and real environments. Together, these elements allow for comprehensive testing and development, enhancing the robot's design and functionality before final deployment.

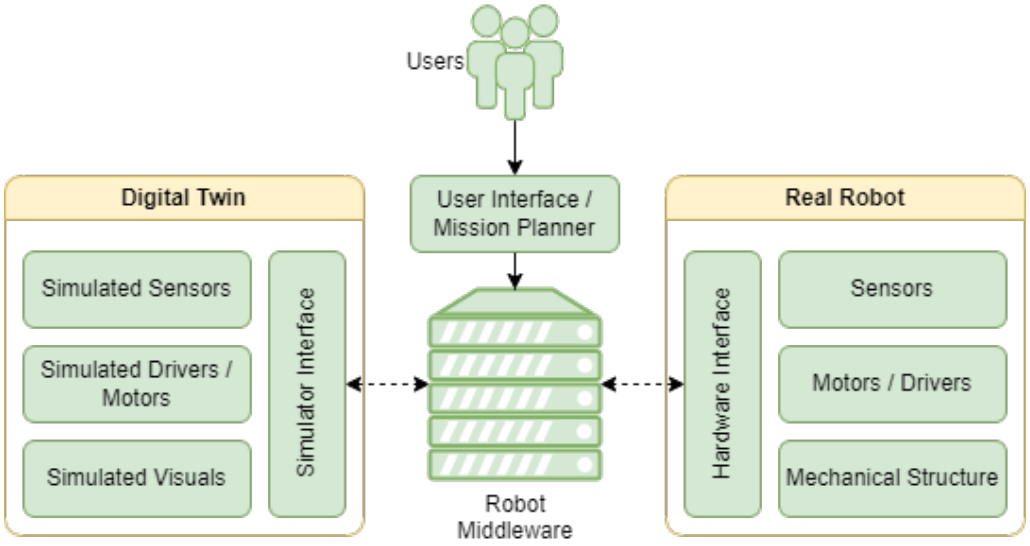


Figure 7: A diagram describing the relationship between the robot simulation, the physical robot, and the operating system.

3.4 General Simulation Workflow

Setting up a Digital Twin for simulation involves systematic steps to ensure the twin accurately reflects the physical counterpart and meets the end user's specific needs. This section describes the general workflow for developing a Digital Twin, from gathering initial requirements to completing the final integration and delivery.

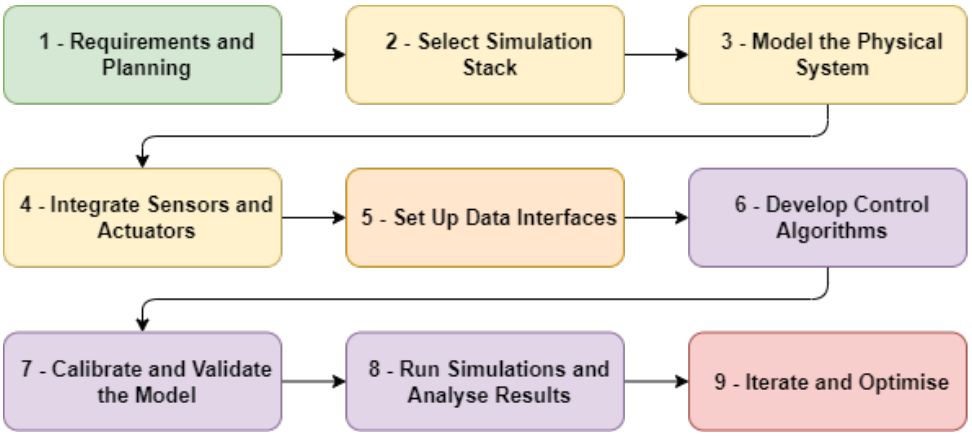


Figure 8: Generalised workflow for the setup and configuration of a Digital Twin and simulation for robotics.

3.4.1 Requirements and Planning

The initial step in the workflow involves scoping the requirements to understand what the end user needs from the simulation. This phase determines the essential features and functionalities the Digital Twin must replicate. It involves detailed discussions with the end user to capture all critical aspects of the physical system's operations, environmental interactions, and expected simulation outcomes, ensuring that the simulation will be tailored to provide actionable insights and practical value.

3.4.2 Select Simulation Stack

The next step is selecting the appropriate software stack and reality capture techniques based on the scoped requirements. This decision critically influences how the robot's physical attributes are digitised and integrated into a simulation platform that best fits the project's needs and the complexity of the robotic system. Below are two comparative tables highlighting the differences between Webots and Isaac Sim as simulation platforms and the choice between reality capture versus pre-existing CAD models.

The choice between these platforms and techniques significantly influences the project's technical alignment, efficiency, and effectiveness in meeting development goals. Webots is often chosen for projects that value quick development turnarounds and are less dependent on

Aspect	Webots	Isaac Sim
Ease of Use	High (User-friendly interface)	Moderate (Requires familiarity with advanced simulation tools)
Resource Consumption	Low (Efficient on standard hardware)	High (Requires advanced GPU and computational resources)
Suitability	Ideal for projects prioritising accessibility and rapid development	Preferred for projects requiring high realism and detailed physics
Development Speed	Faster development cycles due to ease of use and efficiency	Slower due to complex setup and higher resource demands
Realism and Fidelity	Sufficient for many applications, not as high as Isaac Sim	High-fidelity and realistic simulations

Table 3.1: Comparison of Simulation Platforms (Webots vs Isaac Sim)

ultra-high-fidelity simulations. In contrast, Isaac Sim is more suitable for projects where precision and high-level realism are critical, provided there is access to the requisite technological infrastructure.

3.4.3 Physical Geometry

The core of the workflow is setting up the DT. This process includes creating or configuring the robot's model in the chosen simulation software, such as defining the URDF (Unified Robot Description Format), Webots PROTO, or USD (Universal Scene Description) files. Simulated sensors and hardware interfaces are established to match the end-user's operational environment and requirements. This step ensures that all robot functionalities are mimicked accurately, from movement and interaction to sensor feedback and data processing.

Reality capture is useful when no CAD models exist, enabling rapid acquisition of detailed physical data. However, utilising existing CAD models is more efficient when accurate and up to date, reducing the need for extensive manual adjustments and ensuring the digital representation closely matches the physical entity.

3.4.3.1 Modelling vs Scanning

- **CAD Modelling:** Utilise Computer-Aided Design (CAD) tools to meticulously construct detailed 3D models of the physical system by using geometric details, materials, and other visual aspects to ensure that the virtual model mirrors the physical system.

- **3D Scanning:** Alternatively, 3D scanning and reality capture technologies can be employed to digitise the physical system directly. This approach uses advanced scanning devices to capture the exact shape and dimensions of the object, which can be more time-efficient

Aspect	Reality Capture	Using CAD Models
Speed of Deployment	Faster when no existing models are available	Faster if models are accurate and up to date
Fidelity and Accuracy	Ensures high fidelity by capturing exact physical details	Ensures the digital twin closely mirrors the physical robot
Suitability	Useful in dynamic environments needing precise adaptation	Best for scenarios where high fidelity is crucial
Resource Efficiency	Resource-intensive depending on the environment and details needed	More resource-efficient if existing models are readily usable

Table 3.2: Comparison of Digitisation Techniques (Reality Capture vs Using CAD Models)

and may provide higher accuracy for complex or intricate objects. The scanned data is then processed to create a 3D model replicating the physical system's geometry and appearance.

3.4.3.2 Appearance

Apply textures and material properties that accurately reflect the actual object, such as colour surface textures, and material responses like reflectivity and roughness, which are crucial for visual and functional simulations.

3.4.4 Mechanical Properties and Kinematics

Define the mechanical properties of the model, such as mass, inertia, and stiffness. These properties dictate how the model reacts to physical forces and constraints, typically involving essential characteristics such as mass and suspension tension for more straightforward requirements. More complex systems may require more depth, accounting for friction and dynamic weight shifting. Integrate the kinematic chains and dynamics equations that govern the system's movement, including articulations, constraints, and responses to external forces, ensuring that the model's movements are realistic and based on accurate physical laws.

3.4.5 Sensors and Actuators

1. **Identify Sensors**: Determine the sensors (e.g., temperature sensors, accelerometers) and actuators (e.g., motors, hydraulic actuators) essential for the system's operations.

2. **Specify Details**: Specify each component's attributes, such as range, resolution, and response time, to match their real-world counterparts.

3. **Integrate Sensors**: Implement virtual sensors in the simulation that generate data reflecting the environmental conditions. Configure these models to emulate real sensor

behaviour, including potential inaccuracies and noise.

4. **Simulate Actuators**: Simulate actuators such as motors to respond accurately to control signals, accounting for factors like load and environmental conditions affecting performance.
5. **Data Handling and Feedback**: Establish systems for capturing and processing data at rates and accuracies found in the actual system.
6. **Feedback Loops**: Create feedback loops that allow sensor data to influence actuators dynamically, ensuring realistic system responses such as encoders and position sensors.
7. **Calibrate**: Adjust simulation parameters to calibrate sensors and actuators, ensuring outputs align with expected real-world data.
8. **Test**: Conduct tests under various simulated conditions to validate the performance and integration of sensors and actuators.

3.4.6 Control Algorithms and Data Interfaces

Ensure all sensors and actuators interface seamlessly with the twin's control algorithms, using appropriate communication protocols and data formats.

3.4.6.1 Real-time Interaction

Implement real-time interaction capabilities to handle immediate responses necessary for dynamic systems.

3.4.6.2 Identify Data Sources and Destinations

Determine all data sources, such as sensors and user inputs, and where this data needs to be sent, such as to actuators, storage systems, or external analysis tools.

3.4.6.3 Communication Protocols

Choose communication protocols that meet the needs of the simulation in terms of latency, bandwidth, and reliability. Common choices in robotic simulations include MQTT for lightweight messaging and DDS for complex systems with real-time requirements.

3.4.6.4 Configure Data Formats

Standardise data formats across all simulation elements to ensure compatibility and ease of processing. Common data formats include JSON for easy use in direct data transfer and bag files for ROS systems.

3.4.7 ROS Integration

3.4.7.1 Topics and Services

Utilise ROS to manage data flows, mainly when interfacing with real-world systems or other simulations. Set up ROS topics for asynchronous data publishing/subscribing and ROS services for synchronous request/response interactions.

3.4.7.2 Custom ROS Messages

Design custom ROS messages when standard messages do not suffice, involving defining the structure and type of data each message will carry, ensuring all necessary data is transmitted effectively.

3.4.7.3 Error Handling

Implement robust protocols to manage data failures or corruption issues.

3.4.8 Testing and Validation

3.4.8.1 Interface Testing

Regularly test data interfaces to ensure they function correctly, including stress testing under high data loads and penetration testing for security vulnerabilities.

3.4.8.2 Validation with Real-World Data

Validate the interfaces by comparing simulation data flows with real-world data to ensure accuracy and reliability.

3.4.9 Controllers

The control algorithms within the digital twin simulations serve dual purposes: they govern the operational behaviour of the twin and provide a dynamic platform for end-user engineers to develop and refine their control strategies. Critical components like the differential drive controller are integral to managing the simulated actuator responses, ensuring that the digital twin mimics realistic movements and interactions. These embedded algorithms, including stabilisation and safety compliance systems, maintain the twin's operation within realistic parameters.

Additionally, the simulation environment is designed to facilitate external algorithm development. It offers robust APIs and real-time data handling capabilities, allowing engineers to implement and test their algorithms effectively. The environment is equipped with tools for performance tracking and automated testing, supporting iterative development and

optimisation. Engineers can utilise these features to evaluate their control strategies under realistic conditions, refine their approaches based on immediate feedback, and ensure compatibility and performance before real-world application.

3.4.10 Model Calibration and Validation

Model calibration and validation are critical to ensuring the DT's accuracy and reliability. Calibration involves adjusting the simulation parameters to align closely with empirical data, ensuring that the model's physical and operational characteristics reflect real-world conditions. Validation processes follow, where the calibrated model is tested against known benchmarks or real system outputs to verify accuracy. This step is essential for certifying that the digital twin provides a faithful and effective representation of the actual system, thereby supporting its use for further experimentation and decision-making.

3.4.11 Run Simulations

Running simulations involves executing the digital twin model under various controlled scenarios to observe its behaviour and interactions within the simulated environment. This phase is crucial for understanding the system's dynamics, predicting potential issues, and assessing the model's response to different conditions. By systematically exploring these simulations, insights can be gained into the performance and operational limits of the system, which are invaluable for both troubleshooting and enhancing system design.

3.4.12 Iteration and Optimisation

Iteration and optimisation are continuous processes that refine the digital twin model based on insights from simulation runs and validation feedback. These processes involve tweaking model parameters, enhancing the fidelity of simulation inputs, and optimising control algorithms to improve accuracy and efficiency. Such iterative refinements help fine-tune the digital twin to mirror the real-world system better, ensuring that each cycle of updates brings the model closer to an optimal state of reliability and performance. The final stage involves maintaining the repository with ongoing updates and modifications in response to end-user requests. As the end user's needs evolve or additional feedback is gathered, changes are made to the master repository hosted on a git version control system. This setup allows the end user to continuously pull the latest updates, ensuring that the digital twin remains an effective and adaptive tool.

3.5 Validation and Verification

In this thesis, the validation and verification process focused on comparing the behaviour of simulations with the real-world counterpart and confirming the suitability of the simulated version as a development platform for deploying tested methods or code to the live system.

By employing these methodologies, the benefits of using digital twins and simulations can be thoroughly validated and verified, ensuring that these technological tools effectively enhance the robotic development process in terms of accuracy, cost-efficiency, and time savings.

3.5.1 Simulation Behaviour Comparison

3.5.1.1 Procedure

The behaviour of simulations was compared with the real-world system by conducting tests to observe similarities and differences in outcomes.

3.5.1.2 Metrics

Qualitative assessment of how closely the simulation replicates real-world behaviours, focusing on motion dynamics, sensor readings, and environmental interactions.

3.5.2 Cost and Time Efficiency Analysis

3.5.2.1 Procedure

Document and compare the time and costs of traditional development processes versus those of simulations and digital twins.

3.5.2.2 Metrics

Calculate savings regarding development time, cost per development cycle, and overall project budget.

3.6 ROS Integration

Integrating the Robot Operating System (ROS) with a new simulation environment is critical in ensuring robust data handling and system interaction capabilities in robotic system simulations. ROS serves as a middleware that facilitates communication among the components of a robotic system and between the system and the simulation environment.

The integration process starts by establishing a communication framework within ROS and involves setting up ROS topics, services, and action servers essential for transmitting and receiving data pertinent to the simulation. For example, integrating a new simulation might

require setting up ROS topics that publish and subscribe to the necessary data types supported by both ROS and the simulation software.

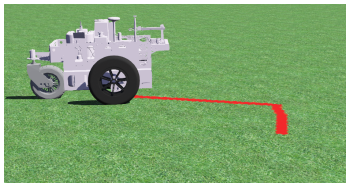
Ensuring the simulation platform supports ROS messages is paramount and often requires developing or configuring plugins or nodes within the simulation software that can handle the ROS communication protocols. These nodes translate simulation data into ROS-compatible formats and ensure seamless data flow between the simulation environment and ROS. This translation and data handling are facilitated by leveraging ROS libraries, which provide the necessary tools for creating these communication channels.

The integration is thoroughly tested to confirm that all components communicate correctly and that the data integrity is maintained across different operational conditions. The testing might include running simulations to verify that the message handling and system responses behave as expected. This phase is crucial for tuning performance and ensuring the simulation reliably serves as a development and testing platform.

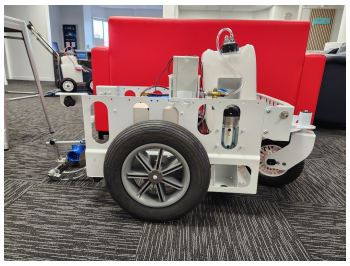
CHAPTER 4

Case Study A - Line Painting Robot Simulation

This case study involved a line-marking robot for sports fields, developed collaboratively between two companies, one handling hardware and the other focusing on autonomous functions like path planning and navigation. Discrepancies in project timelines necessitated the early development of a digital twin, used to refine and validate navigation algorithms based on geo-located markers.



(a) Simulated model from CAD



(b) Real-world initial prototype

(c) Refined prototype after industrial design

Figure 9: Digital Twin and Real-World Prototypes

By designing the pathing algorithm and robot control system alongside the hardware, the project aimed to save time for the senior engineer. The deployment of the first hardware prototype was 1.5 months into the project, a delay worth up to $\$11,520$ in waiting time for the navigation engineer (Appendix A.3).

4.1 Requirements

This case study aimed to facilitate parallel software and hardware development for an autonomous line-marking robot.

4.1.1 Core Requirements

- **Algorithm Testing**: Capability for exhaustive testing of drive control and path planning algorithms within the simulation to pre-emptively refine software behaviours.
- **Visualisation**: Advanced visualisation tools within the simulation to assess algorithm performance and robot behaviour.
- **Hardware Integration**: Designed software compatibility with the hardware components for seamless post-simulation integration.
- **Sensor Precision**: Sensor precision standards for sports field requirements.

These requirements framed the development process, aiming for a software-first approach that would later integrate with the robot's physical components without requiring extensive modification.

4.1.2 Deliverables

The deliverables for this project were designed to meet the core user requirements.

1. **Digital Twin Model**: An approximate digital representation of the robot, enabling detailed simulation of navigation and control algorithms in a virtual environment.
2. **Simulation Environment**: A simulation environment visually replicates real-world sports fields with variable field layouts and potential obstacles.
3. **Virtual Testing Suite**: A collection of tests designed specifically for the Digital Twin, focusing on drive control, navigation, and path planning within the simulation environment.
4. **Documentation on Simulation Integration**: Guidelines and best practices for integrating and calibrating the Digital Twin within the simulation environment, ensuring that virtual performance metrics accurately reflect real-world expectations.
5. **Validation Results**: A comprehensive report on the outcomes of virtual testing, including a detailed analysis of the Digital Twin's performance against predefined criteria and benchmarks, highlighting the efficacy and precision of the simulated model.

4.2 Tech Stack

The development of the DT and its simulation environment leveraged a selective software stack primarily centred around Webots for its lightweight operation and compatibility with low-specification hardware. Webots' inclusion of a simulated Pen device crucially simplified the configuration of the DT for precise line-marking tasks.

4.2.1 Software

- **CAD Assistant & Blender 3D**: The project utilised CAD Assistant to convert the provided STL files into .dae mesh files, which were imported into Blender 3D for necessary file conversion and scaling adjustments. Blender focused on preparing the robot's model for simulation, ensuring the digital twin accurately reflected the proposed physical dimensions and design.
- **Webots Simulation Environment**: Chosen for its comprehensive simulation capabilities, Webots facilitated the virtual testing of the robot within accurately modelled sports fields. It offered an intuitive platform for developing and testing the robot's navigation and control systems, supported by a rich set of simulation tools.

4.2.2 Languages

- **Python 3.8**: The Digital Twin's control algorithms were developed in Python and chosen for their ability to facilitate rapid iteration and development. Python's versatility and the lack of compile-time requirements allowed for efficient algorithm refinement and testing within the Webots environment.

The efficient workflow, facilitated by CAD Assistant and Blender 3D for model preparation and Webots for simulation, enabled seamless model conversion and Digital Twin simulation. Python 3.8's dynamic scripting further streamlined the development and testing of autonomous functionalities, setting the stage for a detailed exploration of the Digital Twin Construction process.

4.3 Simulation Environment

A simplistic simulation world was created in Webots, featuring a solid plane with an integrated repeating grass texture to simulate a sports field environment. Visual detailing was not a high priority because the simulation output was a general test of the pathing algorithms. The simulation environment was a low-fidelity plane with a simple repeating grass texture.

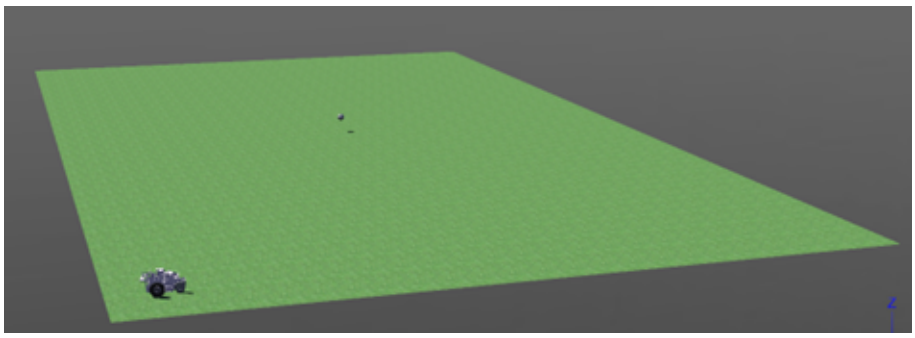


Figure 10: Webots World with Grass Texture

4.4 Robot Digital Twin

This project did not require a detailed environmental simulation; therefore, it focused on creating the robot DT itself—the construction of the digital twin leveraged preliminary `.STL` files from the hardware team were converted into a fully integrated simulation model in Webots. This approach ensured that the digital twin accurately represented its physical counterpart, incorporating essential components for realistic simulation and interaction.

4.4.1 Geometry and Modelling

The mechanical design engineers provided an `.STL` file for the visual element of the robot; however, it should be noted that this visual element was not strictly necessary for the end function of the simulation.

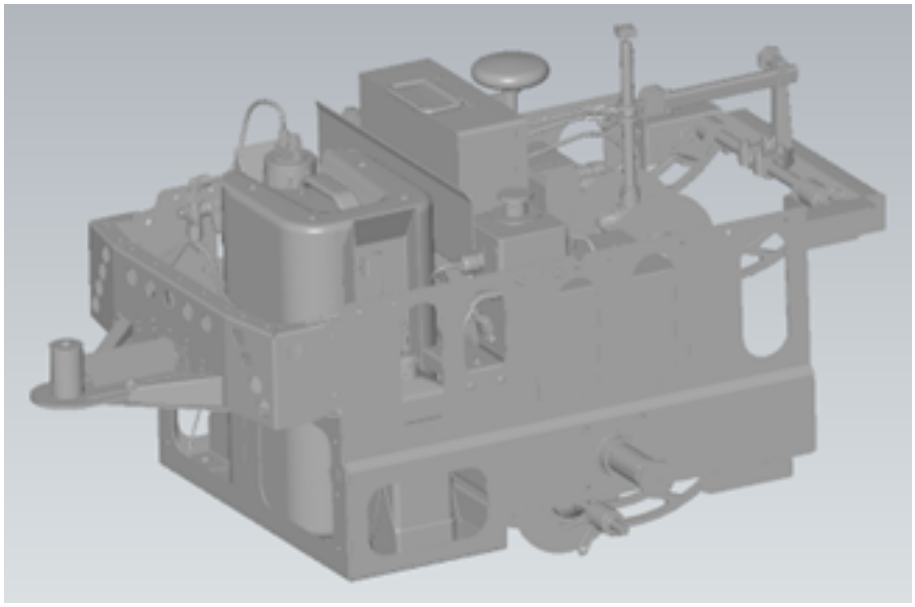


Figure 11: A screenshot of the STL model

The provided `.STL` files were then imported into Blender 3D for essential adjustments, including object origin point alignment and scaling corrections, to ensure accuracy in

representation. The adjusted components were exported as new .dae files prepared for Webots integration.

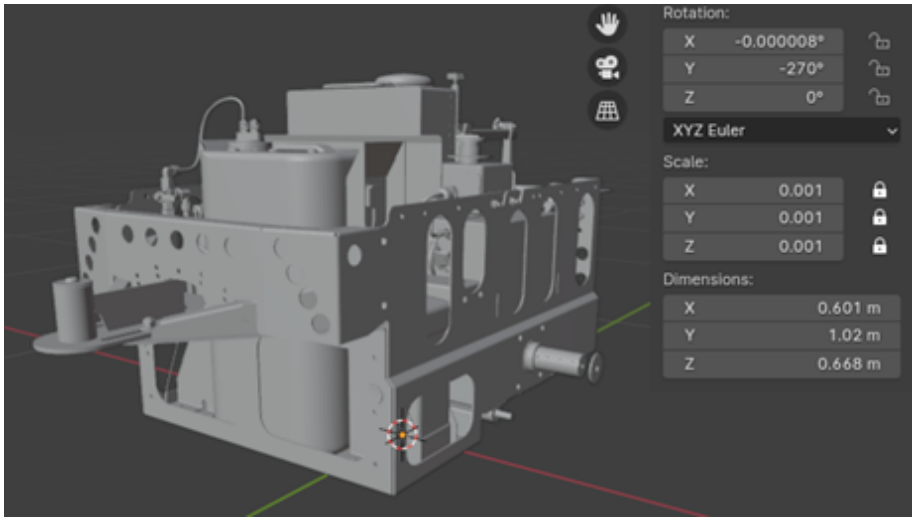


Figure 12: Blender 3D scale adjustments - The mm used in the STL were converted to Blender 3D's standard unit of metres

4.4.2 Custom .Proto File Development

A custom `.Proto` file (Appendix §A- A.1.4) for the robot's Digital Twin was created, encapsulating the robot's critical components. This file included motors, an IMU, GPS, and a Pen device, mirroring the physical robot's capabilities.

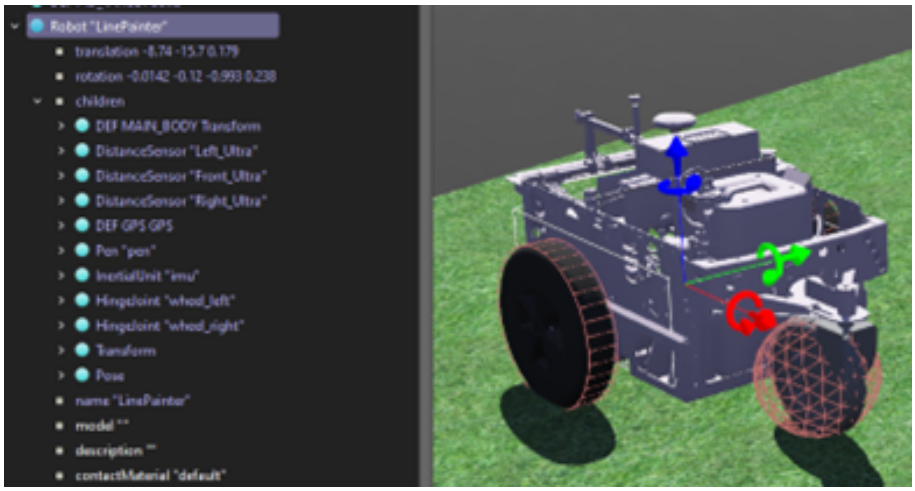


Figure 13: Custom `.Proto` file components and Webots Twin

4.4.3 Sensors

The Digital Twin (DT) calibration process was designed to align simulated sensors' performance with their real-world equivalents. This process mainly focused on calibrating the GPS, IMU, and the spray width of the paint nozzle to meet design specifications and operational needs.

4.4.3.1 GPS Calibration

The simulated GPS was configured to reflect real-world coordinate accuracy, targeting a ±1cm precision to align with hardware specifications. This step was crucial for accurate navigational tasks and path planning in the simulation.

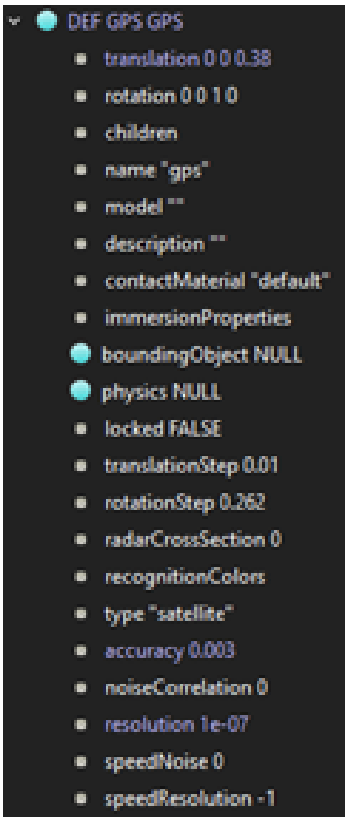


Figure 14: GPS Calibration

4.4.3.2 IMU Precision Tuning

The simulated IMU was tuned as closely as practicable to mirror the physical sensor's orientation and motion detection accuracy. This calibration was essential for maintaining the robot's manoeuvrability and accuracy.

4.4.3.3 Paint Nozzle Calibration

The simulated paint nozzle was configured to accurately represent the physical nozzle's spray width, ensuring line-marking in the simulation matched real-world output, including adjustments such as pressure and nozzle characteristics to maintain line quality.

This calibration ensured that the DT's components operated within expected real-world parameters, facilitating reliable development and testing of control algorithms within the simulation environment.

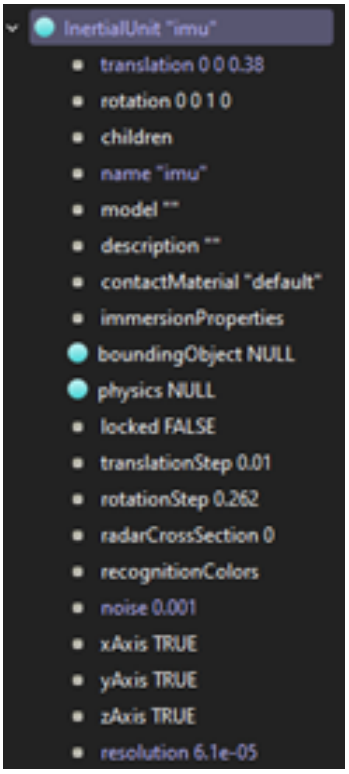


Figure 15: IMU Tuning

4.4.4 Python Controller Scripting

A primary robot controller was scripted in Python (Appendix §A- A.1.2) to facilitate interaction with the Digital Twin, bridging the simulation and the control algorithms.

4.4.5 Waypoint Conversion

The design team's control methodology was translated into a practical application through a Python script that converted a .csv file of line segments into a series of waypoints for the Digital Twin to follow.

4.4.5.1 Algorithm Integration and Testing

Basic waypoint follower routines (Appendix §A- A.1.2) were integrated into the Python controller, allowing for the evaluation and validation of the DT's movement methodology within the simulation environment.

4.5 Integration and Usage

In the final development phase, the integration and usage of the digital twin involved a continuous iterative process characterised by constant communication with the target developer. This phase was methodically structured as follows:

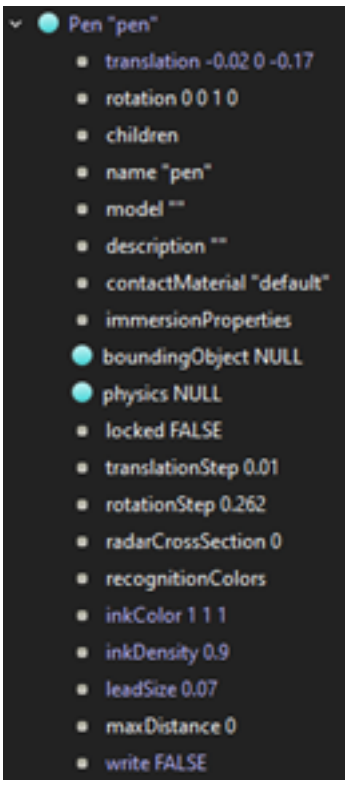


Figure 16: Nozzle Calibration

```
Target Point:    (-7.5000, 15.0000)
Distance to Target:    1.5202m
Distance to Centre:    15.4262m
Bearing Current:    89.9800
Bearing Target:     89.9668
Bearing Centre:     -60.9061
Bearing Perp:       29.0939
Error - Bearing:       -0.0132
Error - Perp Bearing:-60.8861
Error - Radius:        (0.0000 - 15.4262) -15.4262m
Turn Type:          [None]
Drive Type:         [Travelling Fast]
cmd_vel:            (4.0000, 0.0018) [0.0]
```

Figure 17: Real-time Python controller driver output

1. Receiving Data: Unlike typical deployments, the simulation was not provided directly to the end-user. Instead, the end-user sent spreadsheets containing line segments based on the sports fields to be marked (See Appendix §A- A.1.5).

2. Data Extraction and Processing: Upon receiving the data, the simulation's algorithms extracted the line segments at runtime.

3. Waypoint Calculation: The algorithms then calculated the necessary waypoints from the extracted line segments.

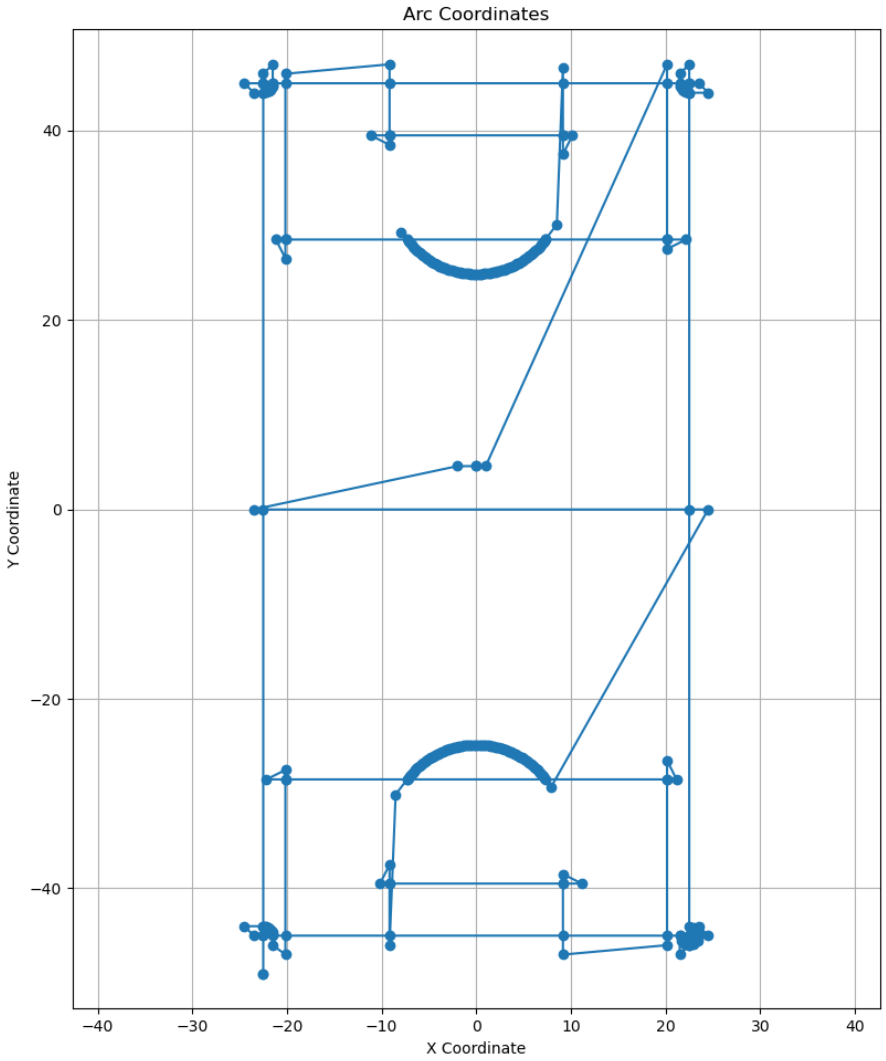
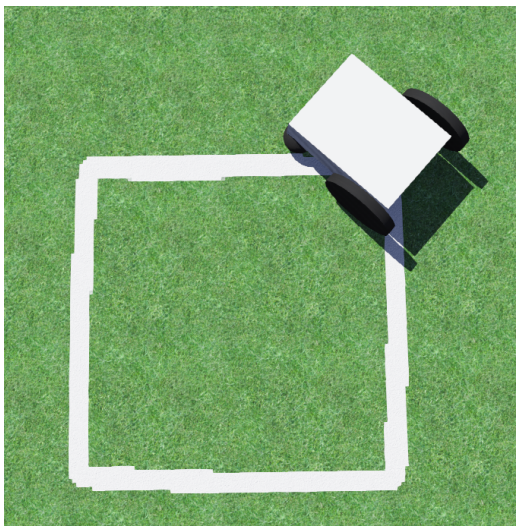


Figure 18: Waypoint Conversion Script

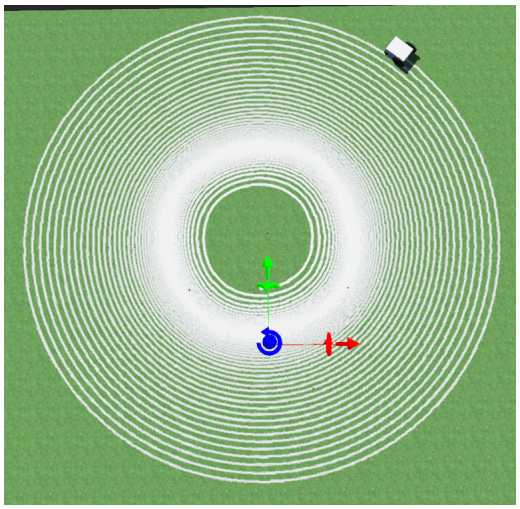
4. Path Execution: Following the calculation, the simulation executed the path using the same logic that would later be successfully implemented in the physical prototype.

This structured process underscores the simulation environment's role as a testing ground for the algorithms and as a direct interface for iterative feedback and refinement in collaboration with the end-user. By refining the algorithm based on user requirements and field data, the approach ensured that the developed solution was both practical and efficient before its physical implementation.

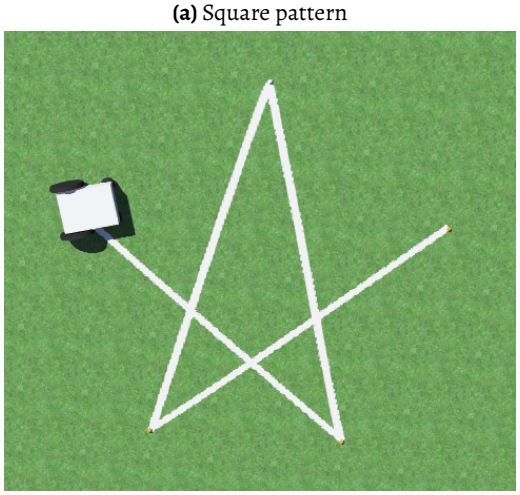
The iterative nature of this approach allowed for constant tuning and adaptation of the navigation and control algorithms, ensuring they were optimally adjusted for real-world operations. Developers and the end-user collaboratively evaluated and refined the system's responses to real scenarios, enhancing the accuracy and reliability of the line-marking robot prior to its deployment in physical prototypes. This method saved significant development time and ensured that the digital twin closely mirrored the operational needs and constraints of the



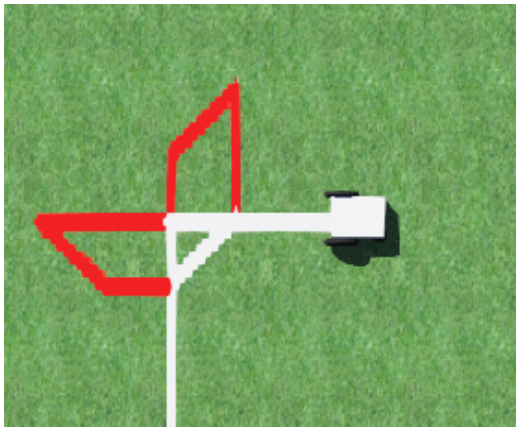
(a) Square pattern



(b) Spiral pattern



(c) Star pattern



(d) Corner painting

Figure 19: Testing waypoint follower routines (a) square (b) spiral (c) star (d) a painted corner with red representing inactive painting nozzle allowing the robot to stabilise its movement before beginning a line.

end-use environment. The `.Proto` file was imported into the Webots simulation world, situating the Digital Twin within the virtual sports field environment (figure 20).

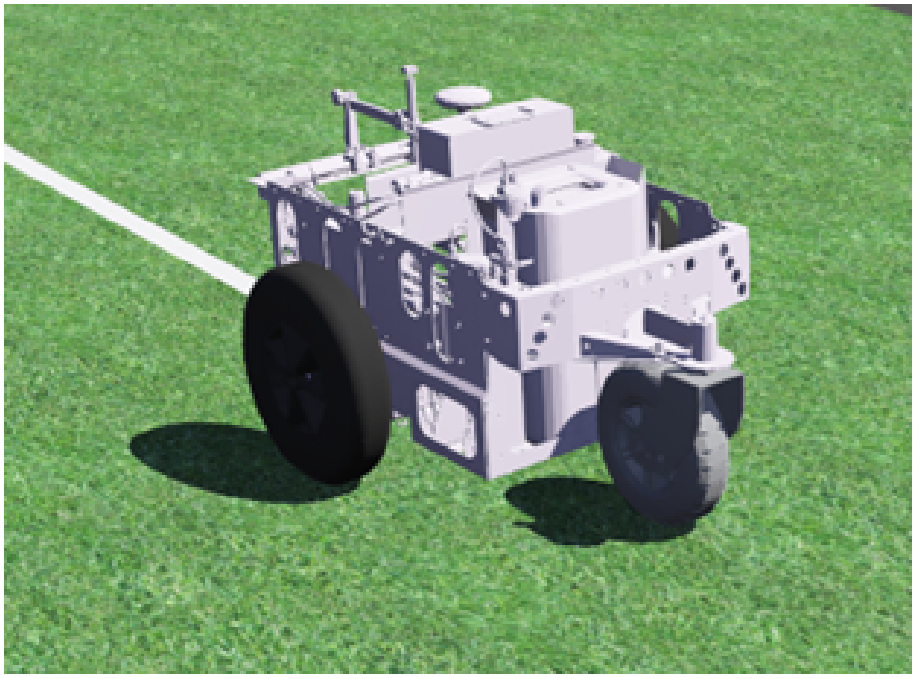


Figure 20: Digital Twin in Simulation World

CHAPTER 5

Case Study B - QR Code Localisation

Case B accompanies a robot development project for an indoor inspection-class mobile robot using April Tags for indoor localisation in a GPS-denied environment. The challenge for this case centred on validating a localisation algorithm using QR codes (specifically April tags) situated throughout an industrial building, necessitating a simulated environment with high visual fidelity and ROS 2 integration.

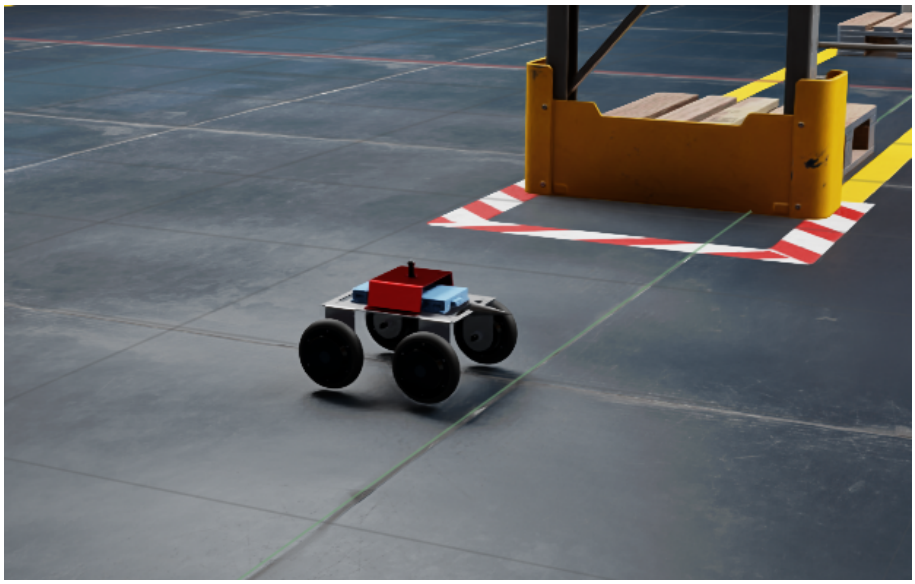


Figure 21: Camera robot in a simulated warehouse environment

5.1 Requirements

This case study aimed to establish a testing framework for QR code identification and robot localisation algorithms within an indoor GPS-denied environment.

5.1.1 Core Requirements

- Test Environment: Develop and evaluate a visually realistic environment and workflow to test a QR Code fiducial-based localisation system without physical assets.
- High-Fidelity Visual Simulation: The ability to realistically represent indoor environments with high visual fidelity, high-quality camera simulation, and QR code integration.
- Camera Integration: Incorporation of a simulated 360-degree camera, mirroring real-world camera specifications, to aid in indoor localisation.

These requirements emphasise the importance of a visually detailed simulation environment for testing and refining indoor localisation strategies, guiding a software-first development approach before any physical application.

5.1.2 Deliverables

- Simulation Environment Setup: A ROS 2 compatible simulation environment that supports detailed environments.
- 360-Degree Camera Simulation: A detailed simulation of a Ricoh Theta V 4k 360 Spherical Camera integrated into the system to provide realistic indoor vision capabilities, replacing traditional GPS methods.
- Robot Model in Suitable File Format: A precisely modelled digital twin of the robot, created in a format suitable for Simulation to facilitate accurate algorithm testing and development within the simulation environment.
- Integration and Calibration Documentation: Detailed guidelines for integrating and calibrating the digital camera model and robot platform within the Simulation, ensuring the simulated system's behaviours align with expected real-world performances.

5.2 Tech Stack

The Digital Twin (DT) development and simulation environment utilised a targeted selection of software and programming languages, focusing on creating a realistic and interactive simulation.

5.2.1 Software

- Blender 3D: Blender 3D was used to convert CAD `.STEP` files into .dae mesh files, adjusting for scale and orientation where necessary.

- Nvidia Omniverse: Chosen for its high visual fidelity and ROS 2 integration, Omniverse provided a photorealistic environment for simulating the robot's behaviours and interactions, significantly enhancing the quality of the Simulation. Additionally, the camera simulation is extensively configurable.
- Isaac Sim ROS 2 Connector: Isaac Sim has a built-in connector that connects a ROS 2 instance to the simulator through its Action Graph system.
- ROS 2: Served as the middleware control system, facilitating communication and control between the DT components and the simulation environment. Its robustness and flexibility were crucial for real-time operation and interaction within the Omniverse platform.
- QR Codes: April Tag Family (included in Isaac Sim)

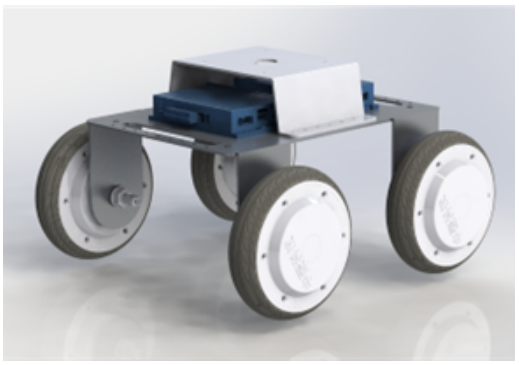
5.2.2 Languages

- C++ and Python 3.8: These languages were used for developing ROS 2 nodes and launch files, combining C++'s performance efficiency with Python 3.8's rapid development capabilities. This mix allowed for real-time responsiveness and flexible adjustments in the Simulation.

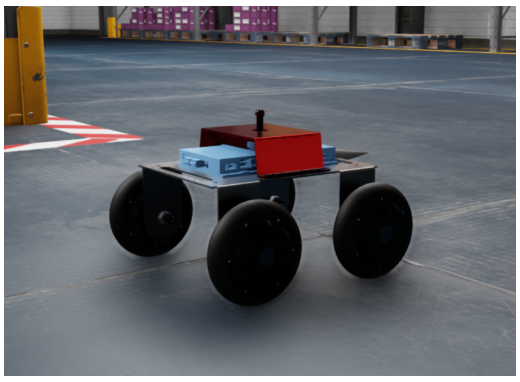
5.3 Robot Digital Twin

5.3.1 Geometry and Modelling

A simple sheet-metal design was constructed in Solidworks, exported as a `.STEP` file, and imported directly into a standalone `.USD` file using Isaac Sim to construct the robot from components.



(a) Solidworks rendering of the placeholder robot



(b) The step file was imported and re-assembled in Isaac Sim as a `.USD` file

Figure 22: Robot Digital Twin Geometry and Modelling

5.3.2 Simulated Camera Configuration

A critical aspect of the DT construction was the calibration and configuration of onboard sensors, particularly the camera, which plays a vital role in localisation and navigation tasks.

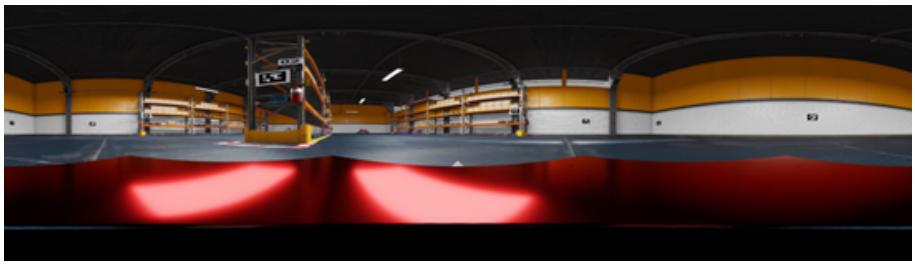


Figure 23: View from the configured simulated camera

The camera was configured to mimic an actual device's specifications and performance, such as adjusting the field of view, resolution, and other parameters, to ensure the DT could accurately simulate real-world navigation and identification tasks within the warehouse. Figure 24 shows the camera configuration settings in the Isaac Sim environment.

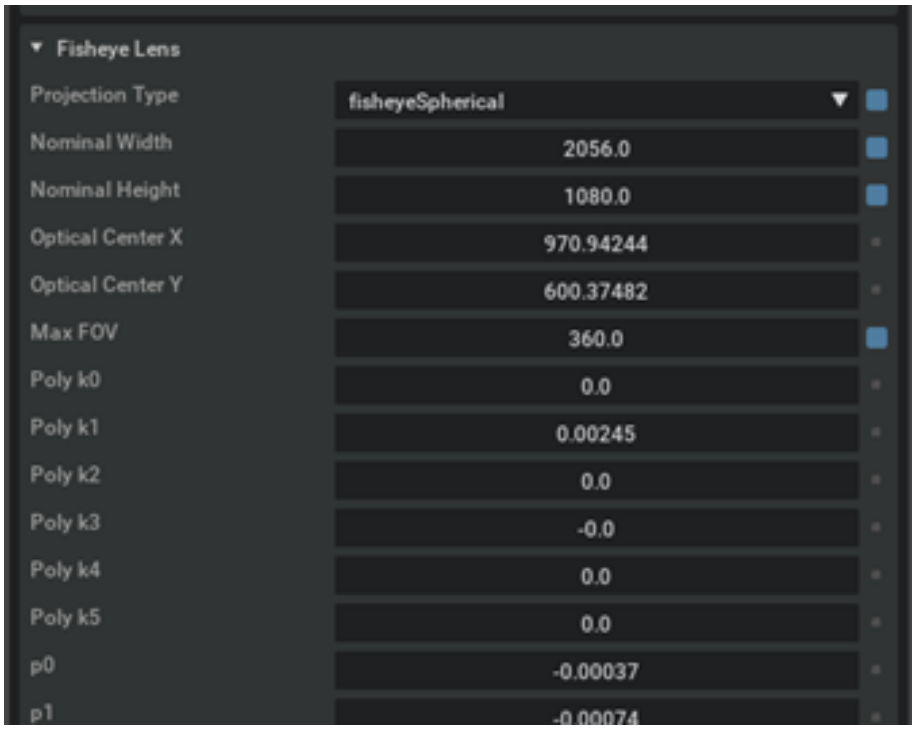


Figure 24: Camera configuration in Isaac Sim

5.3.3 ROS 2 Integration

An Nvidia Action Graph was constructed to enable communication between Isaac Sim and ROS 2 to enable control and data pipelines with the Digital Twin to the robot control system (Figure 25).

Isaac Sim implements camera sensors as a Viewport, using the ROS 2 connector. Figure 26 shows the ROS 2 Camera Helper Node configuration.

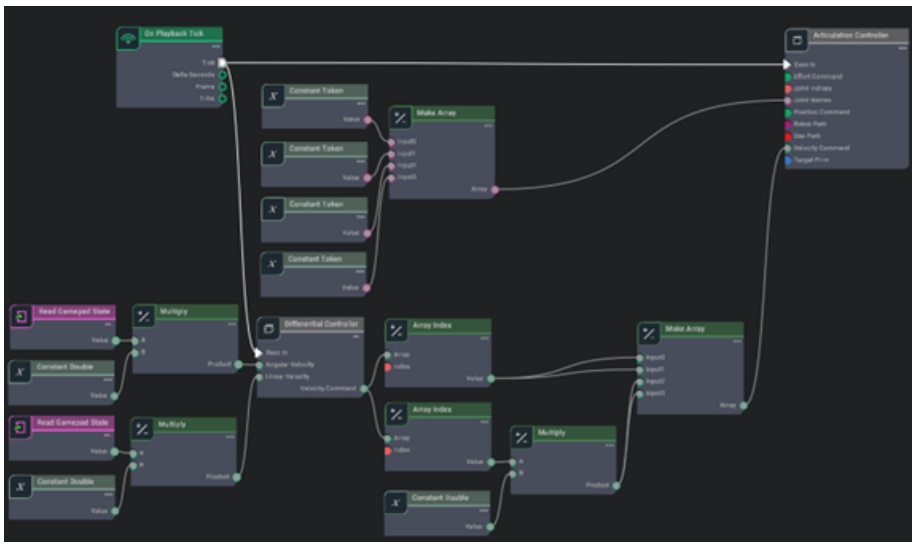


Figure 25: Isaac Sim Action Graph - Teleoperation

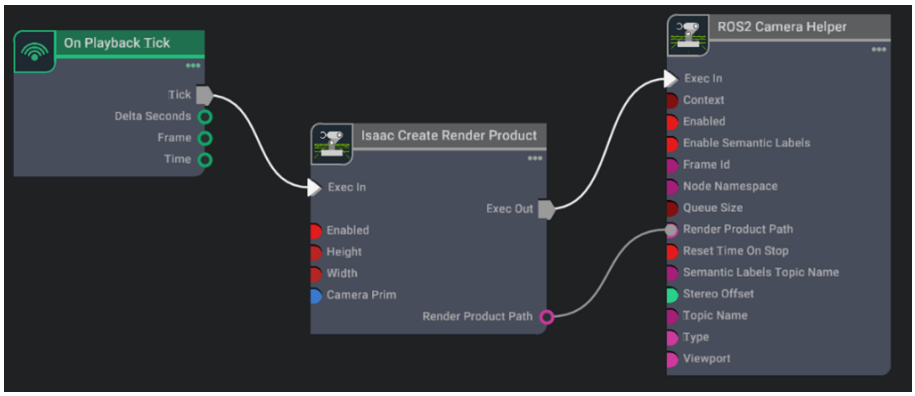


Figure 26: Isaac Sim Action Graph ROS 2 Camera Node

Node	Input Field	Value
Isaac Create Render Product	cameraPrim	/World/Camera_1
	enabled	True
ROS2 Camera Helper	type	RGB
	topicName	RGB
	frameId	turtle

Table 5.1: ROS2 Camera Helper breakdown

5.4 Simulation Environment

The simulation environment for this project was crafted within the Nvidia Omniverse ecosystem, leveraging an existing sample warehouse environment. This pre-designed environment offered a comprehensive and realistic backdrop, mimicking a typical warehouse's visual complexities. This foundational setting was crucial for testing the DT's indoor visual capabilities, ensuring the robot's interactions and navigational decisions could be accurately evaluated against real-world behaviour.

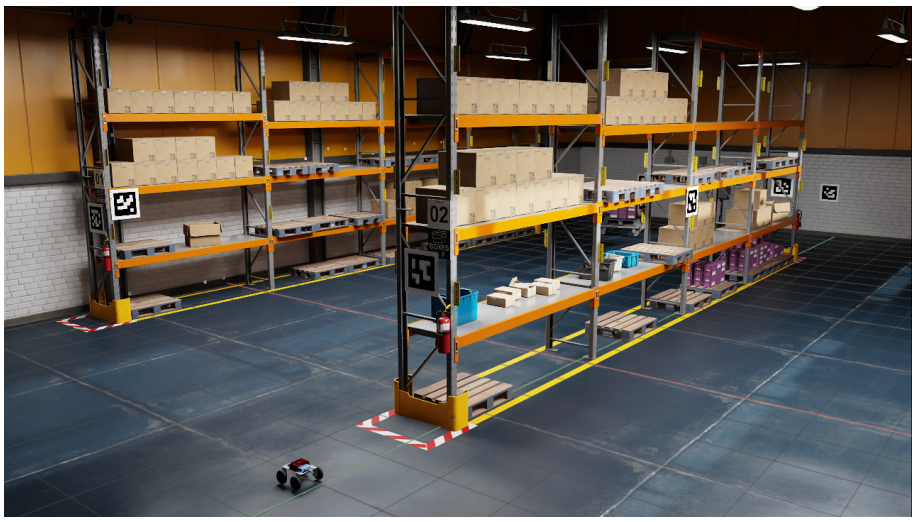


Figure 27: Warehouse environment in Omniverse with the robot model

5.4.1 QR Code Localisation Strategy

The system was designed to utilise April tags for localisation within the GPS-denied industrial setting. The selection of April tags was driven by their robustness and reliability in indoor localisation tasks, serving as fixed reference points for the robot's navigation system.

5.4.1.1 Tag Selection

The 25h9 family of QR codes was used, as it was shown through early testing that the camera's resolution could not identify the initially selected 16H5 family as effectively at the required distance (15m).

5.4.1.2 Tag Positioning

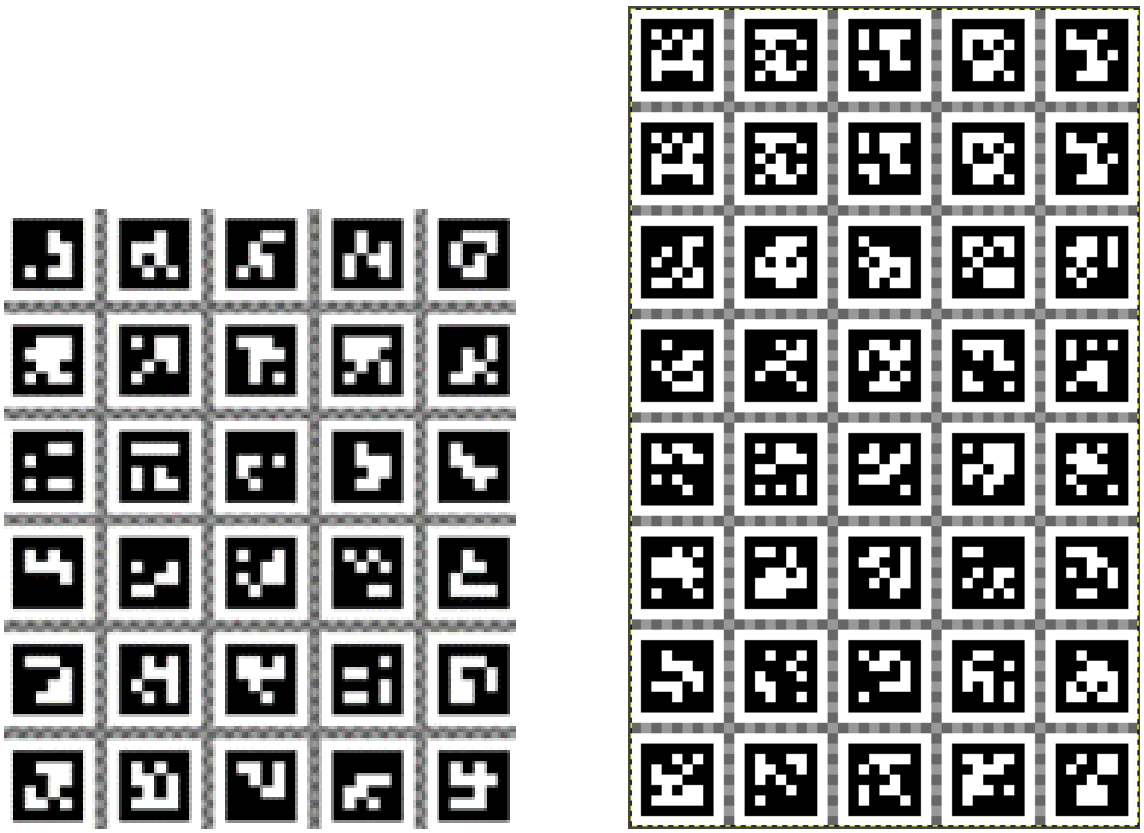
An initial set of tags was added to the environment in initially sparse numbers, adding additional tags based on feedback from the algorithm team.

5.5 Algorithm Testing Workflow

The primary focus of this project was to allow the validation of a vision and fiducial-based localisation algorithm as a proof of concept before a client proposal was confirmed. Due to hardware limitations on the part of the algorithm developer, this project was conducted by sharing ROS 2 bag files with the algorithms engineer and receiving feedback and results data.

5.5.1 Setup

- ROS 2 interface for robotic control initialised.
- Isaac Sim simulation environment activated.



(a) A subset of the 16h5 family QR code family

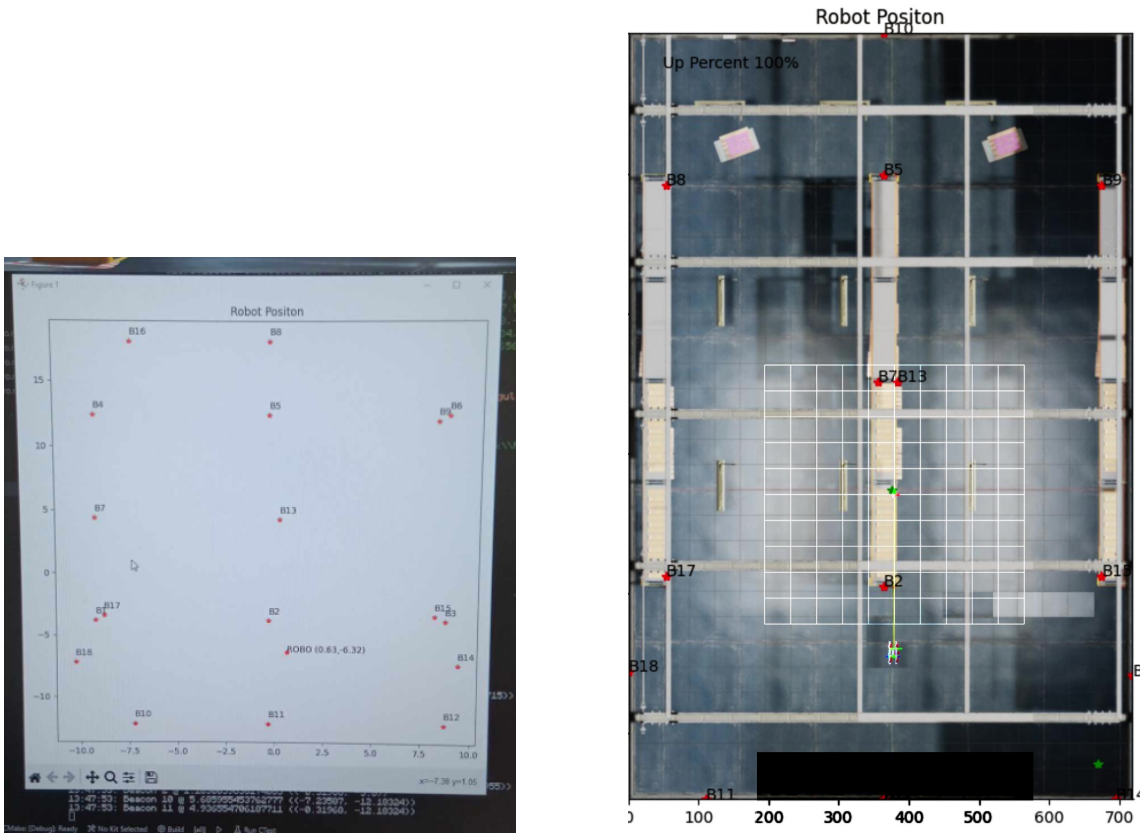
(b) A subset of the 25h9 family QR code family

Figure 28: Overview of the two considered April Tag families.

Table 5.2: Coordinates of Markers

#	x	y
1	-9.23103	-3.8
2	-0.31968	-3.8
3	8.6985	-3.8
4	-9.23103	12.40249
5	-0.31968	12.40249
6	8.6958	12.40249
7	-0.92174	4.39041
8	-8.80214	11.99243
9	8.15625	11.99238
10	-0.31968	17.97719
11	-7.23587	-12.18324
12	-0.31968	-12.18324
13	0.2193	4.28025
14	8.6985	-12.18324
15	8.15853	-3.37909
16	9.36434	-7.31419
17	-8.79823	-3.37978
18	-10.26566	-7.1682
19	-7.33832	17.98589

- Robot teleoperation enabled.



(a) Registered QR Code coordinates with IDs

(b) Locations overlaid on warehouse top-down view

Figure 29: QR Code Positioning

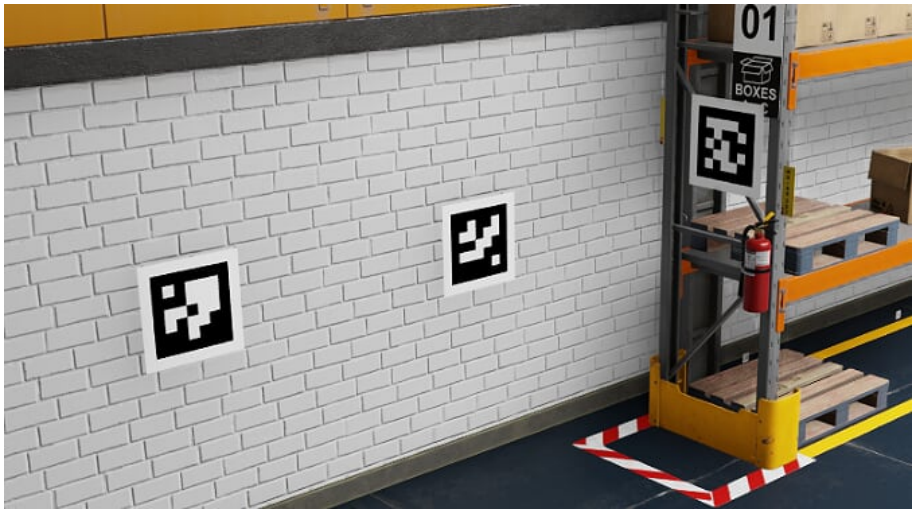


Figure 30: Simulated April tags in the warehouse

5.5.2 Process

1. Start Bag Recording:
    - Run command `rosbag record -a {filename}` to capture all topics published during the simulation.
    - Specify the filename for the bag file.

2. Teleoperate Robot:
    - Manually control the robot in a typical route in a circle around the central shelving unit.
    - Ensure the robot returns to the initial position.
3. Stop Bag Recording:
    - Terminate the bag recording.
4. Provide Bag File to Engineer:
    - Share the recorded bag file with the algorithms engineer for independent analysis and feedback.
5. Receive Feedback and Results:
    - Get feedback and results data from the engineer.
6. Refine Simulation World:
    - Iterate on the simulation world based on received feedback.
    - Repeat the process as necessary to refine the Simulation.
7. Validate and Improve Simulation:
    - Use the collected data and feedback to validate and improve the Simulation.

5.5.3 Completion

- Validated vision and fiducial-based localisation algorithm.
- Refined simulation world.
- Collected data and feedback for future improvements.

CHAPTER 6

Case Study C - Forestry Machine Simulation

This case explores the development of an autonomous control system for a Green Climber LV600 Remote Controlled Mower (a large Italian-made tool carrier) used for forestry operations. This project aimed to help increase time efficiency and safety by minimising the role of human operators in cutting and maintaining forestry tracks.

Figure 31: Green Climber LV600 Tool Carrier with a forestry mulcher attachment

The tool carrier (Figure 31) was a hydraulically powered tracked machine with an integrated remote-control (RC) system. An early technical challenge for this project was the non-linear

control relationship between the linear and angular velocities when controlling the machine via the integrated RC system.

Figure 32: Outdoor testing setup

Each test represented a considerable investment in financial resources, logistical planning, and personnel, with expenses reaching up to $1291.00 (Table 6.1) for a single trip to a remote forest location. During the development of the drive control system, it became necessary to maintain a presence in a semi-rural area with access to farmland for approximately two weeks (Figure 32). This temporary outdoor office configuration increased fuel costs, decreased development ergonomics and productivity, and led to significant dead time for developers waiting their turn to contribute.

Table 6.1: Daily Test Cost

Factor	Cost (Daily)	Quantity	Total Cost
Machine Rental + fuel	385.71	1	385.71
Engineers	344.00	2	688.00
Fuel	28.57	1	28.57
Truck Hireage	189.00	1	189.00
Total		**Daily**	1,291.28

Figure 33: Field Testing logistics – Vehicles, remote locations, and personnel

An additional challenge posed by field testing is the inherently destructive nature of tracked vehicles. Unfortunately, iterative testing in smaller outdoor areas substantially damaged grass and other ground surfaces after repeated testing, especially after recent rain (Figure 34).

Figure 34: Environmental damage from repeated field testing

Initially completed without the aid of digital twins or advanced simulations, this case study retrospectively explores the quantified potential advantages of integrating these technologies from the project's outset. It delves into how digital twin and simulation integration could have mitigated the steep costs and logistical complexities of field testing, offering insights into the transformative impact of such technologies on the early-stage development of robotic systems.

6.1 Requirements

The core requirement for this simulation use case was to reduce the frequency of field tests and associated costs. The aim was to establish a testing workflow for the tool carrier in a simulated

environment that could replicate real-world challenges without extensive physical testing.

6.1.1 Core Requirements

- **Realistic Simulation and Visualisation:** The ability to realistically simulate and visualise the Digital Twin (DT) behaviour.
- **Code Reusability:** The ability to use the same finalised code on the actual machine and the simulation alternately.
- **Dynamic Updates:** The ability to easily update the DT on request and distribute updates to users.

6.1.2 Deliverables

To meet these requirements, the deliverables included:

- **Digital Twin of the Tool Carrier:** A representative model of the machine that roughly approximates the performance and behaviour of the hydraulic motors.
- **ROS Node and Protocol Integration:** An API or interface enabling the end user to connect the ROS control nodes with the simulated hardware using the same protocols and methodologies for the final system, allowing for a hot-swappable codebase to the live system.
- **Simulation Environment:** An appropriately configured simulation environment for visualising the DT.
- **Version Control System:** An appropriate version control system to synchronise and distribute updates to end-users upon request.
- **Workflow Guide:** A guide for the end-users describing how to configure and use the system.

The project aims to streamline the development process by addressing these requirements and deliverables, reducing dependency on costly and logistically complex field tests, and enhancing robotic system development's overall efficiency and sustainability in forestry applications.

6.2 Tech Stack

This section outlines the tools, software, and programming languages chosen for the project, providing justifications for each choice based on the project's specific needs as described in the general methods section.

6.2.1 Tools

- **iPhone 12 Pro with LiDAR Scanning Capabilities:** The iPhone 12 Pro was chosen primarily for its advanced LiDAR scanning capabilities, which enable high-resolution and accurate 3D scanning of physical environments. This tool is particularly useful for quickly capturing the detailed geometries of the forestry tool carrier and its operating environments without needing more cumbersome and expensive 3D scanning equipment. The portability of the iPhone allows for convenient onsite data collection, which was a high priority in forestry locations.

6.2.2 Software

- **Cyberbotics Webots:** Webots was selected as the primary simulation platform due to its comprehensive simulation tools and ease of use, facilitating rapid prototyping and iterative testing. Its robust ROS integration capabilities are crucial for developing and testing the control algorithms in a simulated environment that mirrors the ROS-based control setup used in the tool carrier.

- **Blender 3D:** Blender is utilised to refine and edit the 3D models generated from LiDAR scans. Its powerful modelling tools enable precise manipulation of mesh data to ensure the digital twin accurately reflects the physical characteristics of the tool carrier. Blender's capabilities in texturing and rigging are also beneficial for enhancing the visual realism of the simulation models.

- **GitHub:** GitHub serves as the version control system for this project. It is essential for managing code changes, tracking issues, and collaborating between different team members. GitHub facilitates the efficient distribution of updates and maintains the integrity of the project's codebase, ensuring all team members have access to the latest software revisions and resources.

- **Polycam:** Polycam is a photogrammetry processing app used alongside the iPhone's LiDAR capabilities to enhance the reality capture process. It creates textured 3D models from photos, which can be integrated into the digital twin to provide additional visual accuracy and context. This approach is particularly useful for capturing environmental details that LiDAR alone might miss.

6.2.3 Languages

- **C/C++:** C++ is employed for its performance efficiency and control over system resources, which is critical in developing lower-level control systems that interact directly with the hardware interfaces of the digital twin. Using C++ ensures the simulation can handle the computational demands of real-time physics calculations and complex control algorithms without significant lag.
- **Python:** Python is used for higher-level simulation scripting and data processing in Webots due to its simplicity and extensive library support. It allows for rapid development and testing of control strategies, sensor data processing, and integration with other software tools like Blender and GitHub. Python's readability and versatility make it an excellent choice for scripting within the simulation environment, where changes might need to be made quickly and efficiently based on testing outcomes.

6.3 Robot Digital Twin

6.3.1 Geometry and Modelling

The tool carrier was scanned using the Polycam app and the iPhone LiDAR scanner to obtain an initial mesh representation (Figure 35). Post-scanning, the mesh was cleaned with a series of cleaning and optimisation steps, primarily executed in Blender 3D.

Figure 35: Polycam scan of the tool carrier

Following the capture, the 3D mesh underwent a series of cleaning and optimisation steps in Blender 3D. These steps were crucial to refining the model and enhancing its simulation

suitability. Notable modifications included the removal of the tracks from the initial model. These were later replaced with simulated track components within the simulation environment to allow for better control and interaction within the DT (Figure 36).

Figure 36: Cleaned mesh in Blender 3D

A streamlined Digital Twin of the tool carrier was assembled in Webots. The cleaned-up mesh from Blender 3D served as the foundational element. In addition to the mesh, a Webots' Track Group Node was added to simulate the machine's tracks, offering a more streamlined yet effective representation of the tool carrier's locomotion system (Figure 37).

The Digital Twin was also placed within a LiDAR-scanned forest environment to simulate real-world operating conditions. This environment was created from LiDAR data captured in a similar forest setting, ensuring the simulated conditions closely mirrored actual forestry operation areas. The integration of the tool carrier into this more visually accurate environment allowed for comprehensive visualisation of expected behaviour (Figure 39). For this use case, the 3D forest environment was originally scanned for purely visualisation purposes and was repurposed for the simulation environment.

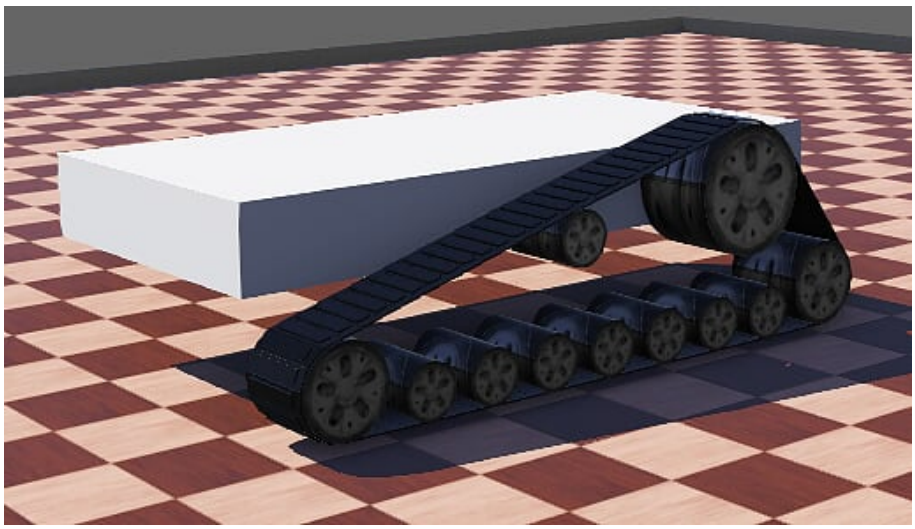


Figure 37: A simplified track device in Webots

Figure 38: Webots full sim-ready Digital Twin

6.4 Simulated Drive Control

A Webots Python controller was written to emulate the system input commands with a gamepad. The intention was to eventually replace the controller input with that of a ROS navigation algorithm.

6.4.1 Initial Setup and Devices Configuration

The simulation employs the Webots robotics simulator, which defines crucial peripherals such as motors, a compass, GPS, and an Inertial Measurement Unit (IMU). These devices are essential for gathering real-world data to navigate and orient the robot within the virtual environment. The Python script (Appendix §A- A.3.2) initiates by setting up the robot and ensuring the joystick

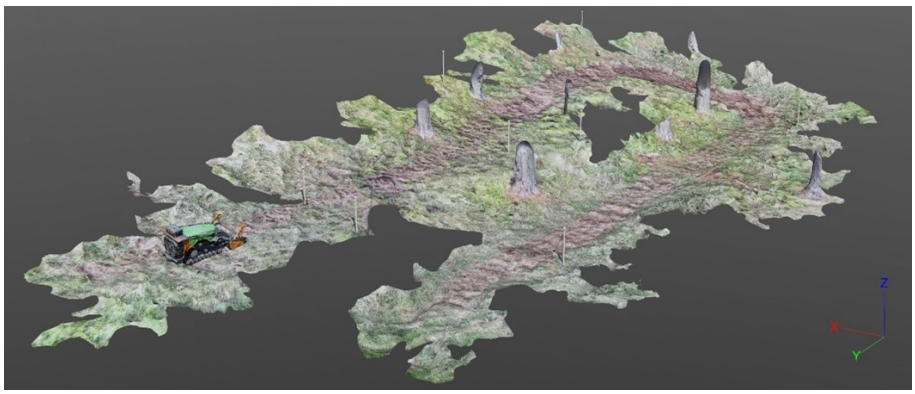


Figure 39: The robot integrated into a LiDAR-scanned forest environment

interfaces are operational and appropriately mapped to influence the robot's movements. The script handles the ROS `cmd_vel` message format as a standardised protocol for velocity commands, anticipating future integration with ROS 2 systems.

6.4.2 Joystick Input Processing and Motor Control

Joystick inputs are scaled to power magnitudes ranging from −1.0 to 1.0, directly controlling the robot's forward and angular motions. The script dynamically adjusts steering based on the current forward power input to realistically simulate the hydraulic power distribution and mechanical load characteristics typical of heavy forestry equipment. This simulation includes a randomised "terrain factor" that introduces variability in steering effectiveness, simulating real-world driving challenges like terrain ruggedness or slippery conditions. The custom robot controller written in Python, using the Webots Robot API (Appendix §A- A.3.2), directly handles these commands, providing a platform for developing movement controllers and interfacing with the emulated hydraulic system logic.

6.4.3 Advanced Control Logic Implementation

The system uses a power reservoir relationship to modify the effectiveness of steering commands based on the forward speed, emulating the total power capacity of the hydraulic system. The adjusted steering and drive commands are converted into motor velocities for the robot's tracks. This methodology ensures that the simulated robot behaves similarly to its real-world counterpart and provides a platform to develop and test control strategies that could be applied to robotic systems.

Additionally, the script supports a state machine setup where different modes of operation can be toggled, such as joystick control or automated ball-following (figure 40), for initial movement control validation. Error handling and system feedback loops are implemented to

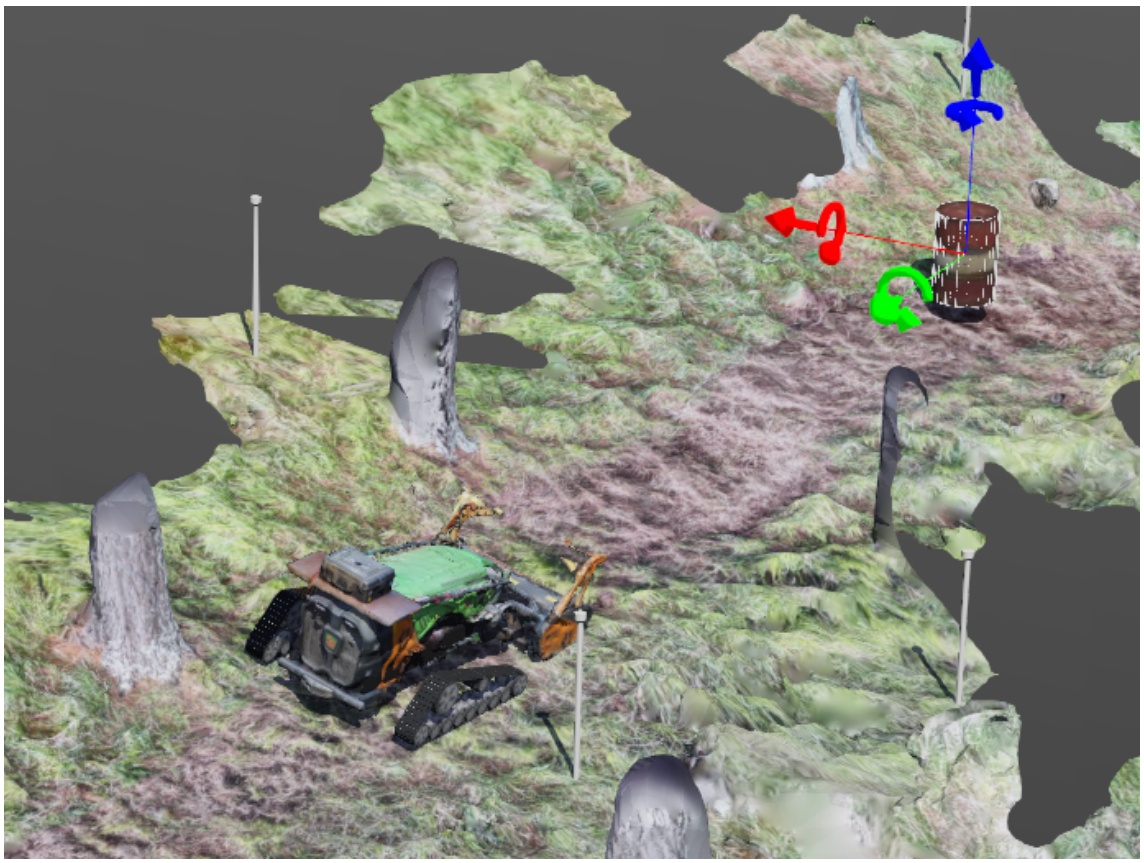


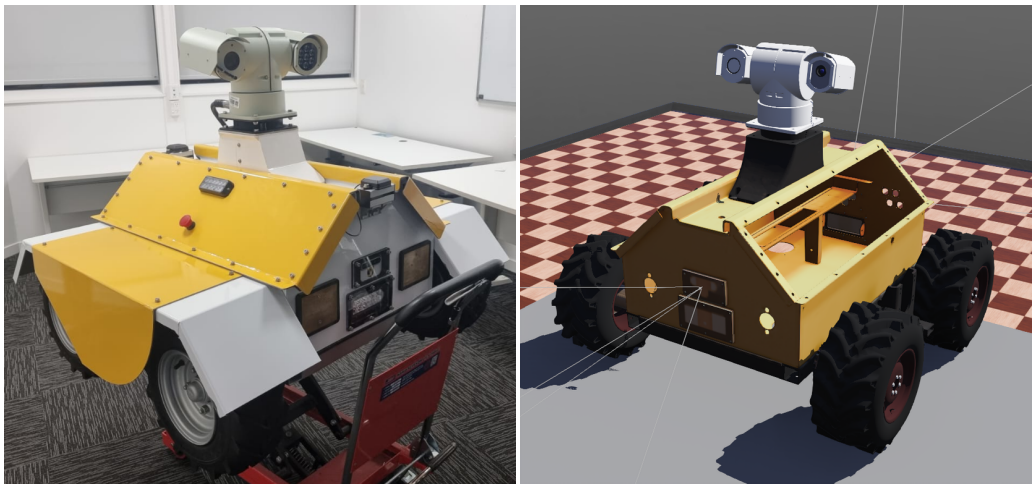
Figure 40: The robot following a barrel

refine control outputs continuously based on the positional and orientation data, thereby enhancing the accuracy and reliability of the robot's navigation capabilities in the simulation.

CHAPTER 7

Case Study D – Inspection Robot

This case study concerns developing and deploying a four-wheel skid-steer inspection robot designed for autonomous operations within a power substation. It is equipped with sophisticated devices and sensors—including dual colour and depth cameras, an inspection-grade PTZ camera, RTK GPS, and an IMU. This project aimed to reduce the staffing required for remote asset inspection in hazardous environments.



(a) Inspection robot prototype under construction **(b)** Inspection Robot Digital Twin in Webots

Figure 41: Inspection Digital Twin

Significant operational delays were anticipated, given the tight project deadlines and the division of tasks between hardware and software teams. The software development depended on the hardware team's progress, creating bottlenecks in the prototype assembly and integration phases. A simulation was proposed to allow for parallel development, thus optimising the workflow, minimising downtime and addressing these challenges.

7.1 Requirements

7.1.1 Core Requirements

- **Parallel Development:** Facilitate concurrent software and autonomy development alongside hardware assembly without waiting for physical components to be completed.
- **End-stage Visualisation:** Enable visual demonstration of the robot's operational capabilities in its intended environment for client presentations.
- **Collaborative Development:** Support multiple developers with local instances of the digital twin, synchronised through a central system for consistency and collaborative efficiency.

7.1.2 Deliverables

- **Simulated Digital Twin:** Create a high-fidelity digital twin of the robot within a simulated environment appropriate for evaluating its autonomy capabilities.
- **Integrated System Interface:** Develop a system interface that integrates seamlessly with the final ROS 2 system, ensuring that the code developed for the digital twin is directly transferable to the actual robot.
- **Version-Controlled Development Environment:** Establish a portable, version-controlled repository for the digital twin and simulation environment, allowing multiple end-users to operate while maintaining alignment with central updates.

7.2 Tech Stack

7.2.1 Software

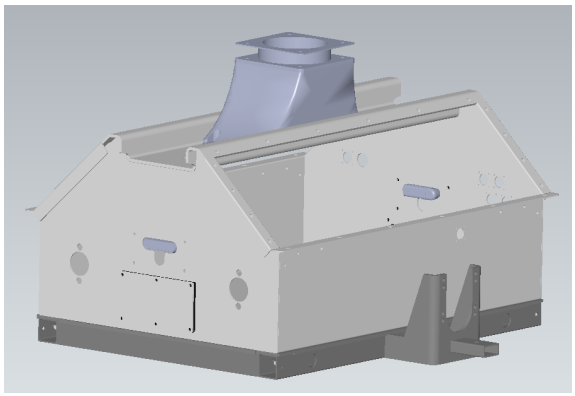
- **Solidworks 3D:** Converts CAD designs of the robot chassis and PTZ camera into STEP files.
- **CAD Assistant:** Utilised to convert STEP files into .obj files.
- **Blender 3D:** Employed to convert .obj files to .dae files, suitable for use in Webots and material applications.
- **Webots Simulation:** Chosen to simulate the digital twin and its environment.
- **Foxglove:** Visualise ROS 2 topics and transforms.

7.2.2 Languages

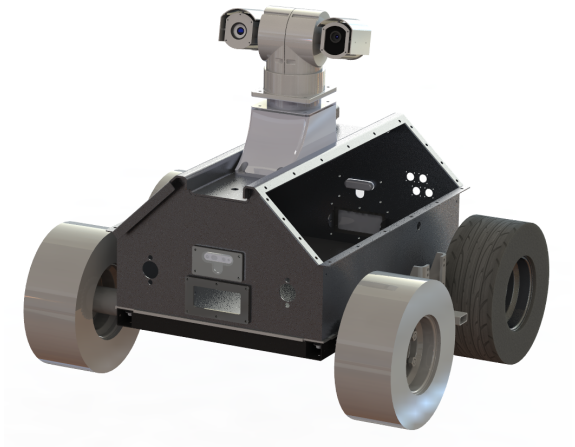
- **Python:** Used for scripting ROS 2 launch files and handling simulation interactions.
- **C++:** Applied in developing ROS 2 hardware interfaces within Webots.

7.3 Robot Digital Twin

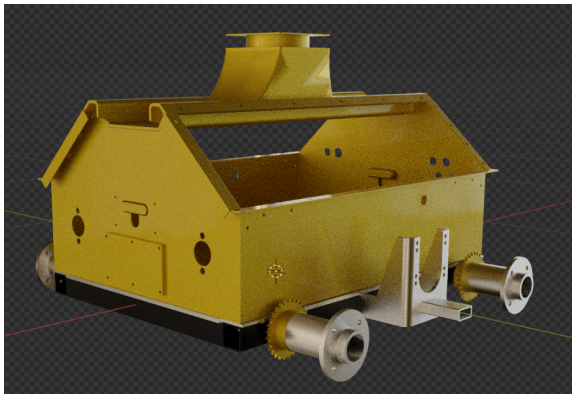
7.3.1 Geometry and Modelling



(a) Base CAD model of the robot prototype in Solidworks 3D.



(b) Assembled Solidworks 3D model of the robot assembly.



(c) Digital Twin chassis with materials added for export to Collada .dae format.

Figure 42: Geometry and Modelling Stages

7.3.2 Sensors

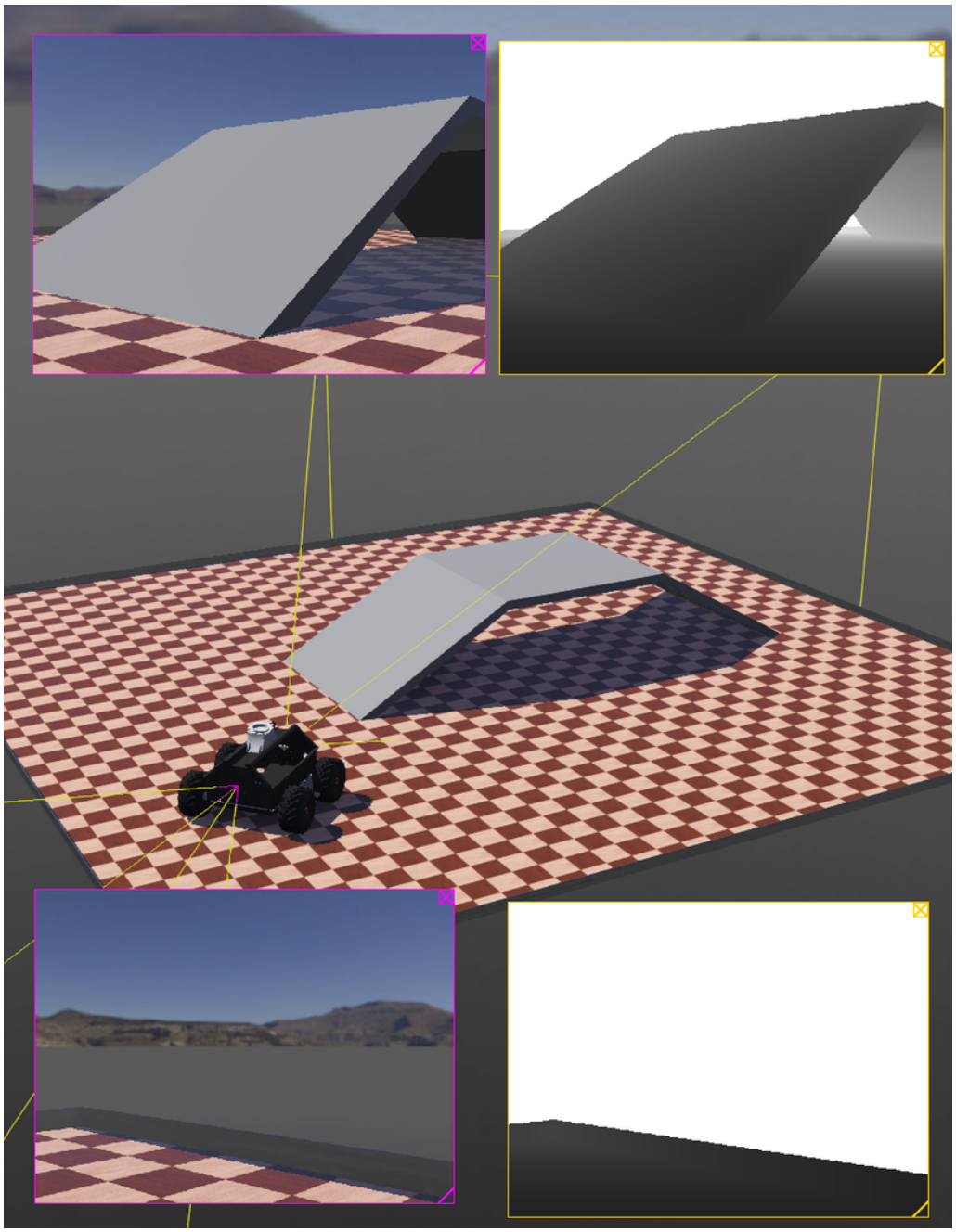


Figure 43: Four streamed simulated camera feeds from the Digital Twin

The inspection robot is equipped with various simulated sensors, including two Intel RealSense colour and depth cameras positioned at the front and back, providing comprehensive visual and depth perception for navigation and inspection tasks. An Inertial Measurement Unit (IMU) tracks movement and orientation, enhancing navigation accuracy in conjunction with a dual antenna RTK GPS module for precise location tracking and positioning, which is crucial for outdoor navigation. Wheel encoders monitor and control wheel movements, contributing to precise movement and positioning accuracy. These sensor simulations are configured to reflect the actual values and characteristics of the physical hardware, ensuring the robot's virtual model

behaves as realistically as possible.

7.4 Simulation Environment and Digital Twin

The project commenced with constructing a basic yet expansive simulation environment using Webots. This initial setup was primarily for early control system testing. As project requirements evolved, a more intricate simulation of a power substation was developed to provide a realistic setting for autonomy evaluations and visual odometry tests.

7.4.1 Development Process

The digital twin of the robot was meticulously crafted from the chassis and PTZ camera CAD models provided by the hardware team. These models were initially converted into STEP files using Solidworks 3D, which were then processed into .obj files with CAD Assistant. Subsequently, Blender 3D was used to convert the `.obj` files to `.dae` format, suitable for integrating into Webots and applying material properties, ensuring visual realism.

7.5 ROS 2 Integration and Control

7.5.1 Integration Techniques

The simulation extensively utilised the webots_ros2 package, facilitating a seamless integration between ROS 2 and the Webots simulation environment. This integration was critical for real-time control testing and ensuring that the digital twin's responses mirrored what would be expected in the physical robot.

7.5.2 Control Systems

The project leveraged the ros2_control package to implement a differential drive controller within the simulation. This setup was pivotal for mimicking the actual control dynamics of the robot, enabling the software team to develop and refine drive algorithms in a controlled virtual environment before hardware completion.

7.6 Operational Workflow and Documentation

7.6.1 Documentation and Guidelines

A comprehensive README file (Listing 3, refer to appendix §A- A.4.1 for the entire file) was created to guide users through the installation of the Webots simulator, the necessary system dependencies, and the operational workflow. This document is crucial for enabling new team

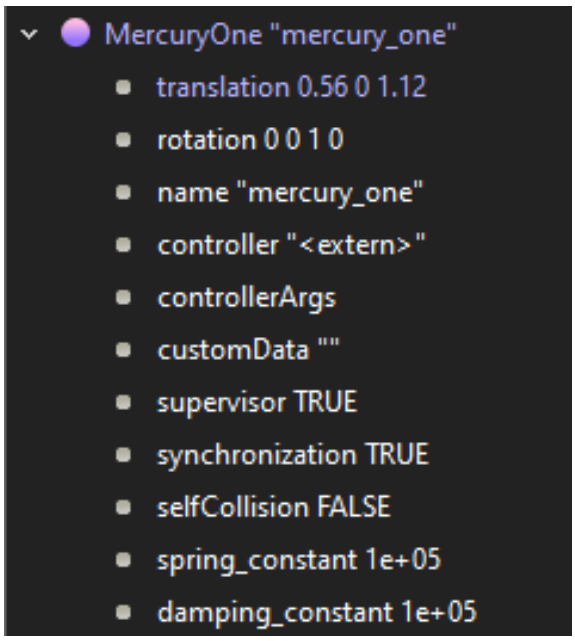


Figure 44: Controller configuration interface in Webots

members to get up to speed quickly and for ensuring consistent simulation practices across the development team.

7.6.2 Version Control and Collaboration

The project's development environment was managed through a version-controlled repository, facilitating smooth updates and integration of changes across multiple local development instances. This approach ensured that all team members had access to the latest simulation tools and resources, promoting an efficient and collaborative development process.

```
# Mercury One Robot Simulator README

## Introduction
This simulator package is designed to facilitate the development and testing of navigation and control algorithms for the Mercury One robot within a virtual environment.

## Prerequisites
Before you begin, ensure you have the following installed on your system:
- [ROS 2](https://docs.ros.org/en/humble/Installation.html) (Humble or newer recommended)
- [Webots](https://cyberbotics.com/doc/guide/installation-procedure) (R2023a or newer recommended)
- [Git](https://git-scm.com/downloads)
```

Code block 1: *Snippet from the README simulation user document.*

7.7 Visualisation

The project utilised detailed imagery and visual aids to enhance understanding and presentation of the robot's capabilities within the simulated environment. These visuals were instrumental in

demonstrating the robot's functionality to stakeholders and clients.

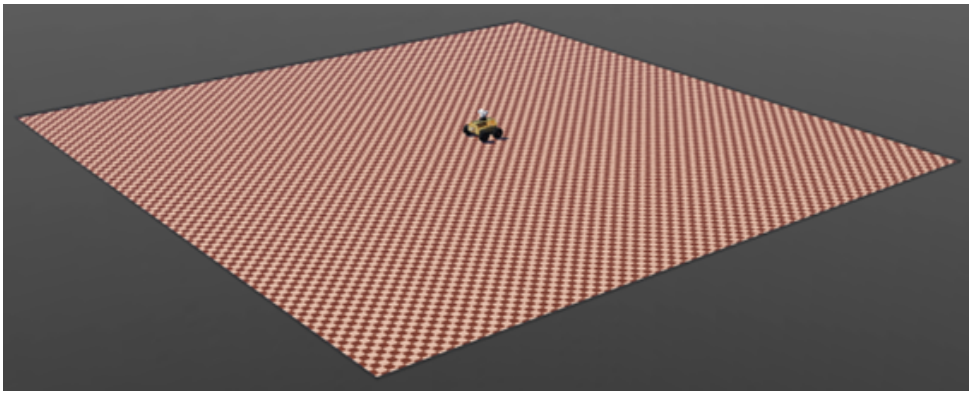


Figure 45: Initial large arena setup in Webots

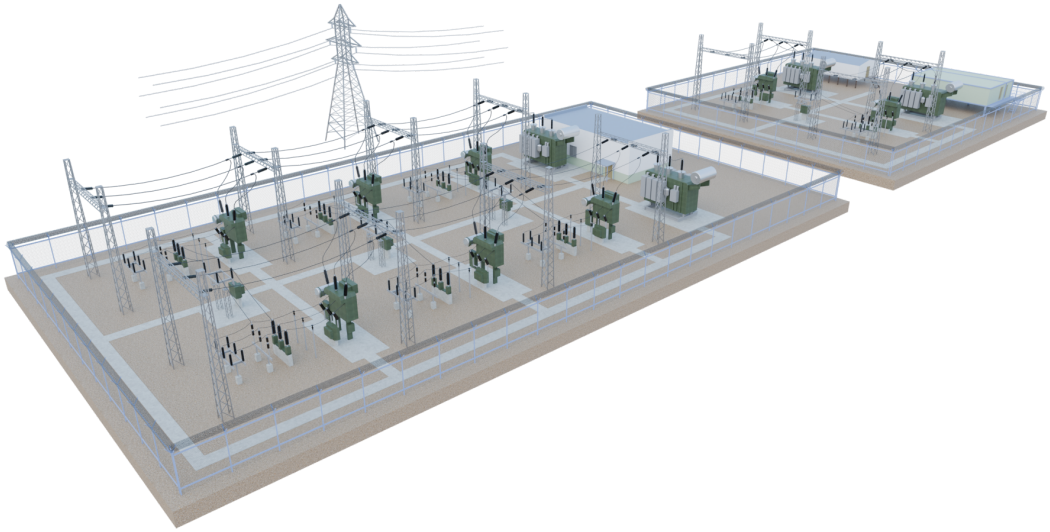


Figure 46: Advanced substation model for visual odometry tests

These sections and visuals underscore the meticulous approach to developing a robust simulation environment, integrating advanced control systems, maintaining effective workflow practices, and using imagery to convey complex technical concepts.

CHAPTER 8

Results & Discussion

This chapter presents the findings from three distinct case studies, each demonstrating the application of simulation and Digital Twins in robotic development. Each case study is structured to describe the findings, discuss the approach's effectiveness, and quantify the savings in terms of time and costs.

8.1 Case Study A - Line Painting Robot Simulation

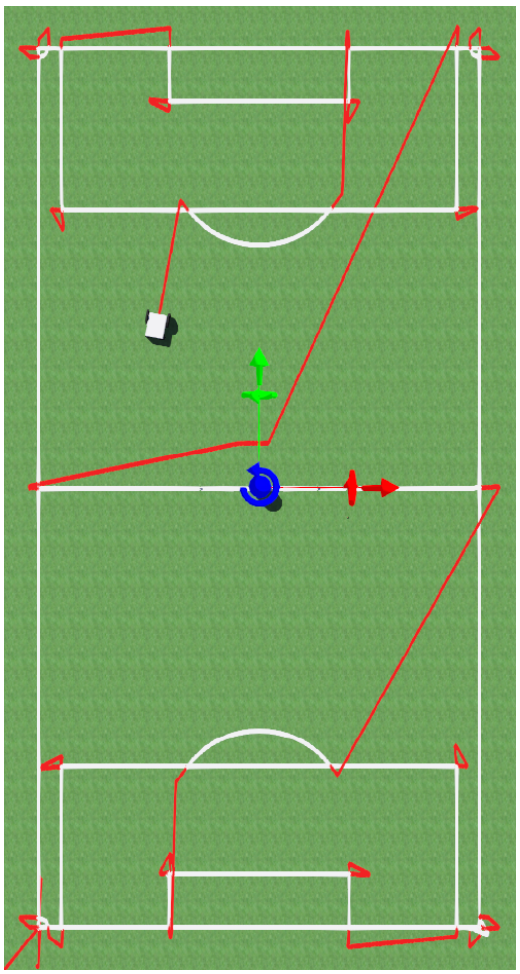
This section presents the findings from integrating a Digital Twin (DT) and simulation workflow to evaluate a pathfinding and waypoint-following system before the mechanical prototype was completed. The aim was to assess the feasibility and effectiveness of the robot's design and functionality using virtual testing environments.

8.1.1 Results

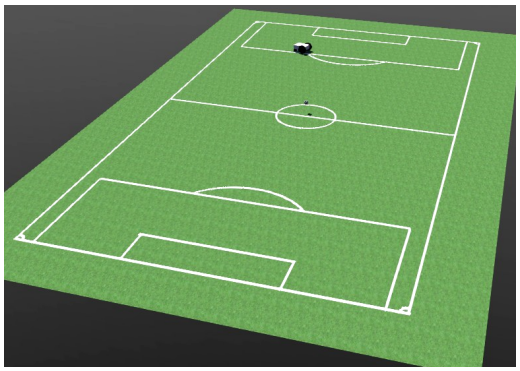
The DT and simulation were employed to evaluate the project's navigation and line painting logic more than a month before the physical prototype was completed. This early implementation allowed the team to fine-tune and optimise the system without the constraints and costs associated with physical testing.

8.1.2 Simulation Benefits

- **Early Validation of Navigation and Pathing:** The simulation facilitated early validation of the robot's navigation methodologies and pathing behaviours. By implementing simulation techniques before completing the mechanical prototype, the team could adjust and correct pathing errors and optimise routes without needing a physical model. This pre-emptive validation significantly reduced the risk of costly revisions during later stages of development.



(a) Simulated painting lines with transition lines (red).



(b) Simulated painting without transition lines.

Figure 47: A full robot painting of a rugby field (scaled down to save time) with (a) and without (b) transition lines.

- **time Savings:** The simulation facilitated early validation of the robot's navigation methodologies and pathing behaviours. By implementing simulation techniques before completing the mechanical prototype, the team could adjust and correct pathing errors and optimise routes without needing a physical model. This pre-emptive validation significantly reduced the risk of costly revisions during later stages of development.

- **Validation Against Physical Prototype:** The simulation's accuracy and effectiveness were confirmed against the completed physical prototype running the same algorithms, as illustrated in Figures 48 and 49.

8.1.3 Discussion

In this case study, using a Webots-based Digital Twin significantly accelerated the software development for a line painting robot. The simulation enabled the concurrent development of navigation algorithms and the programming of waypoints from path data before the completion

Figure 48: An aerial view of the real results of the robot following the same control and pathing scheme.

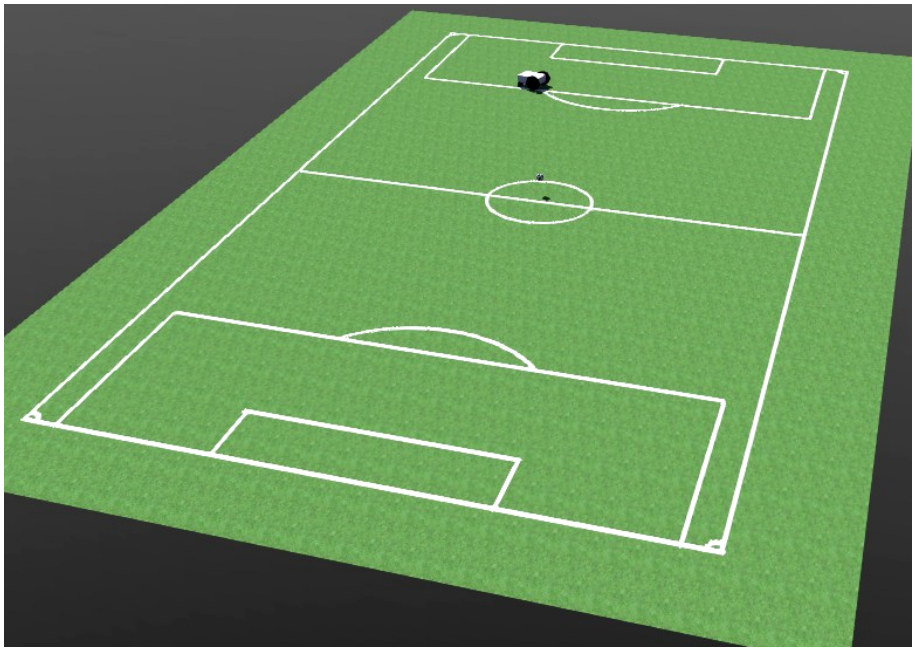


Figure 49: The final robot prototype painting.

of the physical robot. This approach optimised the development process and provided a platform for thorough path-following performance evaluations and identifying potential issues in a controlled environment. The robust software functionality, confirmed by comparative aerial footage of simulated and actual painting runs, underscores the tangible benefits of integrating digital twin technologies early in the development cycle. The financial and operational efficiencies realised highlight the value of simulations in reducing project timelines and costs while enhancing product reliability and performance.

8.2 Case Study B - Warehouse Navigation

This section presents the findings from implementing a Digital Twin (DT) and simulation workflow in developing an indoor camera-based robot localisation system, eliminating the need

for a physical prototype.

8.2.1 Results

The simulation's realistic camera setup enabled the successful application of the localisation algorithm to virtual recordings, mirroring the performance expected in real-world environments. Discrepancies in positioning indicated missing QR codes, illustrated by a gap in the red positions in Figure 50. The strategic placement of additional QR codes within the simulated environment effectively addressed these errors.

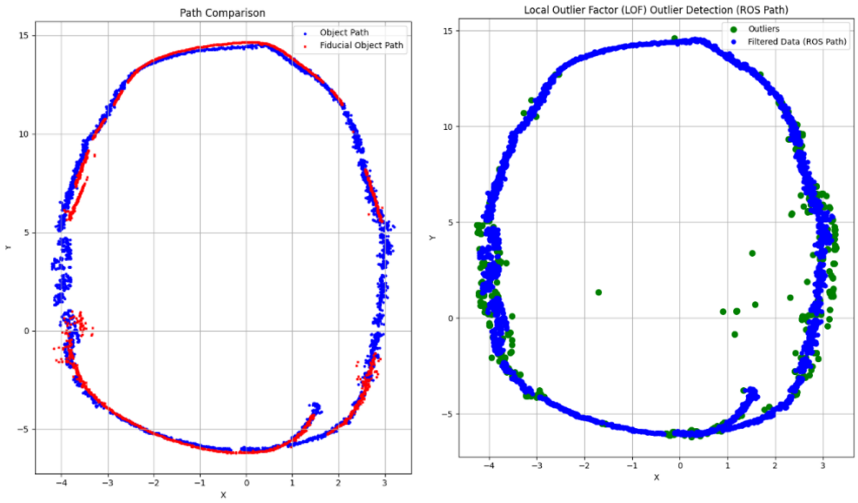


Figure 50: X Y (metres) error-plot of the known position against the calculated position using the localisation algorithm.

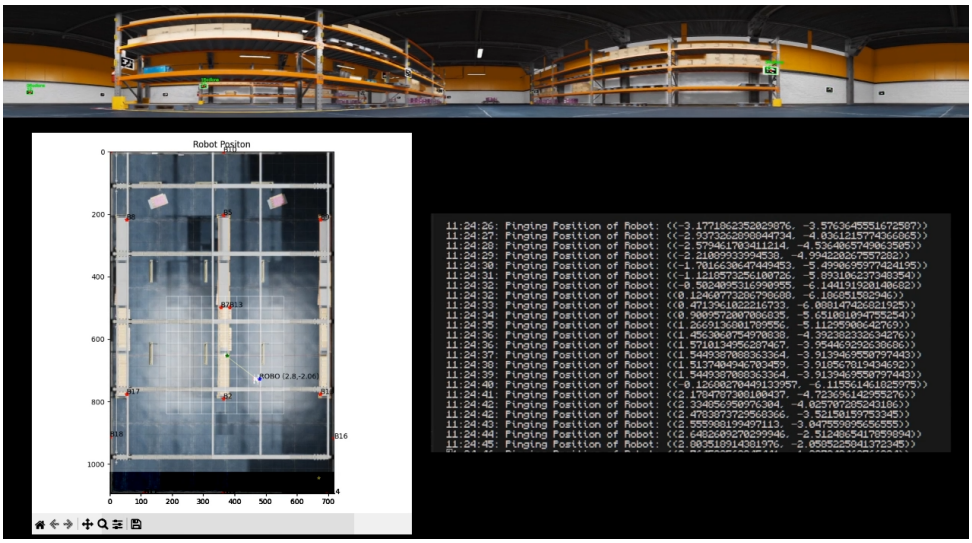


Figure 51: A comparative view of the localisation algorithm running using an overview with the blue ROBO point showing the calculated position of the robot. The top section shows the camera view and the highlighted QR codes as they are being read

Financially, the simulation avoided the cost of purchasing multiple hardware components, including a NZD 501.00 `Ricoh Theta V 4k 360 Spherical Camera` and additional

expenses associated with setting up physical robot platforms for testing, leading to substantial savings.

Table 8.1: Savings

Description	Cost
Total Cost without Sim	43,400.00
Total Cost with Sim	24,120.00
Proposed Savings ($)	19,280.00
Proposed Savings (%)	44.00%

8.2.2 Simulation Benefits

- **cost Savings:** The avoidance of hardware purchase and setup costs resulted in a total savings of NZD 9, 380.00, demonstrating a 44% reduction in costs compared to traditional development methods including hardware savings, environmental hireage, and other associated costs.

- **Time Efficiency:** By simulating the environment and utilising virtual cameras, one month of development time was saved, enabling faster progression to market and reducing overhead costs related to prolonged project durations.

8.2.3 Discussion

This case study exemplifies the effectiveness of high-fidelity simulations provided by Isaac Sim in refining camera and fiducial-based localisation algorithms within a controlled virtual setting. The detailed and responsive simulation environment significantly enhanced the accuracy and reliability of the algorithms by allowing for precise control over environmental variables and the immediate correction of detected deficiencies. The strategic placement of QR codes and the realistic simulation of camera functions facilitated comprehensive testing and optimisation of the localisation system without needing physical prototypes. This proactive approach minimised the necessity for costly physical testing and accelerated the development process, showcasing the profound impact of simulation technologies in streamlining development workflows in robotics. Overall, the integration of advanced simulation technologies proved instrumental in reducing both time and financial expenditures while simultaneously increasing the robustness and efficacy of the localisation system. The financial and operational efficiencies gained underscore the transformative potential of digital twin technologies in the robotics development landscape, particularly for complex applications such as indoor navigation.

8.3 Case Study C - Forestry Machine Simulation

This section presents results from integrating the digital twin and simulation technologies into developing the autonomous control system for the `Green Climber LV600`. The focus is on the savings in time, financial resources, and the reduction in logistical complexities, supplemented by a retrospective analysis of potential benefits had these technologies been employed from the project's outset.

8.3.1 Results

The development of the tool carrier autonomous control system originally spanned over several months and incurred substantial costs in logistics and expertise, totalling approximately NZD 204, 032. The project could have achieved significant financial and operational efficiencies by integrating digital twin and simulation technologies.

The findings from this case study suggest that with the adoption of simulations:

- **Potential Cost Savings:** The transition from physical field tests to simulated testing environments could have reduced the logistical and operational expenses by approximately 20.84%, or NZD 42, 512.00. This figure encompasses direct savings from decreased travel, setup, and fuel costs for numerous field tests and indirect savings from minimised equipment wear and less risk of damage.
- **Time Efficiency:** The project duration could have been reduced by four months by replacing extensive real-world testing with simulations. This change would have allowed for more rapid prototyping, iterations, and adjustments without the delays in organising and conducting field tests.

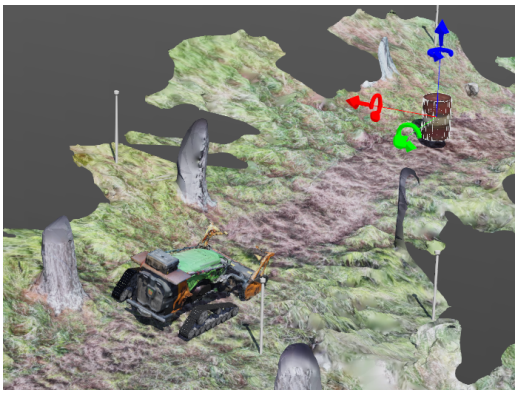


Figure 52: Simulated Tool Carrier following a goal marker (a barrel model)

8.3.2 Discussion

The retrospective analysis of the development process for the autonomous control system of the forestry tool carrier underscores the transformative potential of digital twin and simulation technologies. Initially, the project spanned several months, incurring substantial logistical and expertise costs, totalling approximately NZD 204,032.00 for early-stage control development alone. Integrating digital twin and simulation technologies could have significantly streamlined this development process. By replacing real-world field tests with high-fidelity simulations, the project could have avoided the logistical challenges and delays associated with physical testing. This change would have enabled more rapid prototyping, iterations, and adjustments, effectively reducing the project duration by several months. Moreover, adopting digital twin and simulation technologies would have streamlined the development process and reduced the logistical and operational expenses. The estimated savings from decreased needs for travel, setup, and fuel costs, alongside indirect savings from minimised equipment wear and tear, could have amounted to approximately NZD 42,511.84, representing a 20.84% cost reduction. Avoiding complex logistics in organising field tests in remote locations, often compounded by adverse weather conditions and other unforeseen issues, would have further streamlined the development process, making it more predictable and manageable. This case study highlights how the early integration of digital twins and simulation technologies could minimise costs and streamline development, significantly reducing the environmental and logistical impact of field testing. The strategic use of these technologies exemplifies their broad applicability and substantial benefits in developing complex robotic systems, particularly in challenging and unpredictable environments.

8.4 Case Study D - Inspection Robot

This section presents the findings from deploying digital twin and simulation technologies in developing an inspection robot to establish a workflow enabling software engineers to work on autonomy algorithms and software before the physical prototype was complete.

8.4.1 Results

The Foxglove robot visualisation tool displayed feeds from live and simulated robot operations. Foxglove allowed the team to monitor and verify the ROS 2 bridge connections and ensure that the behaviour of the simulated robot mirrored what was expected in the live robot, providing a crucial layer of validation for software functionality during the development phase.

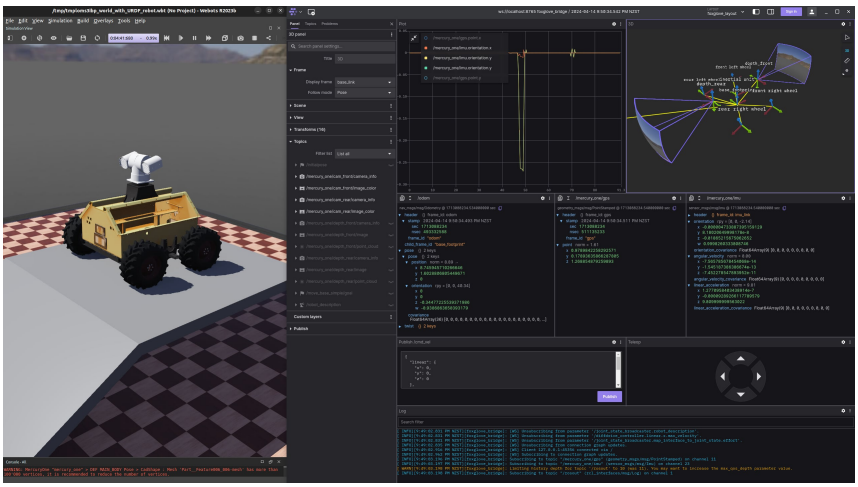


Figure 53: The Webots robot simulation on the left, with a comprehensive visualisation of the published topics and robot state on the right.

```
^linegreen^:~/mercury_ws/src/webots_interface
  /mercury_one/cam_front/camera_info
  /mercury_one/cam_front/image_color
  /mercury_one/cam_rear/camera_info
  /mercury_one/cam_rear/image_color
  /mercury_one/compass/bearing
  /mercury_one/compass/north_vector
  /mercury_one/depth_front/camera_info
  /mercury_one/depth_front/image
  /mercury_one/depth_front/point_cloud
  /mercury_one/depth_rear/camera_info
  /mercury_one/depth_rear/image
  /mercury_one/depth_rear/point_cloud
  /mercury_one/gps
  /mercury_one/gps/speed
  /mercury_one/gps/speed_vector
  /parameter_events
  /remove_urdf_robot
  /rosout
```

Code block 2: *ROS 2 topics published by the robot simulation*

Implementing a simulation environment accelerated the development process, enabling parallel workflows between the software and hardware teams. This approach was essential in reducing idle times and inefficiencies typically seen when software development must wait for hardware components to be constructed and integrated. The ability to test and refine the robot's navigation and operational protocols within a simulated environment ensured a higher level of system readiness and reliability upon deployment.

8.4.1.1 Financial Breakdown

- **Total Cost without Simulation (Sequential Development):** Initially projected at $43,400, assuming a traditional development approach where software development would have to wait for the mechanical prototype, stretching over 15 weeks.

- **Total Cost with Simulation (Parallel Development):** Reduced to $24,120 through the use

of simulation technologies, which shortened the overall project duration to 8 weeks and resulted in a cost reduction of 44%, or $19,280 in savings.

These cost savings were largely achieved through the following reductions in development time, mainly due to a reduction in "dead time" where software engineers would have been waiting for hardware completion:

- **Software Development:** Duration shortened from 11 weeks to 6 weeks, with costs reducing from $18,920 to $10,800.
- **UI Development:** Duration reduced from 9 to 4 weeks, lowering costs from $15,480 to $6,880.
- **Hardware Development:** Maintained at 5 weeks but with reduced daily costs, leading to a total of $3,000.
- **Simulation Development:** The setup and operation of the digital twin and simulation environment incurred $3,440.

8.4.2 Discussion

This case study underscores the strategic importance of digital twin and simulation technologies in facilitating early and efficient software development for complex robotic systems. By employing Foxglove for real-time visualisation, the project team verified the simulated robot's behaviour against expected outcomes, significantly enhancing the reliability of the software before hardware completion. The integration of simulation into the development process led to substantial cost and time savings and improved the development workflow by allowing for simultaneous progress on software and hardware components. This approach demonstrates the benefits of simulation technologies in robotics projects, especially in scenarios with restrictive timelines and synchronised software and hardware development. The project demonstrates how advanced simulation and visualisation tools can transform the robotics development landscape, promoting more efficient and effective methodologies in high-tech sectors.

8.5 Summary of Results

Integrating digital twin (DT) and simulation technologies across various robotics development projects has shown compelling outcomes. This analysis compares use cases, evaluates the effectiveness of these simulation-based approaches, and discusses inherent limitations and challenges.

8.5.1 Comparison of Use Cases

In Case Study A, the line painting robot benefited from early simulation to fine-tune navigation and pathing ahead of physical prototype completion. Case Study B applied simulation to enhance camera-based localisation algorithms for a warehouse navigation robot, mitigating the need for initial hardware setup. Case Study C used simulation to address logistical challenges and lower operational costs of remote field testing in the forestry industry. Despite differing applications—artistic precision, indoor navigation accuracy, and robust outdoor operation—all scenarios effectively utilised DT and simulations to anticipate and solve system-specific demands before real-world implementation.

8.5.2 Effectiveness of Simulation-Based Development

Across all case studies, simulation-based development proved highly effective in accelerating development timelines and reducing costs. The advantages include:

- **Time Savings:** Each project experienced reduced development cycles, with simulations allowing for parallel processes and early error detection.
- **Cost Efficiency:** Significant reductions were noted, particularly from the decreased need for physical materials, travel, and human resources.
- **Quality Improvement:** Simulations helped improve the overall system quality by allowing developers to explore more design variations and address potential issues in a risk-free environment.

8.5.3 Limitations and Challenges

While the benefits are substantial, simulation-based development is not without its challenges:

- **Fidelity and Accuracy:** Simulations depend heavily on the model's realism and the input data's accuracy. Discrepancies between simulated environments and real-world conditions can lead to oversights in system performance.
- **Resource Intensity:** Developing high-fidelity simulations requires significant computational resources and expertise, which can be a barrier for smaller teams or projects with limited budgets.
- **Dependency on Technology:** There is a risk of over-reliance on simulation results, potentially neglecting practical tests and evaluations crucial for final system validations.

8.5.4 Summary of Results: Cost-Savings Through Simulation

The results section of this thesis underscores the significant cost and time savings achieved through the integration of simulation and Digital Twin technologies in the robot development process:

- **Case A - Line Painting:** Utilising a simulation saved $8,080.00, a 58% reduction in costs.
- **Case B - Indoor Navigation:** Simulation avoided $9,380.00 in costs, a 44% reduction, by eliminating the need for hardware for the proof of concept.
- **Case C - Forestry:** By replacing field tests with simulations, the project saved $42,511.84, a 20.84% cost reduction, decreasing the timeline by four months.
- **Case D - Inspection Robot:** Simulations facilitated parallel development, saving $19,280.00 and reducing the development period by 56% over two months.

These case studies demonstrate the powerful role of digital twins and simulations in the iterative process of robotics development, highlighting their transformative potential and the need for careful implementation to navigate their limitations. This approach enhances efficiency and fosters a more sustainable approach to robotics development, optimising timelines, reducing costs, and improving overall project outcomes in robotic engineering.

CHAPTER 9

Conclusions

The research presented in this thesis offers a comprehensive demonstration of the transformative impact of Digital Twins (DTs) and advanced simulations in robotic development. By integrating DTs with the latest GPU technologies, significant advancements over traditional field testing methods have been achieved, leading to enhanced development efficiency, improved precision and quality of robotic systems, and substantial cost savings. Through detailed case studies ranging from line painting robots to forestry machine simulations, the integration of these technologies has proven to effectively streamline development processes, reduce costs, and improve efficiency, significantly supporting the thesis statement and advocating for their adoption to address inefficiencies inherent in traditional development cycles.

9.1 Key Findings

1. **Validation of Thesis Claims**: The case studies confirm that utilizing DTs and simulations can revolutionize robotic system development and deployment by allowing parallel software and hardware advancements, thus accelerating development phases and diminishing the financial and logistical burdens typically associated with robotics development.

2. **Broader Impacts and Efficiency Gains**: This research underscores substantial improvements in timelines and cost efficiencies, highlighting DT and simulations as pivotal tools for enhancing the precision and efficacy of robotic systems across various applications. Integrating these technologies mitigates resource expenditure and promotes a more sustainable approach to robotics development, significantly reducing environmental impact.

3. **Quality Improvement and Cost-Effectiveness**: The findings from various case studies indicate significant reductions in development time and cost, with examples such as the line painting robot project achieving a 30% reduction in time and NZD 20,000 in cost savings. These benefits underscore the economic advantages and improved product reliability of DTs and simulations.

9.2 Future Research and Recommendations

The research paves the way for several promising directions for future work:

- **Enhancing Simulation Realism**: Future studies could focus on advancing simulation realism to encompass more complex and variable environmental factors. This enhancement would aid in developing robots capable of adapting to dynamic real-world conditions, thus increasing their operational robustness.
- **Integrating Advanced Technologies**: There is significant potential for integrating AI and machine learning to refine DTs' accuracy and operational efficiency. These technologies could create more adaptive and responsive robotic systems, further enhancing the capabilities of DTs in robotic development.
- **Expanding Application Areas**: Continuing this research into more complex robotic systems, including collaborative robotics and those operating in dynamic or unpredictable settings, could further validate and expand the applicability of DTs and simulations in various industrial sectors.

By addressing these recommendations, future research can continue to build on the foundational work presented in this thesis, further advancing the field of robotics towards more innovative, efficient, and sustainable development practices.

Bibliography

[1] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can Robot Navigation Bugs Be Found in Simulation? An Exploratory Study," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Prague, Czech Republic: IEEE, Jul. 2017, pp. 150–159 (cited on page 1).

[2] A. Afzal, D. S. Katz, C. L. Goues, and C. S. Timperley. "A Study on the Challenges of Using Robotics Simulators for Testing." arXiv: 2004 . 07368 [cs]. (Apr. 15, 2020), [Online]. Available: http://arxiv.org/abs/2004.07368 (visited on 04/10/2024), preprint (cited on page 1).

[3] V. Makoviychuk, L. Wawrzyniak, Y. Guo, *et al.* "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning." version 2. (2021), [Online]. Available: https://arxiv.org/abs/2108.10470 (visited on 11/29/2023), preprint (cited on pages 1, 4).

[4] C. S. Timperley, A. Afzal, D. S. Katz, J. M. Hernandez, and C. Le Goues, "Crashing Simulated Planes is Cheap: Can Simulation Detect Robotics Bugs Early?" In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, Vasteras: IEEE, Apr. 2018, pp. 331–342 (cited on page 1).

[5] S. Birrell, J. Hughes, J. Y. Cai, and F. Iida, "A field-tested robotic harvesting system for iceberg lettuce," *Journal of Field Robotics*, vol. 37, no. 2, pp. 225–245, Mar. 2020 (cited on page 1).

[6] K. Rosen, "The History of Simulation," in *The Comprehensive Textbook of Healthcare Simulation*, A. I. Levine, S. DeMaria, A. D. Schwartz, and A. J. Sim, Eds., New York, NY: Springer New York, 2013, pp. 5–49 (cited on page 4).

[7] E. Datteri and V. Schiaffonati, "Robotic Simulations, Simulations of Robots," *Minds and Machines*, vol. 29, no. 1, pp. 109–125, Mar. 2019 (cited on page 4).

[8] J. Banks, *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice* (A Wiley-Interscience Publication). New York Weinheim: Wiley, 1998, 849 pp. (cited on pages 4, 16).

[9] J. Hollingum, "Robot simulation system comes to Britain," *Industrial Robot: An International Journal*, vol. 17, no. 4, pp. 181–183, Jan. 1, 1990 (cited on page 4).

[10] Shan-ping Li and Yu-Liang Xu, "A General-Purpose Robots Simulation System on IBM-PC," in *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, Beijing, China: IEEE, 1988, pp. 1218–1221 (cited on page 4).

[11] D. Wloka, "Simulation of Robot Factories Using Robsim," *IFAC Proceedings Volumes*, vol. 20, no. 5, pp. 313–319, Jul. 1987 (cited on page 4).

[12] V. Jovanovic, M. Kuzlu, U. Cali, *et al.*, "Digital Twin in Industry 4.0 and Beyond Applications," in *Digital Twin Driven Intelligent Systems and Emerging Metaverse*, E. Karaarslan, Ö. Aydin, Ü. Cali, and M. Challenger, Eds., Singapore: Springer Nature Singapore, 2023, pp. 155–174 (cited on pages 4, 8, 9, 18, 19).

[13] T. Leal Ghezzi and O. Campos Corleta, "30 Years of Robotic Surgery," *World Journal of Surgery*, vol. 40, no. 10, pp. 2550–2557, Oct. 2016 (cited on pages 4, 18).

[14] R. Goel and P. Gupta, "Robotics and Industry 4.0," in *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*, ser. Advances in Science, Technology & Innovation, A. Nayyar and A. Kumar, Eds., Cham: Springer International Publishing, 2020, pp. 157–169 (cited on pages 4, 8, 18).

[15] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning." arXiv: 2109.11978 [cs]. (Aug. 19, 2022), [Online]. Available: http://arxiv.org/abs/2109.11978 (visited on 11/30/2023), preprint (cited on pages 4, 20).

[16] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '11, New York, NY, USA: Association for Computing Machinery, May 25, 2011, pp. 1–8 (cited on pages 4, 5).

[17] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti, "Introductory Survey to Open-Source Mobile Robot Simulation Software," in *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*, Sao Bernardo do Campo, Brazil: IEEE, Oct. 2010, pp. 150–155 (cited on pages 4, 5).

[18] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *2011 Proceedings of IEEE Southeastcon*, Nashville, TN, USA: IEEE, Mar. 2011, pp. 243–249 (cited on pages 4, 5).

[19] S. Ivaldi, V. Padois, and F. Nori. "Tools for dynamics simulation of robots: A survey based on user feedback." arXiv: 1402.7050 [cs]. (Feb. 27, 2014), [Online]. Available: http://arxiv.org/abs/1402.7050 (visited on 06/25/2023), preprint (cited on pages 4–6).

[20] A. Konrad, *Simulation of Mobile Robots with Unity and ROS : A Case-Study and a Comparison with Gazebo*. 2019 (cited on pages 4, 5, 8).

[21] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA: IEEE, May 2015, pp. 4397–4404 (cited on pages 4–6).

[22] H. Choi, C. Crump, C. Duriez, *et al.*, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, e1907856118, Jan. 5, 2021 (cited on page 5).

[23] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 24–32, Mar. 1996 (cited on page 5).

[24] MathWorks. "Robot Simulation - MATLAB & Simulink." (2023), [Online]. Available: https://www.mathworks.com/help/robotics/robot-simulation.html (visited on 11/30/2023) (cited on page 5).

[25] J. D. González, J. H. Escobar, H. Sánchez, J. D. L. Hoz, and J. R. Beltrán, "2D and 3D virtual interactive laboratories of physics on Unity platform," *Journal of Physics: Conference Series*, vol. 935, p. 012 069, Dec. 2017 (cited on page 5).

[26] O. Ganoni and R. Mukundan, "A Framework for Visually Realistic Multi-robot Simulation in Natural Environment," version 1, 2017 (cited on page 6).

[27] A. Saglam and Y. Papelis, "Scalability of Sensor Simulation in ROS-Gazebo Platform with and without Using GPU," in *2020 Spring Simulation Conference (SpringSim)*, May 2020, pp. 1–11 (cited on page 6).

[28] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "RViz: A toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, Oct. 2015 (cited on page 6).

[29] N. Hongpan, G. Yong, and H. Zhongming, "Application research of PhysX engine in virtual environment," in *2010 International Conference on Audio, Language and Image Processing*, Shanghai, China: IEEE, Nov. 2010, pp. 587–591 (cited on pages 6, 7).

[30] Erwin Coumans. "Bullet Real-Time Physics Simulation | Home of Bullet and PyBullet: Physics simulation for games, visual effects, robotics and reinforcement learning." (Mar. 21, 2022), [Online]. Available: https://pybullet.org/wordpress/ (visited on 06/25/2023) (cited on page 6).

[31] J. Lee, M. X. Grey, S. Ha, *et al.*, "DART: Dynamic Animation and Robotics Toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, Feb. 5, 2018 (cited on page 6).

[32] Cyberbotics, *Cyberbotics: Robotics simulation with Webots*, version 2023a, Cyberbotics, 1998 (cited on pages 6, 7).

[33] J. P. Resop, L. Lehmann, and W. C. Hession, "Drone Laser Scanning for Modeling Riverscape Topography and Vegetation: Comparison with Traditional Aerial Lidar," *Drones*, vol. 3, no. 2, p. 35, 2 Jun. 2019 (cited on pages 6, 11).

[34] R. Wood and M. E. Mohammadi, "LiDAR Scanning with Supplementary UAV Captured Images for Structural Inspections," Jan. 1, 2015 (cited on pages 6, 11).

[35] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 5026–5033 (cited on page 6).

[36] P. Lindemann, "The Gilbert-Johnson-Keerthi Distance Algorithm," 2009 (cited on page 7).

[37] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sendai, Japan: IEEE, 2004, pp. 2149–2154 (cited on page 7).

[38] M. Quigley, B. Gerkey, K. Conley, *et al.*, "ROS: An open-source Robot Operating System," Stanford University, 2009, p. 6 (cited on page 7).

[39] Open Robotics, *Gazebo*, version Garden, 2023 (cited on page 7).

[40] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," version 1, 2017 (cited on pages 7, 12).

[41] FS Studio, *ZeroSim - Robotic Simulation Services*, 2023 (cited on pages 7, 8).

[42] Unity Technologies, *Robotics Simulation | Unity*, Unity Technologies, 2023 (cited on pages 7, 8).

[43] Nvidia, *Isaac Sim - Robotics Simulation and Synthetic Data Generation | NVIDIA Developer*, version 2022.2, 2023 (cited on pages 8, 19).

[44] Z. Zhou, J. Song, X. Xie, *et al.* "Towards Building AI-CPS with NVIDIA Isaac Sim: An Industrial Benchmark and Case Study for Robotics Manipulation." version 1. (2023), [Online]. Available: https://arxiv.org/abs/2308.00055 (visited on 11/29/2023), preprint (cited on page 8).

[45] W. A. Mattingly, D.-j. Chang, R. Paris, N. Smith, J. Blevins, and M. Ouyang, "Robot design using Unity for computer games and robotic simulations," in *2012 17th International Conference on Computer Games (CGAMES)*, Jul. 2012, pp. 56–59 (cited on page 8).

[46] F. P. Audonnet, A. Hamilton, and G. Aragon-Camarasa, "A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, Nov. 2022, pp. 755–762 (cited on page 8).

[47] J. Platt and K. Ricks, "Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 4, p. 80, Dec. 20, 2022 (cited on page 8).

[48] C. Sevastopoulos and S. Konstantopoulos, "A Simulated Environment for Traversability Estimation Experiments in Field Robotics Applications," in *The 14th PErvasive Technologies Related to Assistive Environments Conference*, ser. PETRA 2021, New York, NY, USA: Association for Computing Machinery, Jun. 29, 2021, pp. 256–257 (cited on page 8).

[49] B. D. Allen, "Digital Twins and Living Models at NASA," Nov. 3, 2021 (cited on page 8).

[50] R. K. Phanden, P. Sharma, and A. Dubey, "A review on simulation in digital twin for aerospace, manufacturing and robotics," *Materials Today: Proceedings*, vol. 38, pp. 174–178, 2021 (cited on pages 8, 9).

[51] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference
20th AIAA/ASME/AHS Adaptive Structures Conference
14th AIAA*, Honolulu, Hawaii: American Institute of Aeronautics and Astronautics, Apr. 23, 2012 (cited on page 8).

[52] X. Liu, H. Gan, Y. Luo, Y. Chen, and L. Gao, "Digital-Twin-Based Real-Time Optimization for a Fractional Order Controller for Industrial Robots," *Fractal and Fractional*, vol. 7, no. 2, p. 167, 2 Feb. 2023 (cited on pages 8–10).

[53] Z. Huang, Y. Shen, J. Li, M. Fey, and C. Brecher, "A Survey on AI-Driven Digital Twins in Industry 4.0: Smart Manufacturing and Advanced Robotics," *Sensors*, vol. 21, no. 19, p. 6340, Sep. 23, 2021 (cited on pages 8–10).

[54] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, May 1, 2020 (cited on page 8).

[55] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, vol. 51, no. 11, pp. 1016–1022, Jan. 1, 2018 (cited on page 8).

[56] N. Frick and J. Metternich, "The Digital Value Stream Twin," *Systems*, vol. 10, no. 4, p. 102, Jul. 17, 2022 (cited on page 9).

[57] E. A. Avila, D. P. Chapa, I. D. Arenas, and C. V. Hurtado, "A Digital Twin implementation for Mobile and collaborative robot scenarios for teaching robotics based on Robot Operating System," in *2022 IEEE Global Engineering Education Conference (EDUCON)*, Mar. 2022, pp. 559–564 (cited on page 9).

[58] B. Ahmad, M. U. Sattar, H. W. Khan, *et al.*, "Intelligent Digital Twin to make Robot Learn the Assembly process through Deep Learning | Lahore Garrison University Research Journal of Computer Science and Information Technology," Oct. 20, 2021 (cited on pages 9, 10).

[59] S. Baidya, S. K. Das, M. H. Uddin, C. Kosek, and C. Summers, "Digital Twin in Safety-Critical Robotics Applications: Opportunities and Challenges," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, Nov. 2022, pp. 101–107 (cited on pages 9, 10).

[60] H. Kivrak, P. D. E. Baniqued, S. Watson, and B. Lennox, "An Investigation of the Network Characteristics and Requirements of 3D Environmental Digital Twins for Inspection Robots," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2022, pp. 596–600 (cited on pages 9, 10).

[61] T. Erol, A. F. Mendi, and D. Dogan, "The Digital Twin Revolution in Healthcare," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Istanbul, Turkey: IEEE, Oct. 22, 2020, pp. 1–7 (cited on page 9).

[62] R. M. Viglialoro, S. Condino, G. Turini, M. Carbone, V. Ferrari, and M. Gesi, "Augmented Reality, Mixed Reality, and Hybrid Approach in Healthcare Simulation: A Systematic Review," *Applied Sciences*, vol. 11, no. 5, p. 2338, 5 Jan. 2021 (cited on pages 9, 18).

[63] T. Kaarlela, S. Pieska, and T. Pitkaaho, "Digital Twin and Virtual Reality for Safety Training," in *2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, Mariehamn, Finland: IEEE, Sep. 23, 2020, pp. 000 115–000 120 (cited on page 9).

[64] J. O. Oyekan, W. Hutabarat, A. Tiwari, *et al.*, "The effectiveness of virtual environments in developing collaborative strategies between industrial robots and humans," *Robotics and Computer-Integrated Manufacturing*, vol. 55, pp. 41–54, Feb. 1, 2019 (cited on pages 9, 10).

[65] T. Kaarlela, P. Padrao, T. Pitkäaho, S. Pieskä, and L. Bobadilla, "Digital Twins Utilizing XR-Technology as Robotic Training Tools," *Machines*, vol. 11, no. 1, p. 13, 1 Jan. 2023 (cited on pages 9–11).

[66] L. Damiani, M. Demartini, P. Giribone, M. Maggiani, and R. Revetria, "Simulation and Digital Twin Based Design of a Production Line: A Case Study," *Hong Kong*, 2018 (cited on page 10).

[67] M. Rojas, G. Hermosilla, D. Yunge, and G. Farias, "An Easy to Use Deep Reinforcement Learning Library for AI Mobile Robots in Isaac Sim," *Applied Sciences*, vol. 12, no. 17, p. 8429, Aug. 24, 2022 (cited on pages 10, 20).

[68] M. Kulawiak and Z. Lubniewski, "Improving the Accuracy of Automatic Reconstruction of 3D Complex Buildings Models from Airborne Lidar Point Clouds," *Remote Sensing*, vol. 12, no. 10, p. 1643, 10 Jan. 2020 (cited on page 11).

[69] M. Pepe and D. Costantino, "Techniques, Tools, Platforms and Algorithms in Close Range Photogrammetry in Building 3D Model and 2D Representation of Objects and Complex Architectures," *Computer-Aided Design and Applications*, vol. 18, pp. 42–65, May 22, 2020 (cited on pages 11, 14).

[70] V. Stojaković, "Terrestrial photogrammetry and application to modeling architectural objects," *Facta Universitatis - series : Architecture and Civil Engineering*, vol. 6, Jan. 1, 2008 (cited on pages 11, 14).

[71] S. Gonizzi Barsanti, F. Remondino, and D. Visintini, "3D SURVEYING AND MODELING OF ARCHAEOLOGICAL SITES – SOME CRITICAL ISSUES –," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5/W1, pp. 145–150, Jul. 31, 2013 (cited on page 11).

[72] D. F. Tandy, C. Coleman, J. Colborn, T. Hoover, and J. Bae, "Benefits and Methodology for Dimensioning a Vehicle Using a 3D Scanner for Accident Reconstruction Purposes," presented at the SAE 2012 World Congress & Exhibition, Apr. 16, 2012, pp. 2012-01–0617 (cited on pages 11, 12).

[73] F. Bruno, S. Bruno, G. De Sensi, M.-L. Luchi, S. Mancuso, and M. Muzzupappa, "From 3D reconstruction to virtual reality: A complete methodology for digital archaeological exhibition," *Journal of Cultural Heritage*, vol. 11, no. 1, pp. 42–49, Jan. 2010 (cited on page 11).

[74] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A Survey on LiDAR Scanning Mechanisms," *Electronics*, vol. 9, no. 5, p. 741, 5 May 2020 (cited on page 11).

[75] U. Wandinger, "Introduction to Lidar," in *Lidar*, C. Weitkamp, Ed., vol. 102, New York: Springer-Verlag, 2005, pp. 1–18 (cited on page 11).

[76] P. Babin, P. Dandurand, V. Kubelka, P. Giguère, and F. Pomerleau, "Large-scale 3D Mapping of Subarctic Forests," version 2, 2019 (cited on page 11).

[77] E. Hyyppä, J. Hyyppä, T. Hakala, *et al.*, "Under-canopy UAV laser scanning for accurate forest field measurements," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 164, pp. 41–60, Jun. 1, 2020 (cited on page 11).

[78] M. Olsen, G. Roe, C. Glennie, *et al.*, *Guidelines for the Use of Mobile LIDAR in Transportation Applications*. Jan. 1, 2013 (cited on page 11).

[79] R. Zlot and M. Bosse, "Efficient Large-Scale 3D Mobile Mapping and Surface Reconstruction of an Underground Mine," in *Field and Service Robotics*, K. Yoshida and S. Tadokoro, Eds., vol. 92, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 479–493 (cited on page 11).

[80] J. Cardenal, J. L. Perez-Garcia, E. Mata, *et al.*, "Photogrammetric and LiDAR documentation of the royal chapel (Cathedral-mosque of Cordoba, Spain)," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX-B5, pp. 541–546, Jul. 30, 2012 (cited on pages 11, 14).

[81] C. Wang, C. Wen, Y. Dai, S. Yu, and M. Liu, "Urban 3D modeling with mobile laser scanning: A review," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 3, pp. 175–212, Jun. 2020 (cited on page 11).

[82] C. Mallet and F. Bretar, "Full-waveform topographic lidar: State-of-the-art," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 1, pp. 1–16, Jan. 2009 (cited on page 11).

[83] L. Wallace, A. Lucieer, C. Watson, and D. Turner, "Development of a UAV-LiDAR System with Application to Forest Inventory," *Remote Sensing*, vol. 4, pp. 1519–1543, Dec. 1, 2012 (cited on page 11).

[84] R. Morelle and A. Francis, "Titanic: First ever full-sized scans reveal wreck as never seen before," May 17, 2023 (cited on page 12).

[85] J. D. Gardiner, J. Behnsen, and C. A. Brassey, "Alpha shapes: Determining 3D shape complexity across morphologically diverse structures," *BMC Evolutionary Biology*, vol. 18, no. 1, p. 184, Dec. 5, 2018 (cited on page 12).

[86] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," 2006 (cited on pages 12, 14).

[87] W. Lin and S. Lee, "Visual Saliency and Quality Evaluation for 3D Point Clouds and Meshes: An Overview," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022 (cited on page 12).

[88] Azad Balabanian, director, *iPhone LIDAR vs PC Photogrammetry*, Nov. 8, 2021 (cited on page 12).

[89] S. Miller, R. Gillihan, and E. Helms, "A Comparison of Mobile LiDAR Capture and Established Ground-Based 3D. . .," Mar. 29, 2022 (cited on page 13).

[90] F. Remondino, "Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning," *Remote Sensing*, vol. 3, no. 6, pp. 1104–1138, May 30, 2011 (cited on pages 13, 14).

[91] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, *Close-Range Photogrammetry and 3d Imaging*, 3rd edition. Berlin: Walter de Gruyter GmbH, 2020, 822 pp. (cited on pages 13, 14).

[92] N. M. Zahari, M. A. A. Karim, F. Nurhikmah, N. A. Aziz, M. H. Zawawi, and D. Mohamad, "Review of Unmanned Aerial Vehicle Photogrammetry for Aerial Mapping Applications," in *ICCOEE2020*, B. S. Mohammed, N. Shafiq, S. Rahman M. Kutty, H. Mohamad, and A.-L. Balogun, Eds., ser. Lecture Notes in Civil Engineering, Singapore: Springer, 2021, pp. 669–676 (cited on page 14).

[93] S. I. Deliry and U. Avdan, "Accuracy of Unmanned Aerial Systems Photogrammetry and Structure from Motion in Surveying and Mapping: A Review," *Journal of the Indian Society of Remote Sensing*, vol. 49, no. 8, pp. 1997–2017, Aug. 1, 2021 (cited on page 14).

[94] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, Jun. 2014 (cited on page 14).

[95] William Faucher, director, *RealityCapture to UE5 - Workflow Tutorial*, Dec. 19, 2021 (cited on page 14).

[96] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the World from Internet Photo Collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, Nov. 2008 (cited on page 14).

[97] L. Perumal, "New approaches for Delaunay triangulation and optimisation," *Heliyon*, vol. 5, no. 8, e02319, Aug. 1, 2019 (cited on page 14).

[98] L. Galantucci, G. Percoco, and F. E. R. R. A. N. D. E. S. R, "Accuracy Issues of Digital Photogrammetry for 3D Digitization of Industrial Products," *REVUE INTERNATIONALE D'INGÉNIERIE NUMÉRIQUE*, vol. 2, pp. 29–40, Jan. 1, 2006 (cited on page 14).

[99] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." arXiv: 2003.08934 [cs]. (Aug. 3, 2020), [Online]. Available: http://arxiv.org/abs/2003.08934 (visited on 05/09/2023), preprint (cited on page 15).

[100] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections." version 3. arXiv: 2008.02268 [cs]. (Jan. 6, 2021), [Online]. Available: http://arxiv.org/abs/2008.02268 (visited on 05/09/2023), preprint (cited on pages 15, 16).

[101] M.-J. Rakotosaona, F. Manhardt, D. M. Arroyo, M. Niemeyer, A. Kundu, and F. Tombari. "NeRFMeshing: Distilling Neural Radiance Fields into Geometrically-Accurate 3D Meshes." arXiv: 2303.09431 [cs]. (Mar. 16, 2023), [Online]. Available: http://arxiv.org/abs/2303.09431 (visited on 05/10/2023), preprint (cited on pages 15, 16).

[102] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. "D-NeRF: Neural Radiance Fields for Dynamic Scenes." arXiv: 2011.13961 [cs]. (Nov. 27, 2020), [Online]. Available: http://arxiv.org/abs/2011.13961 (visited on 05/09/2023), preprint (cited on page 16).

[103] H. Turki, D. Ramanan, and M. Satyanarayanan. "Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs." arXiv: 2112.10703 [cs]. (Mar. 28, 2022), [Online]. Available: http://arxiv.org/abs/2112.10703 (visited on 05/09/2023), preprint (cited on page 16).

[104] L. Xu, Y. Xiangli, S. Peng, *et al.* "Grid-guided Neural Radiance Fields for Large Urban Scenes." arXiv: 2303.14001 [cs]. (Mar. 24, 2023), [Online]. Available: http://arxiv.org/abs/2303.14001 (visited on 05/09/2023), preprint (cited on page 16).

[105] C. Li, S. Li, Y. Zhao, W. Zhu, and Y. Lin. "RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering." arXiv: 2212.01120 [cs]. (Dec. 2, 2022), [Online]. Available: http://arxiv.org/abs/2212.01120 (visited on 05/09/2023), preprint (cited on page 16).

[106] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. "PlenOctrees for Real-time Rendering of Neural Radiance Fields." arXiv: 2103.14024 [cs]. (Aug. 17, 2021), [Online]. Available: http://arxiv.org/abs/2103.14024 (visited on 05/09/2023), preprint (cited on page 16).

[107] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang. "S-NeRF: Neural Radiance Fields for Street Views." arXiv: 2303.00749 [cs]. (Mar. 1, 2023), [Online]. Available: http://arxiv.org/abs/2303.00749 (visited on 05/10/2023), preprint (cited on page 16).

[108] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural Volumes: Learning Dynamic Renderable Volumes from Images," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–14, Aug. 31, 2019. arXiv: 1906.07751 [cs] (cited on page 16).

[109] Nvidia, director, *BMW Group Celebrates Opening the World's First Virtual Factory in NVIDIA Omniverse*, Mar. 22, 2023 (cited on pages 16, 19).

[110] Daps Magic, director, *FIRST APPEARANCE: BDX Droids at Star Wars: Galaxy's Edge | Season of the Force | Disneyland Resort*, Apr. 6, 2024 (cited on page 17).

[111] A. Serifi, E. Knoop, C. Schumacher, N. Kumar, M. Gross, and M. Bächer, "Transformer-Based Neural Augmentation of Robot Simulation Representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3748–3755, Jun. 2023 (cited on page 17).

[112] NVIDIA, director, *Amazon Robotics Deploys First Fully Autonomous Robot With NVIDIA Isaac Sim*, Mar. 22, 2023 (cited on page 17).

[113] R. R. Shamshiri, I. A. Hameed, L. Pitonakova, *et al.*, "Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison," *15-31*, 2018 (cited on page 18).

[114] A. Gavazzi, A. N. Bahsoun, W. Van Haute, *et al.*, "Face, content and construct validity of a virtual reality simulator for robotic surgery (SEP Robot)," *The Annals of The Royal College of Surgeons of England*, vol. 93, no. 2, pp. 152–156, Mar. 2011 (cited on page 18).

[115] A. Bauer, D. Wollherr, and M. Buss, "HUMAN–ROBOT COLLABORATION: A SURVEY," *International Journal of Humanoid Robotics*, vol. 05, no. 01, pp. 47–66, Mar. 2008 (cited on page 19).

[116] Universal Robots. "Toyota Motor Hokkaido, Inc | Cobot Case Stories." (2020), [Online]. Available: https://www.universal-robots.com/case-stories/toyota-motor-hokkaido-inc/,%20https://www.universal-robots.com/case-stories/toyota-motor-hokkaido-inc/ (visited on 12/05/2023) (cited on pages 19, 20).

[117] A. Plasencia, Y. Shichkina, I. Suárez, and Z. Ruiz, "Open Source Robotic Simulators Platforms for Teaching Deep Reinforcement Learning Algorithms," *Procedia Computer Science*, vol. 150, pp. 162–170, 2019 (cited on page 20).

APPENDIX A

Appendices

A.1 Case A - Line Marking

A.1.1 Financial Breakdown - Additional Tables

Table A.1: Known Costs

Cost Factor	Value	Units
Engineer Senior (Firmware)	48.00	/hour
Engineer (Simulation)	43.00	/hour
Software Phase Duration	4	weeks
Hardware Phase Duration	6	weeks
Sim Development Duration	1	weeks
Total Duration (sequentially)	10	weeks
Total Duration (parallelised)	7	weeks

Table A.2: Simulation Setup - Line Marking

Cost Component	Description	Quantity (weeks)	Unit Cost	Total Cost
Personnel Costs				
Simulations Engineer	Setup, configuration, and distribution of model	1	344.00	1,720.00
Hardware Costs				
High-end Workstation	Existing systems were capable.	1	0	0
Software Costs				
Webots	Free and open source	1	0	0
Total				1,720.00

Table A.3: Total Cost without Sim (constructing physical test)

Item	Description	Duration (weeks)	Cost/Day	Total Cost
Robotics Engineer	Waiting for hardware	6	384.00	11,520.00
Software Development	Developing pathing and waypoint following algorithms	4	384.00	7,680.00
Total			0	19,200.00

Table A.4: Total Cost with Sim (parallel)

Item	Description	Duration (weeks)	Cost/Day	Total Cost
Software Development	Developing pathing and waypoint following algorithms	4	384.00	7,680.00
Sim Development	Total setup costs to configure the digital twin and sim environment	1	344.00	1,720.00
Total			0	9,400.00

Table A.5: Savings

Item	Cost
Total Cost without Sim	19,200.00
Total Cost with Sim	11,120.00
Proposed Savings ($)	8,080.00
Proposed Savings (%)	42 %

A.1.2 Python Controller Scripting

Request access the code file `case_a_line_painter_controller.py`

A.1.3 Line Painter Sim World File

Request access the code file `DevArena.wbt`

A.1.4 Line Painter Proto File

Request access the code file `LinePainter.proto`

A.1.5 Field Line Segment Spreadsheet

The algorithms tested were brought in via a the following provided .csv file for the rugby field to be painted. Line segments are given a start and end coordinate, segment type, and a bearing for the end point to determine the direction to settle ready for the next line.

Table A.6: Line Segment `.csv` file

#	layer	type	startx	starty	centx	centy	endx	endy	bearing	rad
1	1	0	-22500	-49000	0	0	-22500	-45000	90000	0
0	1	0	-22500	-49000	0	0	-22500	-45000	90000	0
2	0	1	-22500	-45000	0	0	-22500	45000	90000	0
3	1	1	-22500	45000	0	0	-22500	46000	90000	0
4	1	1	-22500	46000	0	0	-21500	47000	45000	0
5	1	1	-21500	47000	0	0	-21500	45000	270000	0
6	0	2	-21500	45000	-22500	45000	-22500	44000	180000	1000
7	1	1	-22500	44000	0	0	-23500	44000	180000	0
8	1	1	-23500	44000	0	0	-24500	45000	135000	0
9	1	1	-24500	45000	0	0	-22500	45000	0	0
10	0	1	-22500	45000	0	0	22500	45000	0	0
11	1	1	22500	45000	0	0	23500	45000	0	0
12	1	1	23500	45000	0	0	24500	44000	315000	0
13	1	1	24500	44000	0	0	22500	44000	180000	0
14	0	2	22500	44000	22500	45000	21500	45000	90000	1000
15	1	1	21500	45000	0	0	21500	46000	90000	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
64 Truncated Lines										
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
81	1	1	-20159	46000	0	0	-9159	47000	5194	0
82	1	1	-9159	47000	0	0	-9159	45000	270000	0
83	0	1	-9159	45000	0	0	-9159	39500	270000	0
84	1	1	-9159	39500	0	0	-9159	38500	270000	0
85	1	1	-9159	38500	0	0	-11159	39500	153435	0
86	1	1	-11159	39500	0	0	-9159	39500	0	0
87	0	1	-9159	39500	0	0	9159	39500	0	0
88	1	1	9159	39500	0	0	10159	39500	0	0
89	1	1	10159	39500	0	0	9159	37500	243434	0
90	1	1	9159	37500	0	0	9159	39500	90000	0
91	0	1	9159	39500	0	0	9159	45000	90000	0
92	1	1	9159	45000	0	0	9159	46650	90000	0
93	1	1	9159	46650	0	0	8514	30098	267768	0
94	1	1	8514	30098	0	0	7312	28500	233049	0
95	0	2	7311	28501	0	34000	-7311	28501	126948	9149
96	1	1	-7312	28500	0	0	-7913	29299	126951	0

A.2 Case B - Indoor Navigation

A.2.1 Financial Breakdown - Additional Tables

Table A.7: Known Costs

Cost Factor	Value ($ or Time)	Units
Indoor Robotic Platform	8,000.00	unit
Test Site hireage	200.00	/day
Engineer (Hardware)	45.00	/hour
Engineer (Software)	43.00	/hour
Engineer (Simulation)	43.00	/hour
Engineer (UX/UI)	43.00	/hour
Software Phase Duration	2	weeks
Hardware Phase Duration	3	weeks
Sim Development Duration	1	weeks
Total Duration (sequentially)	5	weeks
Total Duration (parallelised)	5	weeks

Table A.8: Cost Breakdown

Cost Component	Description	Quantity	Unit Cost	Total Cost
Personnel Costs				
Simulations Engineer	Setup, configuration, and distribution of model	0.5	344.00	860.00
Simulations Engineer	Configure simulation indoor environment with QR Codes and document positions.	0.5	344.00	860.00
Hardware Costs				
High-end Workstation	Gaming laptop with an RTX 3060 GPU was required	1	2,300.00	2,300.00
Software Costs				
Nvidia Omniverse	Free	1		
Total				4,020.00

Table A.9: Total Cost without Sim (constructing physical test)

Item	Description	Weeks	$/Day	Total Cost
Robot Platform	Purchase robot mobile base		8,000.00	8,000.00
Software Development	Writing ROS 2 hardware simulated cameras interfaces for the robot	2	344.00	3,440.00
Hardware Development	Implementing hardware interfaces, power systems, and sensors	3	360.00	5,400.00
Total				16,840.00

Table A.10: Total Cost with Sim (parallel)

Item	Description	Duration (weeks)	Cost/Day	Total Cost
Software Development	Developing autonomous code with ROS 2 using sim	1	344.00	1,720.00
Sim Development	Total setup costs to configure the digital twin and sim environment		4,020.00	4,020.00
Total				5,740.00

Table A.11: Savings

Item	Cost
Total Cost without Sim	16,840.00
Total Cost with Sim	5,740.00
Proposed Savings ($)	11,100.00
Proposed Savings (%)	66 %

A.3 Case C - Forestry Tool Carrier

A.3.1 Financial Breakdown - Additional Tables

Table A.12: Cost Factors

Cost Factor	Value	Units
Machine Hire	2,500.00	/week
Engineer	43.00	/hour
Truck / Trailer Hireage	189.00	/day
Project Duration	64	weeks
Field Testing Phase Duration	24	weeks
Avoidable Testing	16	weeks

Table A.13: Simulation Setup - Forestry

Cost Component	Description	Qty.	Unit Cost	Total Cost
Personnel Costs				
Simulations Engineer	Setup, configuration, and digital twin interfacing	10	344.00	3,440.00
Hardware Costs				
High-end Workstation	Existing systems were capable.	1	0	0
Software Costs				
Webots	Free and open source	1	0	0
Total				3,440.00

Table A.14: Total Field Testing Costs

Description	Details	Cost
Machine Rental	64 weeks	160,000.00
Personnel Costs	2 engineers for setup, testing, debugging	44,032.00
Total		204,032.00

Table A.15: Comparison

Description	Details	Cost
Avoidable Field Testing Costs		
Machine Rental + Fuel	4 months (avoidable testing)	43,199.84
Personnel Costs	2 engineers for setup, testing, debugging	2,752.00
Total Field Testing Costs		
		45,951.84
Simulation Costs		
Webots	Free and Open Source	0
Simulation Design and Setup	Building digital twin of the robot and writing simulated hardware interfaces	3,440.00
Hardware	Existing systems were capable	0
Total Simulation Costs		3,440.00
Avoidable Field Costs - Simulation Costs		42,511.84
Remaining Total Field Costs		161,520.16
Estimated Savings		20.84%

Table A.16: Savings

Description	Cost
Total Cost without Sim	204,032.00
Total Cost with Sim	161,520.16
Proposed Savings ($)	42,511.84
Proposed Savings (%)	20.84%

A.3.2 Forestry Webots Controller

Request access the code file `case_a_line_painter_controller.py`

A.4 Case D - Inspection Robot

A.4.1 User README File

```
# Mercury One Robot Simulator README

## Introduction
This simulator package is designed to facilitate the development and testing
    ↪ of navigation and control algorithms for the Mercury One robot within
    ↪ a virtual environment.

```

```
## Prerequisites
Before you begin, ensure you have the following installed on your system:
- [ROS 2](https://docs.ros.org/en/humble/Installation.html) (Humble or newer recommended)
- [Webots](https://cyberbotics.com/doc/guide/installation-procedure) (R2023a or newer recommended)
- [Git](https://git-scm.com/downloads)

## Installation Instructions

### Step 1: Install Webots
Download and install Webots from the [official Webots installation guide](https://cyberbotics.com/doc/guide/installation-procedure).

### Step 2: Set Up ROS 2 Workspace
Create a new ROS 2 workspace if you don't already have one:
```bash
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws
```

Step 3: Clone the Repositories
In the `src` directory of your ROS 2 workspace, clone the required repositories:

- Clone the Mercury One simulator repository:
  ```bash
  git clone https://github.com/CraigMS-dev/mercury_one_sim.git
  ```
- Clone the Webots ROS 2 interface package:
  ```bash
  git clone --recursive-submodules https://github.com/cyberbotics/webots_ros2.git
  ```

Step 4: Build the Workspace
Navigate back to the root of your ROS 2 workspace and build the packages:
```bash
cd ..
colcon build
```

Step 5: Source the Setup Script
To ensure that your environment is using the local versions of the packages you just built, source the setup script:
```bash
source install/local_setup.bash
```
Important: This step is crucial for making sure that the correct version of `webots_ros2` is used.

Running the Simulation
To launch the Mercury One simulator, use the following command:
```bash
ros2 launch mercury_one_sim robot_sim_bringup.launch.py
```

Optional Features

Joystick Control
```

```
If you have a joystick and wish to control the simulated robot using it, you can enable the joystick support by passing the ‘include_joystick:=true‘ argument when launching:

‘‘‘bash
ros2 launch mercury_one_sim robot_sim_bringup.launch.py include_joystick:=true
‘‘‘

### Foxglove Studio Integration
For real-time visualization and debugging, integrate Foxglove Studio with the simulation by setting up a WebSocket connection to stream live data from the simulation. Enable this feature by using the ‘include_foxglove:=true‘ argument:

‘‘‘bash
ros2 launch mercury_one_sim robot_sim_bringup.launch.py include_foxglove:=true
‘‘‘
- Install Foxglove Studio from [Foxglove Studio’s website](https://foxglove.dev/download).
- Set up the Foxglove ROS bridge following the [Foxglove ROS bridge documentation](https://foxglove.dev/docs/studio/ros).

## Utilization
The simulation publishes various topics that can be interfaced with additional navigation packages in ROS 2. You can use these topics to implement and test navigation strategies, sensor integration, and control algorithms.
```

Code block 3: *Snippet from the README simulation user document.*

A.4.2 Financial Breakdown - Additional Tables

Table A.17: Known Costs

Cost Factor	Value	Units
Engineer (Hardware)	45.00	/hour
Engineer (Software)	43.00	/hour
Engineer (Simulation)	43.00	/hour
Engineer (UX/UI)	43.00	/hour
Software Phase Duration	6	weeks
UX/UI Phase Duration	4	weeks
Hardware Phase Duration	5	weeks
Sim Development Duration	2	weeks
Total Duration (sequentially)	15	weeks
Total Duration (parallelised)	8	weeks

Table A.18: Simulation Cost

Cost Component	Description	Qty.	Unit Cost	Total Cost
Personnel Costs				
Simulations Engineer	Setup, configuration, and ROS 2 interfacing	2	344.00	3,440.00
Hardware Costs				
High-end Workstation	Existing systems were capable.	1	0	0
Software Costs				
Webots	Free and open source	1	0	0
Total				3,440.00

Table A.19: Total Cost without Sim (sequential)

Item	Description	Duration (weeks)	Cost/Day	Total Cost
Software Development	Developing autonomous code with ROS 2	11	344.00	18,920.00
UI Development	Developing robot dashboard	9	344.00	15,480.00
Hardware Development	Designing and building the mechanical prototype	5	360.00	9,000.00
Total				43,400.00

Table A.20: Total Cost with Sim (sequential)

Item	Description	Duration (weeks)	Cost/Day	Total Cost
Software Development	Developing autonomous code with ROS 2 using sim	6	360.00	10,800.00
UI Development	Developing robot dashboard using sim	4	344.00	6,880.00
Hardware Development	Designing and building the mechanical prototype	5	120.00	3,000.00
Sim Development	Total setup costs to configure the digital twin and sim environment		3,440.00	3,440.00
Total				24,120.00